

VOICE CLONING

VOICE CLONING

Clone Your Voice in 5 Seconds

Dinda Majesty

Rolly Maulana Awangga

Mohamad Nurkamal Fauzan

Informatics Research Center, Politeknik Pos Indonesia



Kreatif Industri Nusantara

Penulis:

Dinda Majesty
Rolly Maulana Awangga
Mohamad Nurkamal Fauzan

ISBN : xxx-xxx-xxxxx-x-x

Editor:

Rolly Maulana Awangga

Penyunting:

Muhammad Yusril Helmi Setyawan

Desain sampul dan Tata letak:

Dinda Majesty

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2
Bandung 40191
Tel. 022 2045-8529
Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center
Jl. Sariasisih No. 54
Bandung 40151
Email : irc@poltekpos.ac.id

Cetakan Pertama, 2021

Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin tertulis dari penerbit

*'Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu harus
sanggup menahan
perihnya Kebodohan.'*

Imam Syafi'i

CONTRIBUTORS

DINDA MAJESTY, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

MUHAMMAD NURKAMAL FAUZAN, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

MUHAMMAD YUSRIL HELMI SETYAWAN, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

CONTENTS IN BRIEF

1 Sistem Text to Speech (TTS)	1
2 Neural Network	7
3 Sequence to Sequence (Seq2Seq)	13
4 Speech Synthesis	23
5 Voice Cloning	33
6 Tacotron-2	43
7 WaveNet	49
8 Multi-Speaker Voice Cloning	63
9 Tutorial Pembuatan Multi-Speaker Voice Cloning	69
10 Tutorial Pembuatan Project Menggunakan Google Colab	141
11 Result, Error, and Solution	171

DAFTAR ISI

Daftar Gambar	xiii
Foreword	xxv
Kata Pengantar	xxvii
Acknowledgments	xxix
Acronyms	xxxi
Glossary	xxxiii
List of Symbols	xxxv
Introduction	xxxvii

Dinda MajestyRolly Maulana AwanggaMohamad Nurkamal Fauzan

1 Sistem Text to Speech (TTS)	1
1.1 Text to Speech (TTS)	1
1.1.1 Artificial Intelligence	2
1.1.2 Machine Learning	3
1.1.3 Deep Learning	3
1.2 Cara Kerja TTS	4

2 Neural Network	7
2.1 Neural Network	7
2.1.1 Sejarah Neural Network	9
2.1.2 Struktur Neuron Pada Otak Manusia	10
2.1.3 Struktur Neural Network	11
2.1.4 Cara Kerja Neural Network	11
3 Sequence to Sequence (Seq2Seq)	13
3.1 Model Sequence to Sequence	13
3.1.1 Arsitektur Encoder-Decoder	15
3.1.2 Kekurangan Model Encoder-Decoder	19
3.1.3 Sequence to Sequence dengan Attention	20
3.1.4 Rumusan Masalah untuk Pemodelan Sequence to Sequence	21
4 Speech Synthesis	23
4.1 Speech Synthesis	23
4.1.1 Sejarah Speech Synthesis	25
4.1.2 Perangkat Speech Synthesis	26
4.1.3 Teknologi Speech Synthesis	27
5 Voice Cloning	33
5.1 Voice Cloning	33
5.1.1 Manfaat Voice Cloning	34
5.1.2 Aplikasi Voice Cloning Terbaik	35
5.1.3 Pembuatan Voice Cloning	36
5.1.4 Dampak Negatif Voice Cloning	37
5.1.5 Dampak Positif Voice Cloning	39
5.1.6 Mendeteksi Deepfake Voice	40
6 Tacotron-2	43
6.1 Attention Based Model untuk Speech Recognition	44
6.1.1 Decoder	45
6.1.2 Loss Function	45
7 WaveNet	49
7.1 Talking Machine	51
7.2 WaveNet Improving State of The Arts	52

7.3	Wavenet. The Generative Model	53
7.4	CNN	54
7.5	Softmax Distribution. Mu-law Companding	56
7.6	Gated Activations. Skip and Residual Connections.	56
7.7	Conditioning. Local. Global.	57
7.8	Great Model. Fast Training. Slow Inference?	58
7.9	Normalising Flows. IAF.	58
7.10	Parallel. Faster. WaveNet.	61
8	Multi-Speaker Voice Cloning	63
8.1	Model SV2TTS	63
8.1.1	Speaker Encoder	64
8.1.2	Synthesizer	65
8.1.3	Vocoder	66
8.1.4	Alur Model SV2TTS	66
8.1.5	Mengukur Kualitas Hasil	67
9	Tutorial Pembuatan Multi-Speaker Voice Cloning	69
9.1	Pengenalan Anaconda	69
9.1.1	Instalasi Anaconda 3 Windows 10 x64	69
9.1.2	Update Anaconda dan Spyder	76
9.1.3	Instalasi Anaconda Ubuntu 19.04	77
9.1.4	Konfigurasi <i>Python</i>	80
9.2	Instalasi Pycharm	81
9.3	Membuat Akun Github	84
9.4	Download dan Install Git	88
9.5	Konfigurasi Git	95
9.6	Fork dan Clone Voice Cloning Repository	98
9.7	Menambahkan Interpreter ke dalam Voice Cloning Project	102
9.7.1	Download dan Install Python 3.7	102
9.7.2	Download dan Install FFmpeg	104
9.7.3	Tutorial Membuat Virtual Environment	109
9.7.4	Tutorial Membuat Conda Environment	113
9.7.5	Install Requirements	116
9.7.6	Download and install Pytorch dengan Anaconda	120
9.7.7	Download and install Pytorch dengan Pip	123
9.8	Run Project Menggunakan Pycharm	126
10	Tutorial Pembuatan Project Menggunakan Google Colab	141

10.1	Membuat Akun Google	142
10.2	Membuat New Project Pada Google Colab	147
11	Result, Error, and Solution	171
11.1	Pembahasan Hasil dari Proses	171
11.2	Demo Aplikasi Voice Cloning	186
11.3	Pembahasan Error dan Solusi	191
	Daftar Pustaka	193

DAFTAR GAMBAR

1.1	Korelasi antara Deep Learning, Machine Learning, dan Artificial Intelligence	2
1.2	Perbedaan Machine Learning dan Deep Learning	4
1.3	Cara Kerja Text to Speech (TTS)	5
2.1	Sel Saraf (Neuron)	8
2.2	McCulloch dan Pitts, penemu pertama Neural Network	9
3.1	Contoh Penggunaan Model Sequence to Sequence	14
3.2	Google Translate	14
3.3	Speech Recognition	15
3.4	Video Captioning	15
3.5	Rumus Hidden State	16
3.6	Encoder	16
3.7	Rumus Hidden State	17

3.8	Rumus Hidden State	17
3.9	Decoder	18
3.10	Arsitektur Encoder-Decoder Secara Keseluruhan	18
3.11	Chatbot dengan Seq2Seq Model	19
3.12	Gambar to Text	21
3.13	Mekanisme Attention pada Gambar to Text	21
4.1	Speech Synthesis	24
4.2	Stephen Hawking adalah salah satu orang paling terkenal yang menggunakan sintesis ucapan untuk berkomunikasi	26
5.1	Voice Cloning	34
5.2	Contoh Aplikasi Voice Cloning (SV2TTS Toolbox)	35
5.3	Spoofing Voice Biometrics	38
5.4	Voice Phising, Sumber: https://id.pinterest.com/pin/195765915039905230/	38
5.5	Smart Assistant, Sumber: https://blog.evolvemachinelearners.com/voice-assistant-its-history-twist-and-turn/	40
6.1	Blok Diagram dari Arsitektur Tacotron 2	44
6.2	Spektogram mel yang diprediksi : Seperti yang dapat Anda bandingkan dengan wilayah atas, ia memiliki banyak celah dan masih membutuhkan banyak pelatihan. Sisi kanan (hijau solid) hanya padding dalam satu batch.	46
6.3	Target mel-spektogram	46
6.4	Perhatian (Seperti yang Anda lihat di sisi kiri bawah, sepertinya sedang belajar untuk menyelaraskan tetapi masih membutuhkan sekitar satu minggu pelatihan untuk mendapatkan diagonal yang sempurna untuk perhatian)	46
7.1	Perbandingan WaveNet dengan Model Lain	52
7.2	Probabilitas Gabungan	53
7.3	Konvolusi Kausal untuk memastikan bahwa model tidak dapat melanggar urutan di mana kita memodelkan data.	54
7.4	Tumpukan lapisan konvolusi kausal yang melebar: Menggandakan faktor pelebaran dengan setiap lapisan menghasilkan pertumbuhan reseptif $O(2^n)$.	55

7.5	Ekspresi untuk melakukan kompilasi Mu-law menghasilkan output yang direkonstruksi lebih baik (terdengar mirip dengan audio asli) daripada kuantisasi linier.	56
7.6	Ekspresi untuk fungsi aktivasi terjaga keamanannya digunakan di WaveNet.	57
7.7	Arsitektur WaveNet: menampilkan fungsi aktivasi yang terjaga keamanannya, melewatkkan koneksi dan koneksi residual.	57
7.8	Modifikasi istilah distribusi bersyarat, setelah pengenalan input pengkondisian.	58
7.9	Ekspresi setelah pengenalan istilah bias h .	58
7.10	Distribusi mengalir melalui urutan transformasi yang dapat dibalik.	59
7.11	Transformasi skala dan pergeseran sederhana pada z_t di mana faktor penskalaan (s) dan faktor pergeseran (μ) dihitung dengan menggunakan parameter yang dapat dipelajari dan nilai dalam sampel input z dari langkah waktu sebelumnya.	60
7.12	Dalam Inverse Autoregressive Flow, output pada langkah waktu yang berbeda dapat dihitung secara paralel karena output dari langkah waktu tidak bergantung pada output dari langkah waktu sebelumnya.	60
7.13	Prosedur pelatihan Paralel WaveNet.	61
8.1	Arsitektur SV2TTS umum Sumber: Jia, Zhang, dan Weiss et al.	63
8.2	Representasi visual dari embeddings, setiap warna adalah speaker yang berbeda[1]	64
8.3	Model Arsitektur Multi-Speaker Voice Cloning [1]	65
8.4	Training Synthesizer[1]	66
8.5	Vocoder[1]	66
8.6	Pengukuran Kualitas Model[1]	67
8.7	MOS [1]	68
8.8	Speech Similarity[1]	68
9.1	Run Setup Anaconda	70
9.2	Setup Loading	70
9.3	Welcome to Anaconda Setup	71

9.4	<i>License Agreement</i>	72
9.5	<i>Just Me(recomended)</i>	72
9.6	<i>Pilih lokasi</i>	73
9.7	<i>Centang Anaconda to my PATH</i>	74
9.8	<i>Waiting Installation Complete</i>	74
9.9	<i>Installation Complete</i>	75
9.10	<i>Anaconda+JetBrains</i>	75
9.11	<i>Thanks for install Anaconda</i>	76
9.12	Gambar halaman download	77
9.13	Gambar install anaconda	77
9.14	Gambar eksekusi anaconda	78
9.15	Gambar anaconda license agreement	78
9.16	Gambar perintah yes or no	79
9.17	Gambar path anaconda	79
9.18	Gambar perintah install spyder3	80
9.19	Gambar setpath	81
9.20	Klik Next untuk Install Pycharm	82
9.21	Menentukan lokasi folder dan klik next	82
9.22	Add launchers dir to PATH	83
9.23	Klik Install	83
9.24	Reboot Now	84
9.25	Sign Up sumber: https://omcyber.com/cara-membuat-akun-github/	85
9.26	Form Daftar	85
9.27	Free Plan	86
9.28	Verifikasi Email	86
9.29	Verifikasi Email Address	87
9.30	Verifikasi Email Berhasil	87
9.31	Install Git	88

9.32	Lokasi Instalasi Git	89
9.33	Komponen Tambahan	89
9.34	Select Start Menu Folder	90
9.35	Choose Default Editor	90
9.36	PATH Environment	91
9.37	Choose SSH	91
9.38	Configuring the line ending	92
9.39	Configuring the terminal emulator	92
9.40	Configuring extra options	93
9.41	Install	93
9.42	Proses Install	94
9.43	Command Prompt (CMD)	94
9.44	Versi Git	94
9.45	Konfigurasi Username Git	95
9.46	Konfigurasi Email Git	95
9.47	Generate SSH key	96
9.48	Direktori SSH	96
9.49	SSH Key	96
9.50	Menu SSH dan GPG keys	97
9.51	New SSH key	97
9.52	Add SSH key	98
9.53	Fork Repotori	98
9.54	Klik Tombol Code	99
9.55	Copy Url	99
9.56	Git Clone Repotori	100
9.57	Proses Clone Repotori	100
9.58	Pycharm	101
9.59	Lokasi Folder Repotori	101

9.60	Pop Up Create Virtual Environment	102
9.61	Real-Time-Voice-Cloning project	102
9.62	Download Python 3.7	103
9.63	Add Python to PATH	103
9.64	Success Install Python 3.7	104
9.65	Check Python Version	104
9.66	Download FFmpeg	105
9.67	Download FFmpeg for Windows	105
9.68	Extract File Instalasi	106
9.69	Ganti Nama Folder	106
9.70	Move Folder	107
9.71	Move Folder	107
9.72	Environment Variables	108
9.73	Add FFmpeg to PATH	108
9.74	Save Changes	109
9.75	Check FFmpeg Version	109
9.76	Menu Settings Python Interpreter	110
9.77	Python Interpreter	110
9.78	Add Python Interpreter	111
9.79	Creating Virtual Environment	111
9.80	Pilih Virtual Environment	112
9.81	Apply Virtual Environment	112
9.82	Inisialisasi Virtual Environment	113
9.83	Menu Settings Project	113
9.84	Show All Interpreter	114
9.85	Conda Environment	114
9.86	Create New Conda Environment	115
9.87	Creating Conda Environment	115

9.88	Apply Conda Environment	116
9.89	Progress Bar Inisialisasi Conda Environment	116
9.90	Activate Conda Environment	117
9.91	Install Pip using Anaconda Prompt	117
9.92	Install Requirements using Anaconda Prompt	118
9.93	Sukses Install Requirements pada Conda Environment	118
9.94	Activate Virtual Environment with Command Prompt	119
9.95	Install Requirements using Command Prompt	119
9.96	Berhasil Install Requirement pada Virtual Environment	120
9.97	Select Pytorch with Conda	121
9.98	Install Pytorch with Anaconda Prompt	121
9.99	Proses Install Pytorch with Anaconda Prompt	122
9.100	Instalasi Pytorch Selesai	122
9.101	Check Instalasi Pytorch	123
9.102	Import Pytorch	123
9.103	Print Pytorch Version	123
9.104	Select Pytorch with Pip	124
9.105	Install Pytorch with Pip	124
9.106	Install Pytorch Berhasil	125
9.107	Check Version Pytorch	125
9.108	Import Torch	126
9.109	Print Torch Version	126
9.110	<i>Preprocessing Speaker Encoder Model (Pycharm)</i>	130
9.111	<i>Training Speaker Encoder Model (Pycharm)</i>	131
9.112	<i>Struktur Folder Dataset Synthesizer</i>	133
9.113	<i>Preprocessing Audio (Pycharm)</i>	134
9.114	<i>Preprocessing Audio Embeddings (Pycharm)</i>	135
9.115	<i>Training Synthesizer Model (Pycharm)</i>	136

9.116	<i>Preprocessing Vococder Model (Pycharm)</i>	137
9.117	<i>Training Vococder Model (Pycharm)</i>	139
10.1	<i>Create Google Account</i>	142
10.2	Isi form pendaftaran	143
10.3	Isikan Nomor Handphon	143
10.4	<i>Verify Phone Number</i>	144
10.5	<i>Form Personal Data</i>	145
10.6	<i>Get more from your number</i>	146
10.7	<i>Create Google Account</i>	146
10.8	<i>Create Google Account</i>	147
10.9	<i>Create New Project</i>	148
10.10	Rename File	149
10.11	Script Mounting Google Drive	150
10.12	<i>Add Text to Project</i>	150
10.13	<i>Change Directory</i>	151
10.14	<i>Script Clone Repository</i>	152
10.15	<i>Check Folder</i>	153
10.16	<i>Struktur Dataset</i>	157
10.17	<i>Preprocessing Speaker Encoder Model</i>	157
10.18	<i>Training Speaker Encoder Model</i>	159
10.19	<i>Struktur Folder Dataset Synthesizer</i>	161
10.20	<i>Preprocessing Audio</i>	162
10.21	<i>Preprocessing Audio Embeddings</i>	163
10.22	<i>Training Synthesizer Model</i>	165
10.23	<i>Preprocessing Vococder Model</i>	167
10.24	<i>Training Vococder Model</i>	169
11.1	<i>Struktur Folder Dataset untuk Preprocessing Speaker Encoder</i>	172
11.2	<i>Hasil Preprocessing Speaker Encoder</i>	172

11.3	<i>Folder Hasil Preprocessing Speaker Encoder</i>	173
11.4	<i>File Numpy Array Hasil Preprocessing Speaker Encoder</i>	173
11.5	<i>Log TITML-IDN</i>	174
11.6	<i>Log Common Voice Indonesia</i>	174
11.7	<i>Hasil Training Speaker Encoder Model</i>	175
11.8	<i>File Hasil Training Speaker Encoder Model</i>	175
11.9	<i>UMAP Projection in 79900 steps</i>	176
11.10	<i>File Speaker Encoder Model</i>	176
11.11	<i>Struktur Dataset untuk Preprocessing Audio dan Embeds</i>	177
11.12	<i>Hasil Preprocessing Audio</i>	177
11.13	<i>File Numpy Array dari Salah Satu Folder</i>	178
11.14	<i>Data Pada File train.txt</i>	178
11.15	<i>File Numpy Array pada Folder embeds</i>	179
11.16	<i>Hasil Proses Training Synthesizer</i>	180
11.17	<i>Folder Hasil Proses Training Synthesizer</i>	180
11.18	<i>Folder Wav</i>	181
11.19	<i>Folder Plots</i>	181
11.20	<i>File CharacterEmbeddings.tsv</i>	182
11.21	<i>Folder Mel-Spectrogram</i>	182
11.22	<i>Mel-Spectrogram</i>	183
11.23	<i>Preprocessing Vocoder</i>	183
11.24	<i>Hasil Preprocessing Vocoder</i>	184
11.25	<i>Folder mels-gta</i>	184
11.26	<i>File Synthesized</i>	185
11.27	<i>Training Vocoder</i>	185
11.28	<i>Hasil Generate Audio</i>	186
11.29	<i>Folder Models</i>	186
11.30	<i>Link Akses</i>	187
11.31	<i>Link Akses</i>	187
11.32	<i>Demo Notebook</i>	188
11.33	<i>Record or Upload Audio Sample</i>	189
11.34	<i>Synthesize and Generate Audio</i>	190

Listings

9.1	Config Dataset (Pycharm)	126
9.2	Preprocessing Function (Pycharm)	127
9.3	Preprocessing Encoder Model (Pycharm)	128
9.4	Script to Run Preprocessing Speaker Encoder Model (Pycharm)	130
9.5	Training Speaker Encoder Model (Pycharm)	130
9.6	Script to Run Training Speaker Encoder Model	131
9.7	Training Speaker Encoder Model (Pycharm)	132
9.8	Preprocessing Audio (Pycharm)	133
9.9	Training Speaker Encoder Model (Pycharm)	134
9.10	Preprocessing Embeds (Pycharm)	134
9.11	Training Speaker Encoder Model (Pycharm)	135
9.12	Training Synthesizer Model (Pycharm)	136
9.13	Vocoder Preprocess (Pycharm)	136
9.14	Preprocessing Vococder Model (Pycharm)	137
9.15	Vocoder Train (Pycharm)	138
9.16	Training Vococder Model (Pycharm)	139
10.1	Mounting Google Drive	149
10.2	Change Directory	151

10.3	Clone Repository	151
10.4	Config Dataset	153
10.5	Preprocessing Function	154
10.6	Preprocessing Encoder Model	155
10.7	Script to Run Preprocessing Speaker Encoder Model	157
10.8	Training Speaker Encoder Model	158
10.9	Script to Run Training Speaker Encoder Model	159
10.10	Training Speaker Encoder Model	159
10.11	Preprocessing Audio	161
10.12	Training Speaker Encoder Model	162
10.13	Preprocessing Embeds	163
10.14	Training Speaker Encoder Model	163
10.15	Training Synthesizer Model	164
10.16	Vocoder Preprocess	165
10.17	Preprocessing Vocoder Model	166
10.18	Vocoder Train	167
10.19	Training Vocoder Model	168
11.1	Beberapa Data Pada File train.txt	178
11.2	Setup Project	188
11.3	Record or Upload Audio Sample	189
11.4	Synthesize dan Generate Audio Output	190

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini berisi penjelasan teknologi serta tata cara yang harus dilakukan dalam pembuatan *voice cloning* berbahasa indonesia. Buku ini diharapkan dapat membantu orang-orang memahami teknologi terkini terkait *voice cloning* dan cara kerjanya, hal yang dibutuhkan dalam pembuatan *voice cloning* serta cara membuat voice cloning.

TIM PENULIS

Bandung, Jawa Barat

Desember, 2021

ACKNOWLEDGMENTS

Terima kasih atas semua masukan dari para dosen D4 Teknik Informatika Politeknik Pos Indonesia agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk dosen pembimbing yang telah membimbing dan membantu saya menyelesaikan pembuatan buku ini sebagai syarat kelulusan Internship.

D. M.

ACRONYMS

Seq2Seq	Sequence to Sequence
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
TTS	Text to Speech
SV2TTS	Speaker Verification to Multispeaker Text-To-Speech Synthesis
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
DNN	Deep Neural Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit
GAN	Generative Adversarial Network
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
PC	Personal Computer

OS	Operating System
SSH	Secure Shell
CMD	Command Prompt

GLOSSARY

Text to Speech	Merupakan proses pembuatan suara melalui inputan berupa text.
Speech Synthesis	Merupakan kemampuan berbicara buatan (artificial) yang mirip seperti manusia.
Voice Cloning	Merupakan sistem yang mampu menghasilkan suara tiruan yang memiliki kealamian dan kemiripan ucapan dengan sampel.
Seq2Seq	Merupakan arsitektur jaringan saraf berulang yang biasa digunakan untuk memecahkan masalah bahasa yang kompleks.
Tacotron-2	Merupakan arsitektur rancangan Google untuk menghasilkan mel-spectrogram yang digunakan dalam pembuatan audio.
WaveNet	Merupakan model yang dapat menghasilkan audio berdasarkan mel-spectrogram.

SYMBOLS

A Amplitude

$\&$ Propositional logic symbol

a Filter Coefficient

B Number of Beats

INTRODUCTION

DINDA MAJESTY

ROLLY MAULANA AWANGGA

MOHAMAD NURKAMAL FAUZAN

Informatics Research Center, Politeknik Pos Indonesia
Bandung, Jawa Barat, Indonesia

Perkembangan teknologi yang semakin maju dalam sistem *Text to Speech (TTS)* memungkinkan manusia membuat kloningan suara hanya dengan beberapa detik sampel suara. Hasil kloningan suara dapat dimodifikasi sesuai dengan inputan teks, sehingga kloningan suara tersebut akan mengucapkan kalimat yang sesuai dengan teks yang diinputkan. Suara yang dihasilkanpun terdengar alami mirip seperti suara manusia aslinya.

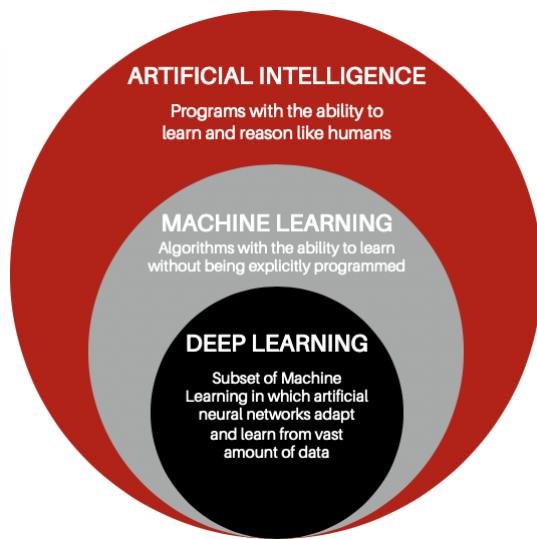
BAB 1

SISTEM TEXT TO SPEECH (TTS)

Sebelum mengetahui teknologi apa saja yang digunakan serta tata cara pembuatan sistem voice cloning ini, akan lebih baik apabila kita memahami terlebih dahulu mengenai sistem *Text to Speech (TTS)*.

1.1 Text to Speech (TTS)

Text to Speech merupakan proses pembuatan suara melalui inputan berupa text[2]. mungkin teman-teman bingung, bagaimana bisa text berubah menjadi suara, semestinya text itu hanya dapat kita lihat menggunakan panca indra kita yaitu mata sedangkan suara merupakan sesuatu yang tidak bisa kita lihat menggunakan mata tetapi bisa kita dengar menggunakan telingga. Text to Speech ini ada untuk menjawab kebingungan teman-teman tersebut. Di zaman sekarang ini, dimana teknologi sudah semakin canggih, mengubah teks menjadi suara bukanlah hal yang sulit untuk dilakukan, apalagi setelah teman-teman mengenal deep learning, machine learning, dan artificial intelligence. Bagi teman-teman yang masih belum kenal, yuk kenalan dulu.



Gambar 1.1 Korelasi antara Deep Learning, Machine Learning, dan Artificial Intelligence

1.1.1 Artificial Intelligence

Artificial Intelligence (AI) merupakan kecerdasan yang dibuat dan diberikan atau diterapkan pada program komputer yang diharapkan memiliki kecerdasan seperti manusia sehingga memiliki kemampuan memahami, merencanakan, mengambil keputusan, dan problem solving. Dalam pembuatannya, setiap algoritma memungkinkan mesin untuk meniru, mengembangkan, dan berprilaku seperti manusia. Sederhananya AI itu merupakan sebuah teknologi yang memungkinkan kita menciptakan sebuah mesin yang memiliki kecerdasan berdasarkan pada perintah atau algoritma yang kita berikan pada program tersebut. Algoritma berupa langkah-langkah yang kita lakukan dalam menyelesaikan suatu permasalahan. Misalnya kita ingin menciptakan mesin yang dapat menjawab semua pertanyaan kita seperti chatbot, maka kita membutuhkan algoritma atau langkah-langkah apa yang dilakukan oleh chatbot untuk memahami dan mampu memberikan respon secara cepat dan tepat kepada kita.

Artificial Intelligence dibuat dengan 3 tujuan, diantaranya:

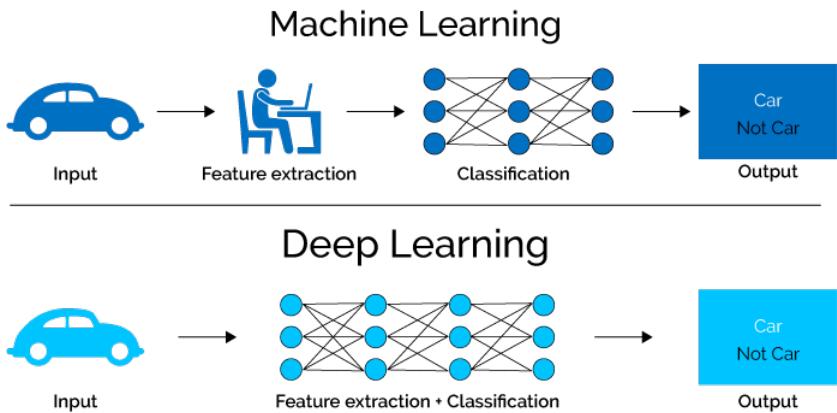
1. Tujuan utama, Membuat mesin menjadi lebih pintar dan terus mengalami peningkatan seiring dengan perkembangan teknologi.
2. Tujuan ilmiah, Memahami apa itu kecerdasan buatan sehingga dapat dimanfaatkan untuk membuat mesin yang lebih cerdas dalam membantu dan memecahkan masalah secara efektif dan efisien.
3. Tujuan entrepreneurial, Membuat mesin lebih bermanfaat sehingga memudahkan manusia dalam melakukan perkerjaan dan memecahkan masalah secara cepat, tepat, dan lebih teliti.

1.1.2 Machine Learning

Machine Learning (ML) merupakan bagian dari Artificial Intelligence yang ditambahkan dengan metode-metode statistika dengan tujuan agar mesin dapat meningkatkan pengalamannya berdasarkan kepada data-data yang ada dan dilatih pada mesin tersebut. Dengan begitu mesin dapat melakukan tugas tertentu berdasarkan data dan algoritma yang diterapkan padanya. Data dan algoritma yang dirancang membuat mesin terus mengalami peningkatan dari waktu ke waktu. Sebagai contoh, pada pembuatan sistem untuk memprediksi tweet yang bersifat positif, negatif, atau netral kita akan membutuhkan data tweet dari para pengguna twitter, kita bisa dapatkan melalui API yang telah disediakan oleh twitter. Pada tahap pertama, kita mengambil 100 data tweet yang terdiri dari tweet positif, negatif, dan netral. Lalu kita melatih mesin menggunakan 100 data tersebut dan menggunakan algoritma naive bayes untuk mengetahui apakah data hasil prediksi kita sama dengan data tweet aslinya dan menghitung akurasi dari algoritma yang telah kita buat. Pada tahap pertama akurasi menunjukkan angka 90% dan ketika kita melakukan training kembali kepada algoritma atau model yang telah kita buat dengan menggunakan data yang lebih banyak lagi maka mesin akan menjadi lebih terlatih dan kemungkinan kesalahan prediksi menjadi sangat kecil bahkan akurasi yang didapatkan dari model bisa mencapai 99% atau bahkan kita bisa mengganti algoritma yang kita gunakan menggunakan algoritma lainnya yang menghasilkan akurasi yang lebih baik dari model atau algoritma kita sebelumnya. Hal inilah yang dimaksud dengan machine learning, ada data dan algoritma serta perhitungan-perhitungan yang sering kita jumpai dalam statistika seperti perhitungan akurasi, rata-rata (mean), median, recall, precision, dll.

1.1.3 Deep Learning

Setelah mengenal AI dan ML saatnya kita mengenal Deep Learning (DL) yang merupakan bagian dari Machine Learning yang berkaitan dengan algoritma yang berdasarkan pada struktur dan fungsi otak atau sering dikenal sebagai jaringan saraf tiruan. Pada dasarnya Deep Learning memiliki konsep seperti Machine Learning hanya saja dalam konteks yang lebih dalam. Contohnya, pada deep learning kita ingin membuat mesin dapat mengetahui apakah hewan yang ada digambar merupakan seekor anjing atau kucing, maka kita akan membuat algoritma untuk melakukan pengecekan seperti melakukan pengecekan pada bentuk ekor, bentuk telinga, apakah memiliki kumis atau tidak, dan ciri-ciri lainnya yang membuat kucing dan anjing terlihat berbeda tanpa kita perlu memberikan fitur pembedanya atau fitur mana yang lebih penting untuk dapat mengidentifikasi hewan tersebut secara manual. Deep Learning dapat mengetahui fitur tersebut tanpa kita beritahu secara manual seperti pada Machine Learning seperti terlihat pada gambar 1.2. Oleh karena itu Deep Learning dianggap sebagai otak utama yang menciptakan kecerdasan buatan yang lebih manusiawi.



Gambar 1.2 Perbedaan Machine Learning dan Deep Learning

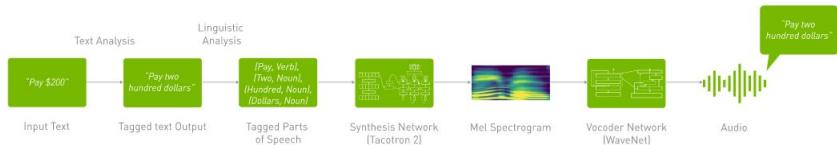
TTS termasuk bagian dari Deep Learning lebih tepatnya Neural Network[3]. Neural Network merupakan cabang ilmu yang mengadopsi cara berpikir dan cara bekerja otak manusia yang memberikan rangsangan/stimulus (input), melakukan proses, dan menghasilkan sesuatu (output). Output dihasilkan dari inputan yang diberikan dan proses yang dilakukan oleh otak manusia. Kemampuan otak untuk memproses inputan atau informasi ini yang akan membuat otak manusia selalu mempelajari hal-hal baru dan terus berkembang, begitupun dengan mesin yang diberikan kecerdasan buatan atau AI. Contohnya saja ketika kita berusia 5 tahun dan otak kita belum memahami informasi-informasi yang sulit seperti pelajaran matematika dasar, menambah dan mengurangi ataupun membagi dan mengalikan angka. Setelah kita mempelajarinya selama 6 tahun di Sekolah Dasar maka otak kita sekarang bisa memahaminya bahkan hal tersebut merupakan hal yang sangat gampang. Pada dasarnya konsep AI sama dengan otak manusia yang selalu berkembang dan mempelajari hal-hal yang baru berdasarkan inputan atau informasi apa yang diberikan kepada otak dan otak akan memproses inputan tersebut hingga dapat memahami dan mengeluarkan hasil dari proses tersebut (output). Dalam pembuatan voice cloning ini kita akan mengajarkan kepada mesin cara membaca text dan mengucapkannya dengan menggunakan suara seperti yang kita contohkan, oleh karena itu kita membutuhkan sistem text to speech.

1.2 Cara Kerja TTS

Cara kerja TTS terdiri dari 2 proses, dapat dilihat pada gambar 1.3:

1. Model Teks ke Spektrogram, model ini mengubah teks menjadi fitur yang selaras dengan waktu seperti spektrogram, mel-spektrogram, atau frekuensi F0 dan fitur linguistik lainnya. Ada beberapa model yang bisa digunakan untuk mengubah teks menjadi spektrogram, diantaranya yaitu Tacotron, Tacotron2, Deep Voice 3, dll. Model ini juga disebut sebagai model encoder-decoder.

2. Model spektrogram ke audio, model ini akan mengonversi spektrogram yang dihasilkan menjadi audio. Model ini disebut sebagai vocoder. Beberapa model yang banyak digunakan untuk membuat vocoder yaitu WaveGlow, Griffin-Lim Algorithm, WaveNet, dll. Model Tacotron-2 sebagai model encoder-decoder dan model WaveNet yang dimodifikasi sebagai vocoder akan dibahas secara mendalam pada buku ini.



Gambar 1.3 Cara Kerja Text to Speech (TTS)

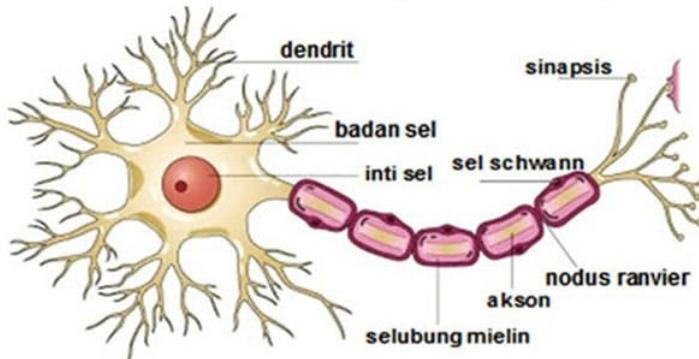
BAB 2

NEURAL NETWORK

2.1 Neural Network

Neural Network terinspirasi dari cara kerja neuron yang ada pada otak manusia, neuron bertugas sebagai penerima stimulus/rangsangan dan pengirim informasi yang akan diolah atau diproses oleh otak menjadi suatu output. Neuron merupakan sistem saraf pusat dan terdapat sekitar 100 miliar neuron yang ada pada tubuh manusia. Perhatikan gambar 2.1

Sel Saraf (Neuron)

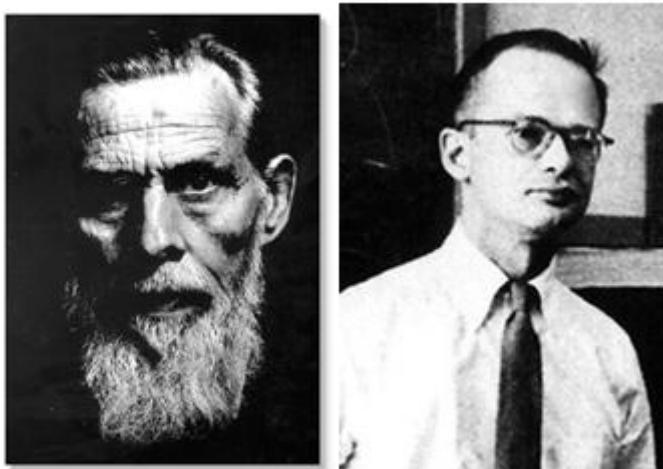


Gambar 2.1 Sel Saraf (Neuron)

Neuron terdiri dari 3 bagian yaitu akson (akar), soma (batang), dan dendrite (cabang). Neuron juga dibedakan menjadi 3 yaitu neuron sensorik atau sel saraf indera karena fungsinya yang berhubungan dengan penerima (indra) dan saraf pusat (otak dan sumsum tulang belakang). Neuron motorik atau sel saraf penggerak yang berfungsi membawa rangsangan dari saraf pusat (otak dan sumsum tulang belakang) ke otot. Terakhir yaitu neuron asosiasi atau sel saraf penghubung, sel ini menghubungkan atau meneruskan rangsangan dari sel saraf sensorik ke sel saraf motorik.

Tiap neuron yang ada dalam otak kita saling terhubung dan berkirim informasi atau rangsangan berupa neurotransmitter, neuron yang saling berinteraksi dengan mengirimkan rangsangan ini akan menghasilkan kemampuan tertentu pada kerja otak kita. Contohnya, ketika kita bertemu dengan seorang kenalan yang menyapa kita maka otak akan berkerja sehingga dapat mengenali orang tersebut dan akan menghasilkan respon atau output, outputnya bisa berupa sapaan, lambaian tangan, obrolan, dan hal-hal lainnya yang biasa kita lakukan apabila bertemu dengan seseorang yang kita kenal. Jadi secara ilmiahnya, inputan atau rangsangan yang diterima yaitu berupa sapaan dari orang yang dikenali. Rangsangan ini akan diterima oleh alat indra (mata, telinga, hidung, lidah, dan kulit) melalui sel reseptör, kemudian akan diteruskan dalam bentuk impuls berupa arus listrik yang diteruskan ke sel saraf sensorik melalui sinapsis, lalu melewati sel saraf koneksi menuju otak. Pada otak informasi akan diolah terlebih dahulu kemudian dikirimkan ke sel saraf motorik dan memberikan atau menghasilkan reaksi berupa sapaan, gerakan berupa lambaian tangan dll. Inilah proses kerja otak manusia begitupun dengan neural network yang terinspirasi dari cara neuron bekerja pada otak manusia. Siapa sangka mesin yang merupakan benda mati akan bisa berprilaku dan berkerja seperti layaknya manusia. Hingga saat ini sangat banyak diminati dan terus dikembangkan dengan tujuan dapat membantu memudahkan manusia dalam bekerja.

2.1.1 Sejarah Neural Network



Gambar 2.2 McCulloch dan Pitts, penemu pertama Neural Network

Neural Network bermula ketika Warren McMulloch yang merupakan seorang neurofisiologi dan Walter Pitts yang merupakan seorang ahli matematika menulis makalah tentang cara kerja dari neuron pada tahun 1943. Kemudian diperkuatnya konsep neuron dalam buku yang ditulis oleh Donald Hebb pada tahun 1949 yang berjudul *The Organization of Behavior* yang menunjukkan bahwa jaringan syaraf akan bertambah kuat setiap kali digunakan. Kemudian dilanjutkan dengan penelitian di IBM untuk mensimulasikan neural network tahun 1950 yang dipimpin oleh Nathania Rochester.

Dengan diadakannya konferensi Dartmouth pada tahun 1956 yang membahas tentang penelitian neural network oleh John McCarthy memperkuat konsep mengenai neural network. Lalu pada tahun 1957, John Von Neumann menyarankan untuk meniru fungsi neuron menggunakan relay telegraf atau tabung vakum. Dengan adanya penemuan two-layer-network yang dikenal dengan perceptron. Perceptron berguna untuk menghitung jumlah input, mengurangi treshold, dan meneruskan salah satu dari dua nilai yang mungkin keluar sebagai hasil, penemuan ini ditemukan oleh Frank Rosenblatt pada tahun 1958.

Pada tahun 1959, diperkenalkan model neural network pertama yang dikenal dengan ADALINE (Adaptive Linear Elements) dan MEDALINE (Multiple Adaptive Linear Elements). Model ini merupakan model pertama yang diterapkan pada permasalahan yang ada di dunia nyata. Model ini berfungsi untuk menghilangkan gema pada saluran telepon. Pengembangan model ini dilakukan oleh Bernard Widrow dan Marcian Hoff dari Stanford.

Pada tahun 1982 dalam makalah yang dipresentasikan pada National Academy of Sciences tentang pendekatan untuk menciptakan perangkat yang berguna, menyenangkan, pandai berbicara dan kharismatik, makalah ini dipresentasikan oleh John

Hopfield. Lalu konferensi International Institute of Electrical and Electronics (IEEE) mengadakan konferensi mengenai Neural Network yang dihadiri oleh lebih dari 1.800 peserta pada tahun 1987. Sekarang, Neural Network telah diterapkan pada classification, approximation, prediction, recognition, memory simulation, clustering, dll.

2.1.2 Struktur Neuron Pada Otak Manusia

Ide dasar dari pembuatan Neural Network dimulai dari cara kerja otak manusia dalam belajar dan struktur otak manusia yang terdiri dari neuron yang saling terhubung, mengirim, dan menerima informasi. Satu neuron memiliki satu akson dan minimal satu dendrit. Setiap sel saraf terhubung dengan sel saraf lain yang saling berinteraksi dan menghasilkan kemampuan tertentu pada kerja otak manusia sehingga otak menjadi semakin pintar atau memiliki banyak kemampuan serta pengetahuan. Misalnya kemampuan otak ketika kita masih kecil dengan kemampuan otak kita saat dewasa tidak akan sama, karena otak kita selalu belajar hingga kita dapat memahami hal-hal yang baru dan kemampuan otak kita semakin berkembang.

Perhatikan gambar 2.1, pada gambar neuron terdiri dari:

1. Dendrit (Dendrites) berfungsi untuk mengirimkan impuls yang diterima berupa arus listrik ke badan sel saraf.
2. Akson (Axon) berfungsi untuk mengirimkan impuls dari badan sel ke jaringan lain.
3. Sinapsis berfungsi sebagai unit fungsional (penghubung) di antara dua sel saraf.
4. Selubung mielin sebagai pelindung akson dan pemberi nutrisi.
5. Nodus Ranvier berfungsi untuk mempercepat impuls saraf.
6. Nukleus merupakan inti sel yang bertugas sebagai pengatur kegiatan sel saraf (neuron).
7. Soma berfungsi untuk mengendalikan metabolisme keseluruhan dari neuron.
8. Sel Schwann adalah penunjang sel saraf berupa lemak yang berfungsi menghasilkan mielin atau selubung saraf.

Berikut cara kerja otak manusia:

Ketika sebuah neuron menerima impuls dari neuron lain melalui dendrit, maka dendrit akan mengirimkan sinyal tersebut ke badan sel saraf. Selanjutnya Akson akan menerima sinyal dari badan sel dan mengirimkannya ke sel saraf lain. Akson dari sel saraf ini bercabang-cabang, akson sel saraf satu berhubungan dengan akson sel saraf dua melalui sinapsis. Sinapsis adalah unit fungsional antara 2 buah sel saraf, misal sel A dan sel B, akson sel A akan berhubungan dengan dendrit sel B melalui sinapsis. Kekuatan sinapsis bisa menurun/meningkat tergantung tingkat propagasi sinyal yang diterimanya. Impuls-impuls sinyal (informasi) akan diterima oleh neuron lain jika memenuhi batasan tertentu, dikenal dengan nilai ambang (threshold).

2.1.3 Struktur Neural Network

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

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

Input, berfungsi seperti dendrite Output, berfungsi seperti akson Fungsi aktivasi, berfungsi seperti sinapsis Neural network dibangun dari banyak node/unit yang dihubungkan oleh link secara langsung. Link dari unit yang satu ke unit yang lainnya digunakan untuk melakukan propagasi aktivasi dari unit pertama ke unit selanjutnya. Setiap link memiliki bobot numerik. Bobot ini menentukan kekuatan serta penanda dari sebuah koneksi.

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

ANN terdiri dari banyak neuron di dalamnya. Neuron-neuron ini akan dikelompokkan ke dalam beberapa layer. Neuron yang terdapat pada tiap layer dihubungkan dengan neuron pada layer lainnya. Hal ini tentunya tidak berlaku pada layer input dan output, tapi hanya layer yang berada di antaranya. Informasi yang diterima di layer input dilanjutkan ke layer-layer dalam ANN secara satu persatu hingga mencapai layer terakhir/layer output. Layer yang terletak di antara input dan output disebut sebagai hidden layer. Namun, tidak semua ANN memiliki hidden layer, ada juga yang hanya terdapat layer input dan output saja.

2.1.4 Cara Kerja Neural Network

Cara kerja Neural Network sama halnya dengan proses belajar yang dilakukan oleh otak manusia dengan menggunakan contoh atau disebut juga supervised learning. Neural network dikonfigurasikan pada aplikasi tertentu seperti pengklasifikasian data atau pengenalan pola. Neural Network memproses informasi seperti cara kerja otak manusia yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan dan berkerja secara paralel dalam menyelesaikan masalah. Neural Network da-

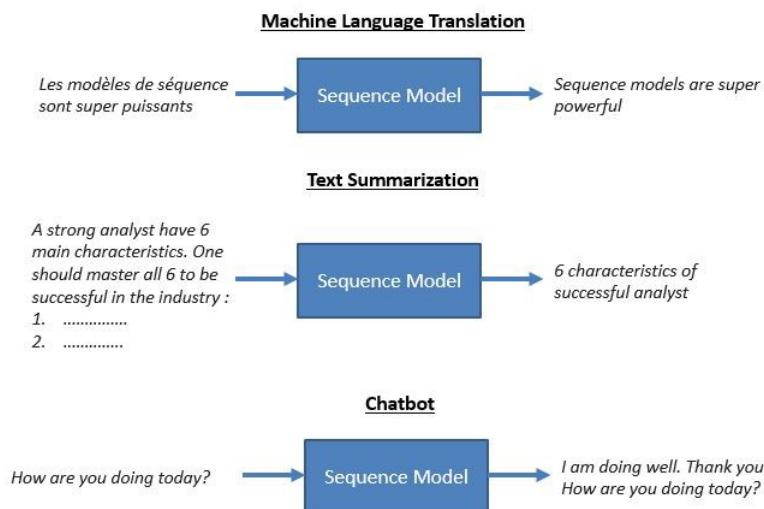
pat digunakan untuk memperoleh informasi atau pengetahuan dari data-data yang rumit, mengekstrak pola, mendeteksi tren yang memiliki kompleksitas yang rumit untuk dipelajari manusia ataupun teknik komputasi lainnya. Oleh karena itu, neural network yang telah dilatih hingga dapat menganalisis dan memproses informasi dari data dapat dikategorikan sebagai ahli. Neural Network sangat cocok untuk menyelesaikan beberapa masalah terkait prediksi yang membutuhkan pemahaman dan analisis yang baik contohnya prediksi pergerakan data time-series. Sedangkan algoritma komputer konvensional lebih cocok untuk menyelesaikan masalah terkait operasi aritmatika. Namun, pada beberapa masalah Neural Network dan Algoritma Komputer Konvensional dikombinasikan agar dapat memberikan kinerja maksimum.

BAB 3

SEQUENCE TO SEQUENCE (SEQ2SEQ)

3.1 Model Sequence to Sequence

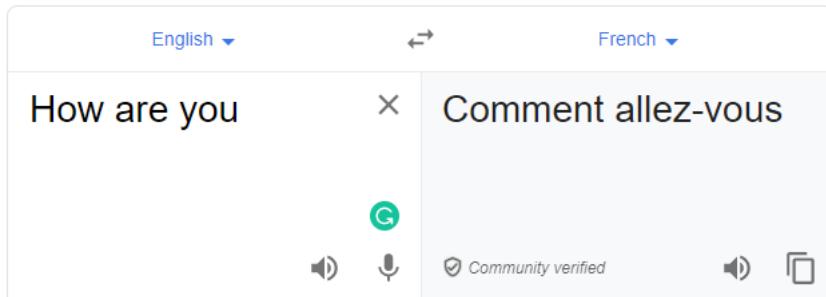
Model Sequence to Sequence (seq2seq) merupakan arsitektur Jaringan Saraf Berulang yang biasa digunakan untuk memecahkan masalah bahasa yang kompleks seperti Terjemahan, Menjawab Pertanyaan, membuat Chatbot, Peringkasan Teks, dll. Contoh penggunaan model seq2seq dapat dilihat pada gambar 3.1



Gambar 3.1 Contoh Penggunaan Model Sequence to Sequence

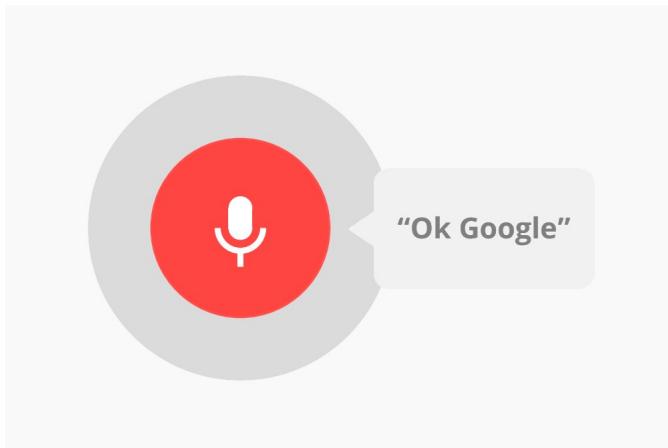
Model seq2seq sering dijumpai pada sistem yang sering kita gunakan sehari-hari. Model seq2seq mendukung aplikasi seperti Google Terjemahan, perangkat yang mendukung suara, voice assistant, chatbot, dll . Berikut ini adalah beberapa aplikasinya:

1. Google Translate, makalah tahun 2016 dari Google menunjukkan bagaimana kualitas terjemahan model seq2seq yang mendekati atau bahkan melampaui semua hasil yang dipublikasikan saat ini.



Gambar 3.2 Google Translate

2. Speech Recognition, makalah Google yang membandingkan model seq2seq yang ada pada pengenalan suara.



Gambar 3.3 Speech Recognition

3. Video Captioning – Secara otomatis membuat subtitle video untuk setiap frame, termasuk deskripsi isyarat suara (seperti mesin yang dinyalakan, orang yang tertawa di latar belakang, dll.).



Gambar 3.4 Video Captioning

Beberapa aplikasi tersebut membuat model seq2seq dipandang sebagai solusi terbaik. Model ini dapat digunakan sebagai solusi untuk setiap masalah berbasis urutan, terutama yang input dan outputnya memiliki ukuran dan kategori yang berbeda.

3.1.1 Arsitektur Encoder-Decoder

Arsitektur yang paling umum digunakan untuk membangun model Seq2Seq adalah arsitektur Encoder-Decoder.

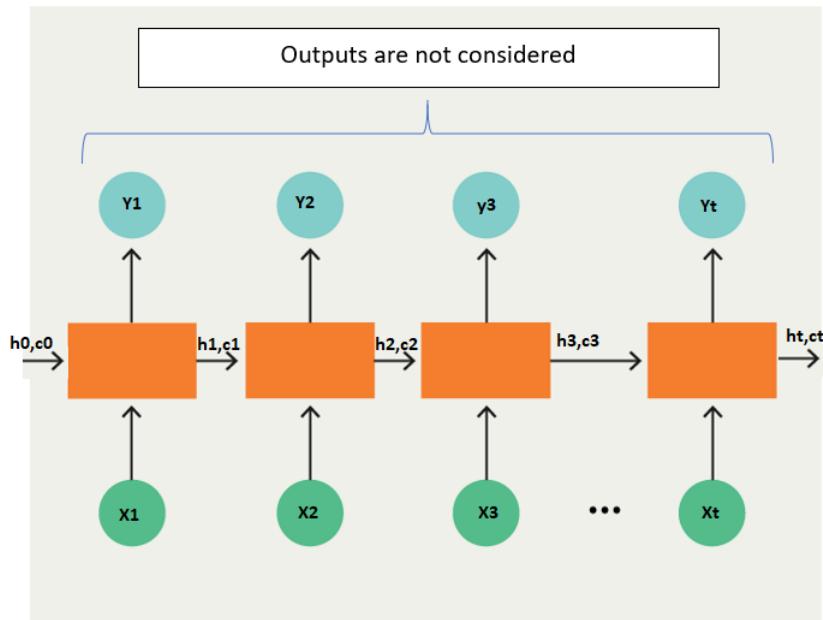
Encoder:

1. Encoder dan decoder adalah model LSTM (atau terkadang model GRU)
2. Encoder membaca urutan input dan merangkum informasi dalam sesuatu yang disebut internal state vectors atau context vector (dalam kasus LSTM ini disebut hidden state dan cell state vectors). Kami membuang output encoder dan hanya mempertahankan status internal. Vektor konteks ini bertujuan untuk merangkum informasi untuk semua elemen input untuk membantu dekoder membuat prediksi yang akurat.
3. Hidden State h_i dihitung menggunakan rumus pada gambar 3.5:

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

Gambar 3.5 Rumus Hidden State

Perhatikan gambar 8.2 LSTM membaca data dari satu urutan secara berurutan. Jadi jika inputnya adalah barisan dengan panjang ' t ', kita katakan bahwa LSTM membacanya dalam langkah waktu ' t '.



Gambar 3.6 Encoder

1. X_i = Urutan input pada langkah waktu i.
2. h_i dan c_i = LSTM mempertahankan dua status ('h' untuk status tersembunyi dan 'c' untuk status sel) pada setiap langkah waktu. h dan c yang dikombinasikan bersama-sama merupakan keadaan internal LSTM pada langkah waktu i.
3. Y_i = Urutan keluaran pada langkah waktu i. Y_i sebenarnya adalah distribusi probabilitas atas seluruh kosakata yang dihasilkan dengan menggunakan aktivasi softmax. Jadi setiap Y_i adalah vektor dengan ukuran "vocab_size" yang mewakili distribusi probabilitas.

Dekoder:

1. Decoder adalah LSTM yang status awalnya diinisialisasi ke final state Encoder LSTM, yaitu context vector dari final cell encoder dimasukkan ke first cell pada jaringan decoder. Dengan menggunakan initial states ini, dekoder mulai menghasilkan output sequence, dan output ini juga dipertimbangkan untuk output selanjutnya.
2. Tumpukan beberapa unit LSTM di mana masing-masing memprediksi output y_{-t} pada langkah waktu t.
3. Setiap unit berulang menerima hidden state dari unit sebelumnya menghasilkan dan mengeluarkan hidden statenya sendiri.
4. Setiap hidden state h_{-i} dihitung menggunakan rumus:

$$h_t = f(W^{(hh)} h_{t-1})$$

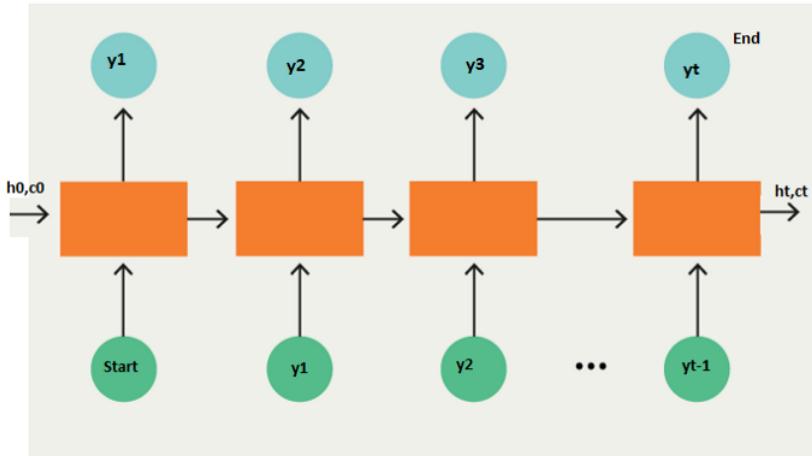
Gambar 3.7 Rumus Hidden State

5. Output y_{-t} pada langkah waktu t dihitung dengan menggunakan rumus:

$$y_t = \text{softmax}(W^S h_t)$$

Gambar 3.8 Rumus Hidden State

Menghitung output dilakukan dengan menggunakan hidden state pada langkah waktu saat ini bersama-sama dengan respective weight $W(S)$. Softmax digunakan untuk membuat vektor probabilitas yang akan membantu kita menentukan hasil akhir (misalnya kata dalam masalah tanya jawab).

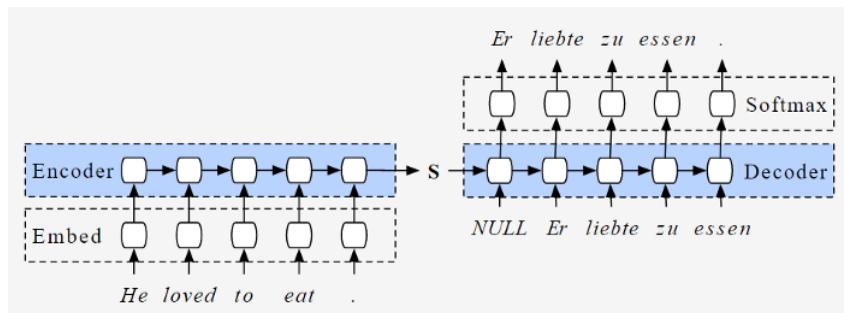


Gambar 3.9 Decoder

Sebagai contoh kita akan menambahkan dua token dalam urutan output sebagai berikut:

Contoh: “ START_John pekerja keras _END ”.

Poin yang paling penting adalah bahwa first state (h_0, c_0) dari dekoder diatur ke final state dari encoder. Ini secara intuitif berarti bahwa dekoder dilatih untuk mulai menghasilkan urutan output tergantung pada informasi yang dikodekan oleh encoder. Sehingga loss dihitung pada output yang diprediksi dari setiap langkah waktu dan error disebarluaskan kembali melalui waktu untuk memperbarui parameter jaringan. Training network dalam jangka waktu yang lebih lama dengan jumlah data yang cukup besar menghasilkan prediksi yang cukup bagus.

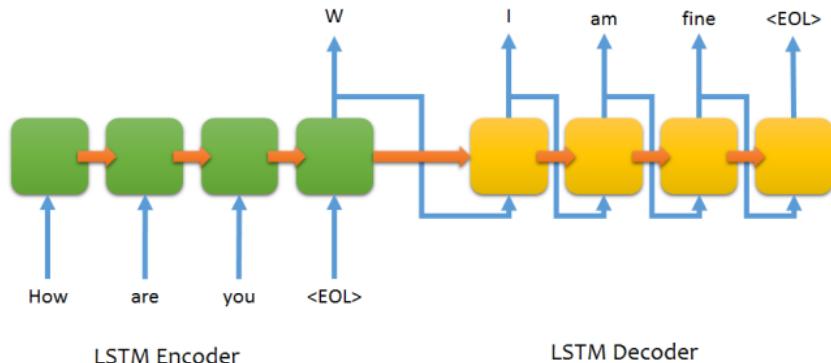


Gambar 3.10 Arsitektur Encoder-Decoder Secara Keseluruhan

1. Selama inferensi dihasilkan satu kata pada satu waktu.
2. Initial state dekoder diatur ke final state encoder.

3. initial input ke dekoder selalu berupa token START.
4. Pada setiap time step kita harus mempertahankan status dekoder dan menetapkannya sebagai status awal untuk time step berikutnya.
5. Pada setiap time step, output yang diprediksi diumpulkan sebagai input pada time step berikutnya.
6. loop akan dihentikan ketika decoder memprediksi token END.

Sebuah sequence untuk model urutan memiliki dua bagian - encoder dan decoder. Kedua bagian itu praktis adalah dua model jaringan saraf yang berbeda digabungkan menjadi satu jaringan raksasa. Secara umum, tugas jaringan encoder adalah memahami urutan input, dan membuat representasi dimensi yang lebih kecil darinya. Representasi ini kemudian diteruskan ke jaringan decoder yang menghasilkan urutannya sendiri yang mewakili output. Mari kita ambil contoh agen percakapan untuk memahami konsepnya.



Gambar 3.11 Chatbot dengan Seq2Seq Model

Pada gambar 3.11, urutan input adalah "Bagaimana kabarmu". Jadi ketika urutan input seperti itu dilewatkan melalui jaringan encoder-decoder yang terdiri dari blok LSTM (sejenis arsitektur RNN), decoder menghasilkan kata-kata satu per satu di setiap langkah waktu iterasi dekoder. Setelah satu iterasi penuh, urutan output yang dihasilkan adalah "Saya baik-baik saja".

3.1.2 Kekurangan Model Encoder-Decoder

Ada dua kelemahan utama arsitektur ini, keduanya terkait dengan panjang.

Pertama, seperti halnya manusia, arsitektur ini memiliki memori yang sangat terbatas. Keadaan tersembunyi terakhir dari LSTM, yang kami sebut S atau W , adalah saat Anda mencoba menjelaskan keseluruhan kalimat yang harus Anda terjemahkan.

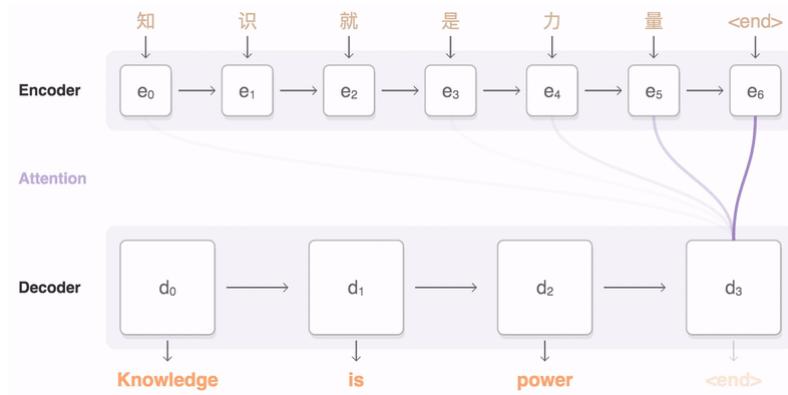
S atau W biasanya hanya beberapa ratus unit (baca: bilangan floating-point) semakin Anda mencoba untuk memaksa ke dalam vektor dimensi tetap ini, semakin besar kerugian jaringan saraf yang dipaksakan. Memikirkan jaringan saraf dalam hal "kompresi lossy" yang harus mereka lakukan terkadang cukup berguna. Kedua, sebagai aturan umum, semakin dalam jaringan saraf, semakin sulit untuk dilatih. Untuk jaringan saraf berulang, semakin panjang urutannya, semakin dalam jaringan saraf sepanjang dimensi waktu. Ini menghasilkan gradien yang hilang, di mana sinyal gradien dari tujuan yang dipelajari oleh jaringan saraf berulang menghilang saat bergerak mundur. Bahkan dengan RNN yang dibuat khusus untuk membantu mencegah gradien yang hilang, seperti LSTM, ini masih merupakan masalah men-dasar. Selanjutnya, untuk kalimat yang lebih kuat dan panjang, kami memiliki model seperti Attention Models dan Transformers .

3.1.3 Sequence to Sequence dengan Attention

Deep Learning dalam skala besar mengganggu banyak industri dengan membuat chatbot dan bot yang belum pernah ada sebelumnya. Di sisi lain, seseorang yang baru memulai Deep Learning akan membaca tentang Dasar-dasar Neural Networks dan berbagai arsitekturnya seperti CNN dan RNN. Tapi sepertinya ada lonjakan besar dari konsep sederhana ke aplikasi industri Deep Learning. Konsep-konsep seperti Batch Normalization, Dropout dan Attention hampir merupakan persyaratan untuk diketahui dalam membangun aplikasi deep learning. Berikut dua konsep penting yang digunakan dalam aplikasi terkini dalam Pengenalan Ucapan dan Pemrosesan Bahasa Alami – yaitu pemodelan Sequence to Sequence dan Attention Model. Sekadar memberikan gambaran tentang potensi penerapan kedua teknik ini Sistem AI Baidu menggunakan untuk membuat kloningan suara manusia. Ini mereplikasi suara seseorang dengan memahami suaranya hanya dalam tiga detik pelatihan. Kita dapat melihat beberapa sampel audio yang disediakan oleh Tim Riset Baidu yang terdiri dari suara asli dan suara yang disintesis. Ketika manusia mencoba memahami sebuah gambar, ia memfokuskan pada bagian-bagian tertentu dari gambar untuk mendapatkan keseluruhan esensi dari gambar tersebut. Dengan cara yang sama, kita dapat melatih sistem buatan untuk fokus pada elemen tertentu dari gambar untuk mendapatkan keseluruhan "gambar". Ini pada dasarnya adalah bagaimana mekanisme perhatian bekerja. Mari kita ambil contoh masalah teks gambar, di mana sistem harus menghasilkan teks yang sesuai untuk gambar. Dalam skenario ini, untuk menghasilkan keterangan, mekanisme perhatian membantu model untuk memahami bagian-bagian individu dari gambar yang paling penting pada contoh tertentu. Perhatikan gambar 3.12

**Gambar 3.12** Gambar to Text

Untuk menerapkan mekanisme attention, kami mengambil input dari setiap langkah waktu encoder tetapi memberi bobot pada langkah waktu. Pembobotan tergantung pada pentingnya langkah waktu tersebut bagi dekoder untuk menghasilkan kata berikutnya secara optimal dalam urutan, seperti yang ditunjukkan pada gambar 3.13

**Gambar 3.13** Mekanisme Attention pada Gambar to Text

3.1.4 Rumusan Masalah untuk Pemodelan Sequence to Sequence

Kita tahu bahwa untuk memecahkan masalah pemodelan sequence, Jaringan Syaraf Tiruan adalah arsitektur terbaik yang dapat kita pilih. Mari kita ambil contoh Sistem Penjawab Pertanyaan untuk memahami seperti apa masalah pemodelan urutan. Misalkan terdapat serangkaian pernyataan sebagai berikut:

Jo pergi ke dapur. Fred pergi ke dapur. Joe mengambil susu itu.

Joe pergi ke kantor. Joe meninggalkan susu. Jo pergi ke kamar mandi.

Pertanyaannya dimana joe sebelum pergi ke kantor? Jawaban yang tepat adalah "dapur". Pandangan sekilas membuat ini tampak seperti masalah sederhana. Tetapi untuk memahami kompleksitasnya terdapat dua dimensi yang harus dipahami oleh sistem:

1. Pemahaman dalam bahasa Inggris dan urutan karakter/kata yang membentuk kalimat.
2. Urutan peristiwa yang berkisar pada orang-orang yang disebutkan dalam pernyataan.

Ini dapat dianggap sebagai masalah pemodelan urutan, karena memahami urutan itu penting untuk membuat prediksi apa pun di sekitarnya. Ada banyak skenario masalah pemodelan urutan seperti itu, yang dirangkum dalam gambar di bawah ini. Contoh yang diberikan di atas adalah masalah banyak inputan dengan satu output (Jika Anda menganggap sebuah kata sebagai output tunggal).

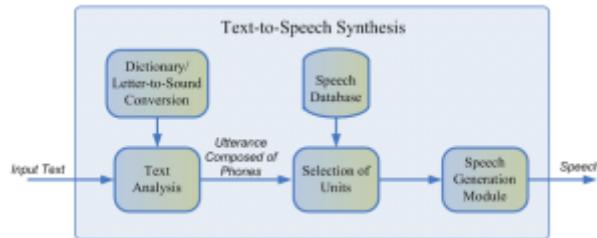
BAB 4

SPEECH SYNTHESIS

4.1 Speech Synthesis

Di berbagai media, teman-teman mungkin pernah menyaksikan Stephen Hawking berbicara di depan mahasiswanya. Fisikawan yang terkenal dengan teori black hole-nya ini sudah tidak mampu lagi mengeluarkan suara dari lisannya, namun berkat teknologi speech synthesizer, dia masih bisa bercakap-cakap. Mesin speech synthesizer Hawking memang cukup kompleks. Alat ini tidak hanya memproduksi suara, tetapi juga menangkap input dari gerakan mata sang doktor. Demikian pula, misalkan, dengan aplikasi voice command yang banyak tertanam di smartphone mutakhir yang memadukan speech recognizer dengan speech synthesizer.

Aplikasi speech synthesizer yang paling sederhana sebenarnya ada pada setiap PC yang menggunakan sistem operasi Windows. Bila anda menekan tuts Winkey + U di keyboard, Windows akan mengaktifkan Utility Manager, yang di dalamnya terdapat aplikasi Microsoft Narrator. Aplikasi ini akan membaca setiap jendela yang anda aktifkan, termasuk tombol-tombol di dalamnya atau mungkin apakah teman-teman pernah menginstal aplikasi microsoft reader di PC. Aplikasi yang diperuntukkan bagi file LTT ini pun dilengkapi dengan kemampuan menerjemahkan teks menjadi suara (text to speech) yang merupakan contoh teknologi speech synthesizer.



Gambar 4.1 Speech Synthesis

Speech synthesis adalah sebuah kemampuan bicara manusia yang dibuat oleh manusia (artificial). Sebuah sistem komputer digunakan untuk tujuan ini yang disebut sebagai speech synthesizer, dan dapat diimplementasikan ke dalam software atau hardware. Sebagai contoh sebuah sistem text-to-speech (TTS) yang dapat mengkonversikan teks dengan bahasa biasa menjadi suara[4].

Sebuah sistem komputer yang digunakan untuk tujuan ini disebut speech synthesizer, dan dapat diimplementasikan dalam perangkat lunak atau perangkat keras produk. Sebuah teks-to-speech (TTS) sistem mengkonversi teks bahasa normal menjadi berbicara; sistem lain membuat representasi linguistik simbolik seperti transkripsi fonetik ke dalam pidato, pidato disintesis dapat dibuat dengan menggabungkan potongan pidato direkam yang disimpan dalam database. Sistem berbeda dalam ukuran unit pidato disimpan, sebuah sistem yang menyimpan telepon atau diphones menyediakan berbagai keluaran terbesar, tapi mungkin kurang jelas[3].

Speech synthesis adalah transformasi dari teks ke arah suara (speech). Transformasi ini mengkonversi teks ke pemandu suara (speech synthesis) yang sebisa mungkin dibuat menyerupai suara nyata, disesuaikan dengan aturan – aturan pengucapan bahasa. TTS (text to speech) dimaksudkan untuk membaca teks elektronik dalam bentuk buku, dan juga untuk menyuarakan teks dengan menggunakan pemanduan suara. Sistem ini dapat digunakan sebagai sistem komunikasi, pada sistem informasi referral, dapat diterapkan untuk membantu orang-orang yang kehilangan kemampuan melihat dan membaca[5].

Speech synthesis dapat diciptakan dengan menggabungkan beberapa potongan-potongan dari pembicaraan/pidato yang sudah direkam dalam sebuah basis data. Kualitas dari sebuah speech synthesizer dilihat dari kemiripannya dengan suara manusia dan kemampuannya untuk bisa dipahami. Program TTS yang jelas dapat membantu orang dengan gangguan visual atau ketidakmampuan membaca. Banyak Sistem Operasi pada komputer yang telah dimasukkan speech synthesizer sejak tahun 1980-an.

Sebuah sistem text-to-speech (atau “mesin”) terdiri dari dua bagian: front-end dan back-end. Front-end memiliki dua tugas utama. Pertama, mengubah teks mentah berisi simbol seperti angka dan singkatan menjadi setara dengan kata-kata tertulis. Proses ini sering disebut normalisasi teks, pra-pengolahan, atau tokenization. Front-end kemudian memberikan transkripsi fonetik untuk setiap kata, dan membagi dan menandai teks ke unit prosodi, seperti frase, klausa, dan kalimat. Proses menetapkan

transkripsi fonetik untuk kata-kata ini disebut text to phonem atau grafem konversi untuk fonem. Transkripsi fonetik dan informasi prosodi bersama-sama dengan front-end membentuk output representasi linguistik simbolik. Back-end sering disebut sebagai synthesizer untuk mengubah representasi linguistik simbolik menjadi suara. Dalam sistem tertentu, bagian ini meliputi perhitungan dari target prosodi (kontur pitch, durasi fonem), yang kemudian dikenakan pada *speech output*.

4.1.1 Sejarah Speech Synthesis

Upaya yang paling awal untuk menghasilkan lahirnya speech synthesis yaitu pada abad ke-XVIII. Terlepas dari kenyataan bahwa upaya pertama adalah pembuatan mesin mekanis, kita dapat mengatakan hari ini bahwa synthesizer sudah memiliki kualitas yang tinggi. Pada tahun 1779 di St Petersburg, Rusia Profesor Kratzenshtein Kristen fisiologis menjelaskan perbedaan antara lima vokal panjang (/ A /, / e /, / i /, / o /, dan / u /) dan membuat alat untuk menghasilkan mereka artifisial. Tahun 1791 di Wina, Wolfgang von Kempelen memperkenalkan nya “Akustik-Mekanik Speech Machine”. Dalam sekitar pertengahan 1800-an Charles Wheatstone membangun versi mesin berbicara von Kempelen’s.

Generasi dari sistem speech synthesis ini dapat dibagi ke dalam 3 masa, yaitu:

1. Generasi pertama (1962-1977). Format sintesis dari fonem adalah teknologi dominan. Teknologi ini memanfaatkan aturan berdasarkan penguraian fonetik pada kalimat untuk kontur frekuensi forman. Beberapa sintesis masih minim atau kurang dalam kejelasan dan kealamiannya.
2. Generasi kedua (1977-1992). Metode speech synthesis adalah diphone diwakilkan dengan parameter LPC. Hal tersebut menunjukkan bahwa kejelasan yang baik pada pemandu suara dapat diperoleh dengan andal dari input teks dengan menggabungkan diphone yang sesuai dengan unit. Kejelasan meningkat selama sintesis forman, tetapi kealaman dari pemandu suara masih tetap rendah.
3. Generasi ketiga (1992-sekarang). Generasi ini ditandai dengan metode ‘unit selection synthesis’ yang diperkenalkan dan disempurnakan oleh Sagisaka di Labs ATR Kyoto. Hasil dari speech synthesis pada periode ini sangat mendekati human-generated speech pada bagian kejelasan dan kealaman.

Teknologi pemandu suara modern melibatkan metode dan algoritma yang canggih dan rumit. alat pemandu suara dari keluarga “Infovox” mungkin menjadi salah satu multi bahasa TTS yang paling dikenal saat ini. Versi komersial pertamanya, Infovox-SA 101, dikembangkan pada tahun 1982 di Institute Teknologi Royal, Swedia dan didasarkan pada sintesis forman.

AT & T Bell Laboratories (Lucent Technologies) juga memiliki tradisi yang sangat panjang tentang pemandu suara (speech synthesis). TTS lengkap yang pertama didemostrasikan di Boston pada tahun 1972 dan diliris pada tahun 1973. Hal ini didasarkan pada model artikulatoris yang sikembangkan oleh Ceceil Coker (Klatt 1987). Pengembangan proses dari sistem penggabungan sintesis ini dimulai oleh

Joseph Olive pada pertengahan tahun 1970-an (Bell Labs 1997). Sistem ini sekarang sudah tersedia untuk bahasa Inggris, Perancis, Spanyol, Italia, Jerman, Rusia, Rumania, Cina, dan Jepang (McBius et al 1996).



Gambar 4.2 Stephen Hawking adalah salah satu orang paling terkenal yang menggunakan sintesis ucapan untuk berkomunikasi

4.1.2 Perangkat Speech Synthesis

Pidato sistem sintesis berbasis komputer pertama diciptakan pada akhir 1950-an. Pertama umum Inggris sistem text-to-speech dikembangkan oleh Noriko Umeda et al. Pada tahun 1968 di Laboratorium Elektroteknik, Jepang. Pada tahun 1961, fisikawan John Larry Kelly, Jr dan Louis rekan Gerstman menggunakan IBM 704 komputer untuk mensintesis pidato, acara yang paling menonjol dalam sejarah Bell Labs. Kelly perekam suara synthesizer (vocoder) ulang lagu "Daisy Bell", dengan irungan musik dari Max Mathews .Kebetulan, Arthur C. Clarke mengunjungi teman dan kolega John Pierce di fasilitas Bell Labs Murray Hill. Clarke begitu terkesan oleh demonstrasi bahwa ia digunakan dalam adegan klimaks dari skenario-Nya untuk

novel nya 2001: A Space Odyssey, di mana HAL 9000 komputer menyanyikan lagu yang sama seperti yang sedang ditidurkan oleh astronot Dave Bowman. Meskipun keberhasilan pidato sintesis murni elektronik, penelitian masih terus dilakukan ke synthesizer pidato mekanis.

Handheld elektronik menampilkan sintesis pidato mulai muncul pada 1970-an. Salah satu yang pertama adalah Telesensory Systems Inc(TSI) Pidato + kalkulator portabel untuk orang buta pada tahun 1976. Perangkat lain yang diproduksi terutama untuk tujuan pendidikan, seperti Bicara & Eja , yang diproduksi oleh Texas Instruments pada tahun 1978. Fidelity merilis versi berbicara komputer catur elektronik pada tahun 1979. Yang pertama video game yang memiliki fitur sintesis pidato adalah 1.980 shoot 'em up arcade game , Stratovox , dari Sun Electronics. Contoh lain awal adalah versi arcade dari Berzerk , dirilis pada tahun yang sama.Pertama multi-player permainan elektronik menggunakan sintesis suara adalah Milton dari Milton Bradley Company , yang memproduksi perangkat di tahun 1980.

4.1.3 Teknologi Speech Synthesis

Yang paling penting dalam kualitas sistem speech synthesis adalah kealamian dan kejelasannya. Kealamia menjelaskan bagaimana dekatnya suara output dengan suara manusia, sementara kejelasan adalah dengan kemudahan di mana output tersebut dapat dipahami. Speech synthesizer yang ideal adalah yang alami dan jelas. Sistem speech synthesis biasanya mencoba untuk memaksimalkan kedua karakteristik.

Kualitas terpenting dari sebuah aplikasi speech synthesizer adalah seberapa alami dan inteligibel output yang dihasilkannya. Alami, artinya seberapa dekat suara yang dihasilkan aplikasi speech synthesizer dengan suara manusia. Sedangkan inteligibel adalah seberapa mudah output tersebut dipahami oleh manusia. Semua aplikasi speech synthesizer berusaha untuk menghasilkan output yang alami dan inteligibel sekaligus.

Sampai saat ini, ada banyak teknologi untuk meng-generate gelombang suara sintetis ini. Dua teknologi yang paling banyak digunakan adalah concatenative synthesis[6] dan formant synthesis. Keduanya memiliki keunggulan dan kekurangan sendiri-sendiri.

Teknologi pertama, concatenative synthesis[6], berbasis pada rangkaian (merangkai bersama) segmen-semen dari suara yang direkam. Umumnya, teknologi ini menghasilkan suara sintesis yang terdengar paling alami.Namun, perbedaan antara suara alami yang direkam dengan segmentasi gelombang bunyi kadang menghasilkan suara yang mengganggu. Mirip seperti suara pemberitahuan nomor antrean di bank atau suara call center operator ponsel yang menyebutkan sisa pulsa dan masa berlaku kartu ponsel anda.

Teknologi kedua, formant synthesis, tidak menggunakan sampel suara manusia melainkan membuat suara sintesi menggunakan model akustik. Parameter-parameter seperti frekuensi dasar, alunan suara, dan tingkat kebisingan bervariasi dari waktu ke waktu untuk menciptakan gelombang suara buatan.

Kebanyakan aplikasi berbasis teknologi ini menghasilkan suara buatan (tidak alami) seperti suara robot. Melihat keterbatasan kedua teknologi ini dalam menghasilkan

suara buatan, seperti kita harus sabar menunggu pengembangannya lebih lanjut dalam beberapa tahun atau dekade ke depan.

Kualitas yang paling penting dari sebuah sistem sintesis pidato kewajaran dan dimengerti. Kealamian menjelaskan seberapa dekat output terdengar seperti suara manusia, sementara kejelasan adalah kemudahan yang output dipahami. Speech synthesizer yang ideal adalah baik alam dan dimengerti. Sistem sintesis pidato biasanya mencoba untuk memaksimalkan kedua karakteristik.

Dua teknologi utama dalam pembuatan gelombang suara synthetic speech adalah Concatenative Synthesis dan Formant Synthesis. Setiap teknologi mempunyai kekuatan dan kelemahannya, dan penggunaan yang ditujukan dari sistem synthesis akan menentukkan pendekatan mana yang digunakan.

1. Concatenative Synthesis Concatenative synthesis didasarkan dengan penggabungan dari segmen-semen dari pembicaraan yang sudah direkam. Secara umum, concatenative synthesis memproduksi synthesized speech dengan suara yang paling alami. Tetapi, perbedaan antara variasi alami dalam pembicaraan dan sifat dari teknik otomasi untuk pensemantasi gelombang suara terkadang menghasilkan kesalahan suara dalam output. Namun, perbedaan antara variasi alami dalam pidato dan sifat teknik otomatis untuk membagi bentuk gelombang kadang-kadang menyebabkan gangguan terdengar pada output. Ada tiga sub-jenis utama dari sintesis concatenative[6].
 - (a) Sintesis Pemilihan unit Sintesis Pemilihan unit menggunakan besar database pidato direkam. Selama pembuatan database, setiap ucapan tercatat tersegmentasi ke dalam beberapa atau semua hal berikut: individu telepon , di-phones , setengah-telepon, suku kata , morfem , kata , frase , dan kalimat . Biasanya, pembagian ke dalam segmen dilakukan dengan menggunakan di-modifikasi khusus recognizer pidato disetel ke “keselarasan dipaksa” mode dengan beberapa koreksi manual setelah itu, dengan menggunakan representasi visual seperti yang gelombang dan spektrogram . Sebuah indeks unit dalam database pidato kemudian dibuat berdasarkan segmentasi dan parameter akustik seperti frekuensi dasar (lapangan), durasi, posisi dalam suku kata, dan telepon tetangga. Pada waktu berjalan , target ucapan yang dinginkan dibuat dengan menentukan rantai terbaik unit calon dari database (pemilihan unit). Proses ini biasanya dicapai dengan menggunakan khusus tertimbang pohon keputusan.

segmen dari Tempat yang menghasilkan kurang dari sintesis ideal (misalnya kata-kata kecil menjadi tidak jelas) bahkan ketika pilihan yang lebih baik ada dalam database. Baru-baru ini, peneliti telah mengusulkan berbagai metode otomatis untuk mendeteksi segmen alami di unit-pilihan sistem sintesis pidato.

- (b) Sintesis diphone Sintesis diphone menggunakan database pidato minimal berisi semua diphones (suara-to-suara transisi) yang terjadi dalam suatu bahasa. Jumlah diphones tergantung pada fonotaktik bahasa: misalnya, Spanyol memiliki sekitar 800 diphones, dan Jerman sekitar 2500. Dalam sintesis diphone, hanya satu contoh dari setiap diphone terkandung dalam database pidato. Pada saat runtime, target prosodi kalimat ditumpangkan pada unit-unit minimal dengan cara pemrosesan sinyal digital teknik seperti linear predictive coding ,PSOLA atau MBROLA. Diphone sintesis menderita gangguan sonik sintesis concatenative dan robot-terdengar sifat sintesis forman, dan memiliki beberapa keuntungan baik pendekatan lain dari ukuran kecil. Dengan demikian, penggunaannya dalam aplikasi komersial menu run, meskipun terus digunakan dalam penelitian karena ada beberapa implementasi perangkat lunak tersedia secara bebas.
- (c) Domain-spesifik sintesis Domain-spesifik sintesis concatenates direkam sebelumnya kata dan frase untuk menciptakan ucapan-ucapan yang lengkap. Hal ini digunakan dalam aplikasi di mana berbagai teks output sistem akan terbatas pada domain tertentu, seperti jadwal angkutan pengumuman atau laporan cuaca. Teknologi ini sangat sederhana untuk menerapkan, dan telah digunakan secara komersial untuk waktu yang lama , dalam perangkat seperti berbicara jam dan kalkulator. Tingkat kealamian sistem ini bisa sangat tinggi karena berbagai jenis kalimat terbatas, dan mereka cocok dengan prosodi dan intonasi dari rekaman asli.
- Karena sistem ini dibatasi oleh kata-kata dan frasa dalam database mereka, mereka tidak tujuan umum dan hanya dapat mensintesis kombinasi kata dan frase yang mereka telah terprogram. Campuran kata-kata dalam bahasa alami diucapkan namun masih dapat menyebabkan masalah kecuali banyak variasi diperhitungkan. Misalnya, dalam non-rhotic dialek dari bahasa Inggris “r” dalam kata-kata seperti “jelas” biasanya hanya diucapkan ketika kata berikut memiliki vokal sebagai huruf pertama (misalnya “membersihkan” direalisasikan sebagai). Demikian juga di Perancis , banyak konsonan akhir menjadi tidak lagi diam jika diikuti oleh sebuah kata yang dimulai dengan vokal, efek yang disebut penghubung . Ini pergantian tidak bisa direproduksi oleh sistem kata-Rangkaian sederhana, yang akan membutuhkan kompleksitas tambahan untuk konteks-sensitif .
2. Formant Synthesis Formant synthesis tidak menggunakan pembicaraan manusia sebagai sample pada runtime. Daripada itu, synthesized speech yang dihasilkan dibuat dengan additive synthesis dan sebuah model akustik (physical modelling synthesis).

Parameter seperti frekuensi dasar, penyuaraan, dan tingkat kebisingan di variasikan dari waktu ke waktu untuk menciptakan gelombang buatan (artificial) dari sebuah pembicaraan. Banyak sistem yang berdasarkan formant synthesis menciptakan pembicaraan yang seperti robot yang tidak mungkin dapat dikenal sebagai suara manusia. Tetapi, kealamian maksimum bukan selalu tujuan dari sebuah sistem speech synthesis, dan sistem formant synthesis mempunyai keuntungan dari sistem concatenative. Pembicaraan yang di-formant synthesis-kan dapat menjadi sangat jelas, bahkan dalam kecepatan yang tinggi, sehingga menghindari kesalahan suara yang sering dialami sistem concatenative.

Formant synthesis biasanya program yang lebih kecil dari concatenative sistem karena ia tidak menggunakan basis data dari sampel-sampel pembicaraan. Oleh karena itu formant synthesis dapat ditanamkan dalam sistem yang mempunyai memory dan mikroprosesor yang terbatas. Karena sistem yang berdasarkan formant mempunyai kendali penuh dari seluruh aspek dari hasil pembicaraan, variasi yang luas dari prosodi dan intonasi dapat dihasilkan, menyampaikan tidak hanya pertanyaan dan pernyataan tetapi juga emosi dan nada suara.

Formant sintesis tidak menggunakan sampel suara manusia pada saat runtime. Sebaliknya, keluaran suara yang disintesis dibuat menggunakan aditif sintesis dan model akustik (sintesis pemodelan fisik). Parameter seperti frekuensi dasar , menyuarakan , dan kebisingan tingkat yang bervariasi dari waktu ke waktu untuk membuat gelombang pidato buatan. Metode ini kadang-kadang disebut aturan berbasis sintesis; Namun, banyak sistem concatenative juga memiliki aturan berbasis komponen. Banyak sistem yang didasarkan pada teknologi sintesis forman menghasilkan buatan, robot yang terdengar pidato yang tidak akan pernah salah untuk pidato manusia. Namun, kealamian maksimum tidak selalu tujuan sistem sintesis pidato, dan sistem sintesis forman memiliki keunggulan dibandingkan sistem concatenative. Pidato forman-disintesis dapat diandalkan dimengerti, bahkan pada kecepatan yang sangat tinggi, menghindari Glitches akustik yang biasanya wabah sistem concatenative. Kecepatan tinggi disintesis pidato digunakan oleh tunanetra untuk navigasi cepat komputer menggunakan pembaca layar . Synthesizer forman adalah program biasanya lebih kecil dibandingkan dengan sistem concatenative karena mereka tidak memiliki database contoh pidato. Karena itu mereka dapat digunakan dalam embedded system , di mana memori dan mikroprosesor daya terutama terbatas. Karena sistem berbasis forman memiliki kontrol penuh dari semua aspek pidato output, berbagai prosodies dan intonasi dapat menjadi output, tidak hanya menyampaikan pertanyaan dan pernyataan, tetapi berbagai emosi dan nada suara.

Contoh non-real-time tapi sangat akurat kontrol intonasi dalam sintesis forman meliputi pekerjaan yang dilakukan pada akhir tahun 1970 untuk Texas Instruments mainan Bicara & Eja , dan pada awal tahun 1980 Sega arcade mesin dan dalam banyak Atari, Inc. game arcade menggunakan TMS5220 LPC Chips . Menciptakan intonasi yang tepat untuk proyek ini adalah telaten, dan hasilnya masih harus dicocokkan dengan real-time text-to-speech interface.

3. Sintesis artikulatoris Sintesis artikulatoris mengacu pada teknik komputasi untuk sintesis pidato berdasarkan model manusia saluran vokal dan artikulasi proses yang terjadi di sana. Synthesizer artikulatoris pertama teratur digunakan untuk percobaan laboratorium dikembangkan di Haskins Laboratories di pertengahan 1970-an oleh Philip Rubin, Tom Baer, dan Paul Mermelstein. Synthesizer ini, yang dikenal sebagai ASY, didasarkan pada model saluran vokal dikembangkan di Bell Laboratories pada tahun 1960 dan 1970-an oleh Paul Mermelstein, Cecil Coker, dan rekan.

Sampai saat ini, model sintesis artikulatoris belum dimasukkan ke dalam sistem sintesis pidato komersial. Sebuah pengecualian adalah NeXT sistem berbasis awalnya dikembangkan dan dipasarkan oleh Trillium Suara Research, sebuah perusahaan spin-off dari University of Calgary , di mana banyak riset asli dilakukan. Setelah runtuhan berbagai inkarnasi NeXT (dimulai oleh Steve Jobs pada akhir tahun 1980 dan bergabung dengan Apple Computer pada tahun 1997), perangkat lunak TRILLIUM diterbitkan di bawah GNU General Public License , dengan bekerja terus sebagai gnuSpeech . Sistem, pertama kali dipasarkan pada tahun 1994, memberikan penuh text-to-speech konversi berbasis artikulatoris menggunakan Waveguide atau transmisi-line analog dari saluran mulut dan hidung manusia dikendalikan oleh Carré ini “model daerah khas”.

4. Sintesis berbasis HMM Sintesis berbasis HMM adalah metode sintesis berdasarkan model Markov tersembunyi , juga disebut statistik Parametrik Sintesis[7]. Dalam sistem ini, spektrum frekuensi (vokal),frekuensi dasar (sumber vokal), dan durasi (prosodi) berbicara dimodelkan secara bersamaan oleh HMMs. Pidato bentuk gelombang yang dihasilkan dari HMMs sendiri berdasarkan maksimum kriteria[8].
5. Sintesis Sinewave Sintesis sinewave adalah teknik untuk sintesis pidato dengan mengganti forman (band utama energi) dengan peluit nada murni.

Ada beberapa masalah yang terdapat pada pemanduan suara, yaitu:

- (a) User sangat sensitif terhadap variasi dan informasi suara. Oleh sebab itu, mereka tidak dapat memberikan toleransi atas ketidak sempurnaan pemandu suara.
- (b) Output dalam bentuk suara tidak dapat diulang atau dicari dengan mudah.
- (c) Meningkatkan keberisikan pada lingkungan kantor atau jika menggunakan handphone, maka akan meningkatkan biaya pengeluaran.

BAB 5

VOICE CLONING

5.1 Voice Cloning

Kloning suara sering diimbangi dengan istilah lain, seperti suara deepfake, sintesis ucapan, dan suara sintetis, yang memiliki arti yang sedikit berbeda. Kloning suara adalah proses di mana seseorang menggunakan komputer untuk menghasilkan ucapan individu nyata, menciptakan tiruan dari suara mereka yang spesifik dan unik menggunakan kecerdasan buatan (AI)[9]. Sistem text-to-speech (TTS), yang dapat mengambil bahasa tertulis dan mengubahnya menjadi komunikasi lisan, tidak sama dengan kloning suara. Sistem TTS jauh lebih terbatas dari output yang mereka hasilkan dibandingkan dengan teknologi kloning suara, yang sebenarnya lebih merupakan proses kustom. Dengan sistem TTS, data pelatihan, komponen kunci untuk setiap media yang dibuat secara sintetis, menginformasikan produksi keluaran suara. Dengan kata lain, suara yang Anda dengar adalah suara yang diberikan dalam kumpulan data. Sekarang, dengan diperkenalkannya teknologi AI kloning suara, itu berubah. Metode telah diterapkan untuk memberikan analisis yang lebih dalam dan ekstraksi karakteristik suara target. Atribut-atribut ini kemudian dapat diterapkan pada bentuk gelombang ucapan yang berbeda, memungkinkan seseorang untuk mengubah keluaran suara dari satu suara ke suara lainnya[10].



Gambar 5.1 Voice Cloning

Berkat kemajuan dalam kecerdasan buatan (AI), khususnya pembelajaran mendalam, bagian dari pembelajaran mesin di bawah payung AI, kami telah mampu menghasilkan replika suara yang akurat. Tapi ini hanya dimungkinkan oleh dua hal:

1. Perangkat keras yang kuat dengan kemampuan komputasi awan untuk memproses dan merender secara tepat waktu dan efisien
2. Data pelatihan ekstensif dari suara yang ditargetkan dari mana model dapat memanfaatkan untuk membuat klon suara yang akurat

Dengan AI yang tepat dan keahlian dan alat pengembangan, itu benar-benar tergantung pada yang terakhir. Anda memerlukan sejumlah besar ucapan yang direkam untuk memiliki data yang cukup untuk melatih model suara. Informasi seputar suara disimpan dalam embedding , ruang berdimensi cukup rendah di mana Anda dapat menerjemahkan variabel diskrit menjadi vektor berdimensi tinggi. Dengan kata lain, lebih mudah untuk bekerja dengan input besar dengan model pembelajaran mesin. Agar tidak terlalu teknis, kami akan berhenti di situ, tetapi jangan ragu untuk menyelemah lebih dalam ke subjek jika itu menarik minat Anda.

5.1.1 Manfaat Voice Cloning

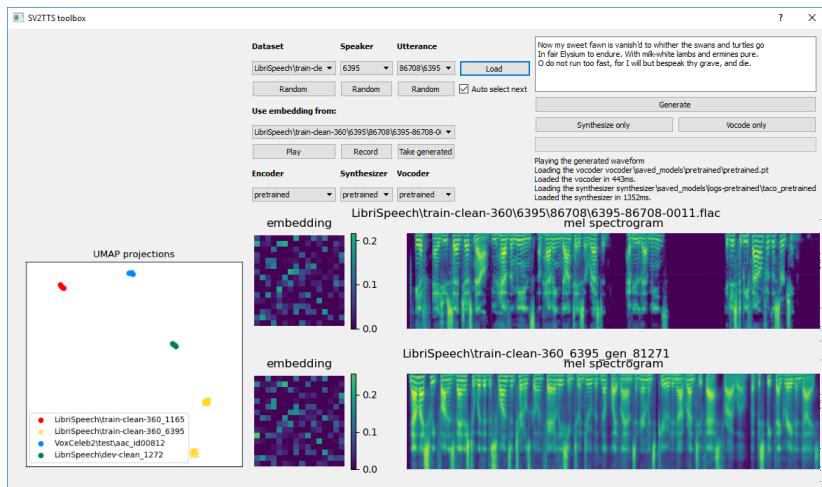
Mari kita mulai dengan yang baik. Ada banyak kasus penggunaan potensial untuk kloning suara yang sering kali dibayangi oleh penggunaan negatif, yang akan kita bahas sebentar lagi. Beberapa aplikasi positif dari teknologi meliputi:

1. Meningkatkan peluang iklan dan sponsor untuk pengisi suara, selebritas, dan influencer
2. Bantu perusahaan bekerja dengan talenta selama waktu tersibuk mereka dalam setahun, seperti musim sepak bola untuk pemain atau pelatih

3. Menghidupkan kembali suara-suara dari masa lalu untuk digunakan dalam hiburan guna membantu menceritakan kisah dalam dokumenter, film, dan acara TV
4. Diversifikasi konten siaran untuk konten berulang seperti laporan cuaca atau pembaruan olahraga
5. Lokalkan konten sehingga dapat didengar dalam suara pembawa acara atau narator dalam bahasa lain

Ini hanyalah beberapa kegunaan positif dari kloning suara, dan seiring dengan berkembangnya teknologi, lebih banyak lagi yang akan muncul. Tapi tentu saja, semuanya bergantung pada etika penggunaan suara seseorang. Itulah mengapa perlunya gerakan menuju standarisasi proses persetujuan sangat penting untuk melindungi suara semua orang dan memastikan mereka memiliki kendali penuh atas cara penggunaannya.

5.1.2 Aplikasi Voice Cloning Terbaik



Gambar 5.2 Contoh Aplikasi Voice Cloning (SV2TTS Toolbox)

Untuk mempersempit pencarian Anda untuk menemukan aplikasi kloning suara terbaik, Anda harus terlebih dahulu menentukan apa yang Anda cari. Apakah Anda memerlukan sesuatu yang lebih untuk output text-to-speech? Atau apakah Anda membutuhkan sesuatu yang lebih khusus?

Setelah Anda mengetahui mengapa Anda memerlukan aplikasi kloning suara, Anda harus mengasah tiga kriteria utama:

1. Kualitas keluaran: Anda pasti ingin memastikan bahwa keluaran terdengar otentik dan memenuhi kebutuhan yang ditentukan. Biasanya, mereka akan memi-

liki sampel tentang apa yang dapat dilakukan produk tersebut. Jika tidak, Anda harus mempertimbangkan untuk meminta demo, jika tersedia, untuk menentukan seberapa manusiawi produk mereka.

2. Antarmuka intuitif: seberapa mudah menggunakan aplikasi? Apakah sulit untuk menemukan sesuatu saat Anda berada di aplikasi atau dapatkah Anda menavigasi dan menggunakannya untuk memenuhi kebutuhan Anda? Sekali lagi, ini dapat ditentukan oleh video produk, konten pemasaran, dan demo.
3. Perlindungan suara: Anda pasti ingin memastikan bahwa perusahaan mengikuti penggunaan suara yang etis. Jika itu adalah layanan kustom yang memerlukan data pelatihan, maka penting untuk menanyakan tentang perlindungan data dan bagaimana suara, saat dibuat, tidak akan digunakan secara tidak semestinya.

Implikasi etis seputar kloning suara adalah hubungan dari Veritone MARVEL.ai, aplikasi suara sebagai layanan kami . Dibangun dalam kerangka aplikasi adalah tuas untuk memberi pengguna kendali atas suara mereka, memungkinkan perlindungan yang tepat sehingga mereka memutuskan siapa yang dapat menggunakan suara mereka. Ini membantu kami memberikan solusi voice-as-a-service khusus kami untuk memungkinkan pengalaman sarung tangan putih lengkap untuk talenta yang bekerja dengan kami.

5.1.3 Pembuatan Voice Cloning

Perangkat lunak kloning suara AI online dimulai dengan menggunakan komputer untuk mensintesis suara. Text-to-Speech (TTS) adalah teknologi berusia puluhan tahun yang mengubah teks menjadi ucapan sintetis, memungkinkan suara digunakan untuk interaksi komputer-manusia.

Di masa lalu ada dua pendekatan untuk TTS. Yang pertama, TTS Concatenative[6], menggunakan rekaman audio untuk membuat perpustakaan kata dan satuan suara (fonem) yang dapat dirangkai menjadi kalimat. Meskipun keluarannya berkualitas tinggi dan dapat dipahami, ia tidak memiliki emosi dan infleksi yang ditemukan dalam ucapan manusia yang alami. Saat menggunakan TTS Concatenative, setiap gaya bicara atau bahasa baru memerlukan database audio baru. Dan tentu saja upaya untuk mengkloning suara individu menggunakan metode ini membutuhkan investasi yang sangat besar, biasanya hanya dilakukan untuk mendukung suara bermerek.

Pendekatan kedua adalah Parametrik TTS[8], metode yang menggunakan model statistik ucapan untuk menyederhanakan pembuatan suara, mengurangi biaya dan upaya dibandingkan dengan Penggabungan. Namun, upaya untuk menciptakan satu suara apa pun secara historis mahal, dan hasilnya jelas bukan manusia.

Saat ini, Kecerdasan Buatan (AI) dan kemajuan dalam Pembelajaran Mendalam meningkatkan kualitas ucapan sintetis. Pengajuan TTS sekarang sudah lumrah. Setiap orang yang telah berinteraksi dengan sistem Respon Suara Interaktif berbasis telepon, Siri Apple, Amazon Alexa, sistem navigasi mobil, atau banyak antarmuka suara lainnya, telah mengalami ucapan sintetis.

Jika Anda terbiasa dengan konsep video deepfake, perangkat lunak kloning suara AI online adalah setara dengan ucapan. Hanya dengan beberapa menit rekaman ucapan, pengembang dapat membangun kumpulan data audio dan menggunakannya untuk melatih model suara AI yang dapat membaca teks apa pun dalam suara target.

Pembuatan voice cloning secara signifikan menjadi lebih mudah berkat berbagai alat bertenaga jaringan saraf seperti Tacotron dan Wavenet atau Lyrebird Google, yang memungkinkan hampir semua suara direplikasi dan digunakan untuk "membaca" input teks. Kualitas output terus meningkat, seperti yang ditunjukkan oleh tiruan suara podcaster Joe Rogan ini. Insinyur pembelajaran mendalam di Dessa yang membuat klon juga menyiapkan kuis. Ambil untuk melihat apakah Anda dapat melihat Rogan palsu.

Model TTS berbasis jaringan saraf meniru cara otak beroperasi dan sangat efisien dalam mempelajari pola dalam data. Meskipun ada pendekatan berbeda untuk penggunaan pembelajaran mendalam dalam suara sintetis, sebagian besar menghasilkan pengucapan kata yang lebih baik, serta menangkap kehalusan seperti kecepatan dan intonasi untuk menciptakan ucapan yang lebih mirip manusia.

Penting untuk dicatat bahwa alat yang disebutkan di atas dan alat lain seperti ini tidak dibuat untuk tujuan penipuan atau penipuan. Tetapi kenyataannya adalah bahwa bisnis dan konsumen perlu mewaspadai ancaman baru yang terkait dengan perangkat lunak kloning suara AI online. Kami menjelajahi beberapa kegunaan — baik dan buruk — di bawah ini.

5.1.4 Dampak Negatif Voice Cloning

1. Voice adalah pengenal pribadi unik yang mudah diakses oleh penipu. Hal ini tentu berlaku bagi publik figur termasuk selebriti, politisi dan pemimpin bisnis, tetapi kenyataannya adalah siapa saja bisa menjadi sasaran. Video online, pidato, panggilan konferensi, percakapan telepon, dan posting media sosial semuanya dapat digunakan untuk mengumpulkan data yang diperlukan untuk melatih sistem untuk mengkloning suara.
2. Spoofing biometrik suara — Suara adalah pengidentifikasi unik dan ukuran yang andal untuk keamanan biometrik. Namun, penjahat dapat menggunakan serangan presentasi termasuk suara yang direkam, suara yang diubah komputer dan suara sintetis, atau kloning suara, untuk mengelabui sistem biometrik suara agar mengira mereka mendengar pengguna yang sebenarnya dan berwenang dan memberikan akses ke informasi dan akun sensitif.



Gambar 5.3 Spoofing Voice Biometrics

3. Penipuan phishing – Perangkat lunak kloning suara AI online juga memungkinkan jenis baru penipuan phishing yang mengeksplorasi fakta bahwa korban percaya bahwa mereka sedang berbicara dengan seseorang yang mereka percayai. Tahun lalu, seorang CEO yang berbasis di Inggris ditipu untuk mentransfer lebih dari \$240.000 berdasarkan panggilan telepon yang dia yakini berasal dari bosnya, CEO perusahaan induk organisasi Jerman.



Gambar 5.4 Voice Phising, Sumber: <https://id.pinterest.com/pin/195765915039905230/>

4. Penipuan seperti ini adalah evolusi penipuan email di mana email eksekutif dipalsukan dalam upaya agar penerima membocorkan nomor rekening bank, informasi kartu kredit, kata sandi, dan data sensitif lainnya. Sekarang scammers, dipersenjatai dengan klon suara, menggunakan panggilan telepon dan pesan suara. Dan serangan itu tidak hanya mengancam bisnis. Dalam generasi

baru penjahat "mother's deception" menyamar sebagai anggota keluarga yang membutuhkan dana darurat.

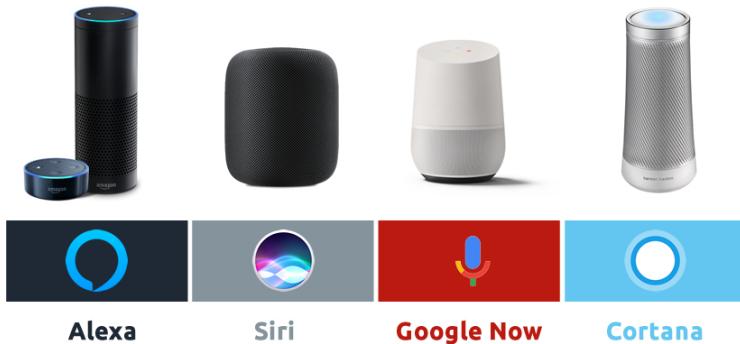
5. Misinformasi – Berita palsu dan bentuk misinformasi serupa merupakan ancaman serius. Banyak dari kita yang akrab dengan bagaimana video yang di-manipulasi berdampak pada lanskap politik. Misalnya, video populer menunjukkan Barack Obama memanggil Trump – yah, anggap saja itu tidak terlalu bagus. Itu juga tidak nyata. Teknologi perangkat lunak text-to-speech berbasis AI akan semakin mendorong upaya ini untuk mempengaruhi opini publik, menghidupkan sumbangsih kampanye palsu, mencemarkan nama baik tokoh masyarakat, dan banyak lagi. Di bidang bisnis, pertimbangkan bagaimana pernyataan eksekutif atau figur publik yang dimanipulasi dapat memengaruhi pasar saham.
6. Bukti – Suara sintetis dan deepfake lainnya dapat digunakan untuk membuat bukti palsu yang berdampak pada kasus kriminal. Meskipun ada pemeriksaan untuk memvalidasi bukti audio dan video yang disajikan di pengadilan, mencegah taktik ini memengaruhi kesaksian berdasarkan apa yang orang yakini mereka lihat atau dengar mungkin merupakan tantangan.
7. Pemerasan dan intimidasi – Video dan audio yang dimanipulasi dari orang-orang yang melakukan atau mengatakan hal-hal yang tidak mereka katakan dapat digunakan untuk intimidasi online dan ancaman untuk mengekspos konten palsu dan memalukan jika korban menolak untuk membayar biaya.

5.1.5 Dampak Positif Voice Cloning

1. Pendidikan, Mengkloning suara tokoh sejarah menawarkan peluang baru untuk pengajaran interaktif dan penceritaan yang dinamis. Misalnya, pada 22 November 1963 Presiden Kennedy sedang dalam perjalanan untuk memberikan pidato di Dallas ketika dia dibunuh. Kita sekarang dapat mendengar pidato itu dengan kata-katanya sendiri menggunakan teknologi deepfake. Dalam penggunaan deepfake AI lainnya yang menakjubkan, pengunjung Museum Dalí di St. Petersburg akan disambut oleh Salvador Dalí sendiri. Dalí berinteraksi dengan tamu menggunakan kutipan aktual dan membuat komentar, dan bahkan berfoto selfie dengan mereka. Lihat video dan pelajari lebih lanjut tentang bagaimana Dalí dihidupkan kembali melalui kekuatan AI.
2. Audiobooks, Menggunakan perangkat lunak kloning suara AI, suara selebriti dapat digunakan untuk menceritakan buku, otobiografi dapat dibaca oleh penulis, dan tokoh sejarah dapat menceritakan kisah mereka dengan suara mereka sendiri. Hasilnya adalah pengalaman mendengarkan yang imersif dan berkualitas tinggi.
3. Assistive Tech, Suara sintetis dapat digunakan untuk membantu penyandang disabilitas atau masalah kesehatan yang memengaruhi kemampuan bicara mereka.

Misalnya, orang yang tunarungu atau menderita gangguan seperti Penyakit Parkinson atau ALS dapat meningkatkan kemampuan mereka untuk berkomunikasi menggunakan versi sintetis dari suara dan TTS mereka.

4. Voice Branding, menggunakan suara pribadi dan khusus sebagai merek utama perusahaan Anda dalam asisten percakapan dan aktivitas pemasaran Anda.
5. Digital Dubbing and Animation, Kloning suara memungkinkan untuk membuat suara yang dapat diidentifikasi untuk karakter digital unik atau avatar dalam sistem interaksi manusia-komputer atau untuk produksi konten multimedia dan audiovisual.
6. Smart Assistant, Suara smart assistant terdengar semakin alami karena kemajuan di bidang AI teknologi text-to-speech. Kloning suara memungkinkan personalisasi mereka, melalui penggunaan suara tertentu atau favorit untuk mengembangkan asisten percakapan yang disesuaikan.



Gambar 5.5 Smart Assistant, Sumber: <https://blog.evolvemachinelearners.com/voice-assistant-its-history-twist-and-turn/>

5.1.6 Mendeteksi Deepfake Voice

Karena teknologi suara terus meningkat, memiliki teknologi yang dapat mengenali dan menghentikan penggunaan ucapan palsu untuk penipuan dan penipuan sangat penting.

Anti-spoofing suara, juga disebut deteksi keaktifan suara, adalah teknologi yang mampu membedakan antara suara langsung dan suara yang direkam, dimanipulasi, atau sintetis. Banyak pemalsuan saat ini tidak terlihat oleh telinga manusia, tetapi dapat dideteksi oleh perangkat lunak berbasis AI yang dilatih untuk mengidentifikasi artefak yang tidak ada dalam suara langsung.

Teknologi yang mendeteksi perangkat lunak kloning suara AI pada awalnya dibuat untuk memecahkan masalah spoofing biometrik suara. Di mana biometrik suara mencocokkan suara seseorang dengan templat suara pada file, teknologi anti-spoofing

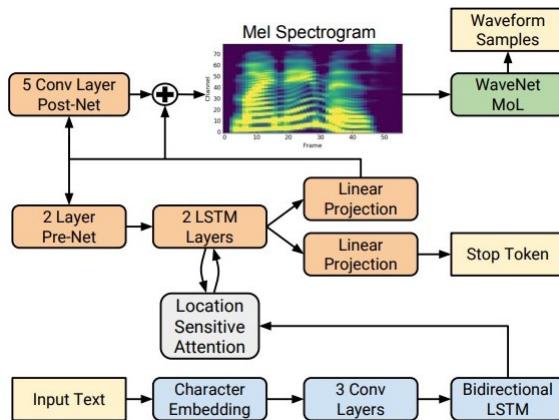
memeriksa untuk memastikan suara itu hidup. Teknologi ini akan terus beradaptasi untuk mengatasi kasus penggunaan tambahan karena penipuan kloning suara menjadi lebih umum. Topik tersebut bahkan menjadi fokus lokakarya FTC baru -baru ini dengan peserta dari ID R&D DARPA, University of Florida, MIT Sloan School of Management, dan program Defending Democracy dari Microsoft.

BAB 6

TACOTRON-2

Tacotron 2 merupakan penelitian yang diteliti oleh Google pada bulan Desember 2016. Tacotron-2 merupakan implementasi jaringan saraf untuk Text-to-Speech Synthesis. Silahkan kunjungi website <https://google.github.io/tacotron/publications/tacotron2/> ini apabila teman-teman penasaran dengan penelitian tersebut. Apabila teman-teman perhatikan dan mencoba mendengarkan sampel suara yang ada di website tersebut bukankah sangat mirip dengan suara asli manusia? Setelah mendengarkan sampel suara tersebut saya menjadi sangat tertarik dengan penelitian tentang Arsitektur Tacotron-2 ini dan mencari tahu serta menerapkan model tersebut dalam project pembuatan voice cloning berbahasa indonesia ini. Cara kerja sistem dijelaskan oleh Jonathan Shen dan Ruoming Pang, Software Engineers, Google Brain and Machine Perception Teams[11].

"Singkatnya cara kerjanya seperti ini: Menggunakan model sequence to sequence yang dioptimalkan untuk TTS untuk memetakan urutan huruf ke urutan fitur yang mengkodekan audio. Fitur-fitur ini, spektrogram audio 80-dimensi dengan bingkai yang dihitung setiap 12,5 milidetik, tidak hanya menangkap pengucapan kata-kata, tetapi juga berbagai seluk-beluk ucapan manusia, termasuk volume, kecepatan, dan intonasi. Akhirnya fitur ini diubah menjadi bentuk gelombang 24 kHz menggunakan arsitektur mirip WaveNet."



Gambar 6.1 Blok Diagram dari Arsitektur Tacotron 2

Bagian pertama dari model yaitu arsitektur Seq2Seq yang bertanggung jawab untuk mengubah teks menjadi mel-spektogram dan spektogram ini dimasukkan dalam model wave-net untuk menghasilkan bentuk gelombang audio. Satu hal yang menarik adalah kedua bagian dari arsitektur Tacotron (Seq2Seq dan vocoder Wavenet) ini dapat dilatih secara mandiri. Saya bekerja pada model Seq2Seq. Modelnya adalah penyiapan enkoder-perhatian-dekoder di mana mereka menggunakan 'Perhatian sensitif lokasi'. Bagian pertama adalah Encoder yang mengubah urutan karakter menjadi vektor penyisipan kata. Representasi ini kemudian dikonsumsi oleh Decoder untuk memprediksi spektogram. Karena saya menggunakan dataset Indonesia, saya memastikan bahwa ruang karakter saya memiliki abjad Indonesia.

Encoder terdiri dari 3 lapisan konvolusi yang masing-masing berisi 512 filter berbentuk 5×1 , diikuti oleh normalisasi batch dan aktivasi ReLU. Bagian selanjutnya adalah jaringan attention yang mengambil keluaran enkoder sebagai masukan dan mencoba meringkas urutan yang disandikan penuh sebagai vektor konteks dengan panjang yang tetap untuk setiap langkah keluaran dekoder. Output dari lapisan konvolusi akhir dilewaskan ke lapisan LSTM dua arah tunggal yang berisi 512 unit (256 di setiap arah) untuk menghasilkan fitur yang dikodekan[12].

6.1 Attention Based Model untuk Speech Recognition

Mekanisme attention yang digunakan di sini memperhitungkan lokasi fokus pada langkah sebelumnya dan fitur urutan input.

Katakanlah kita memiliki data $x = x_1, x_2, x_3, \dots, x_N$. Kami meneruskan data ini ke encoder yang menghasilkan urutan keluaran yang dikodekan $h = h_1, h_2, h_3, \dots, h_N$. $A(i) = \text{Perhatian}(s(i-1), A(i-1), h)$ di mana $s(i-1)$ adalah status decoding sebelumnya dan $A(i-1)$ adalah perataan sebelumnya. $s(i-1)$ adalah 0 untuk iterasi pertama dari langkah pertama. Fungsi perhatian biasanya diimplementasikan dengan menilai setiap elemen dalam h secara terpisah dan kemudian menormalkan skor. $G(i) = A(i, 0)$

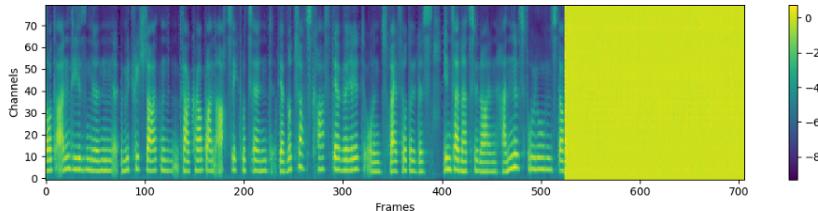
$h(0) + A(i,1) h(1) + \dots + A(i,N) h(N)$ $Y(i)$ Hasilkan ($s(i-1), G(i)$) di mana s adalah output decoding, $A(i)$ adalah vektor bobot perhatian yang disebut keselarasan. Akhirnya, $s(i) = \text{Pengulangan} (s(i-1), G(i), Y(i))$ Pengulangan biasanya LSTM.

6.1.1 Decoder

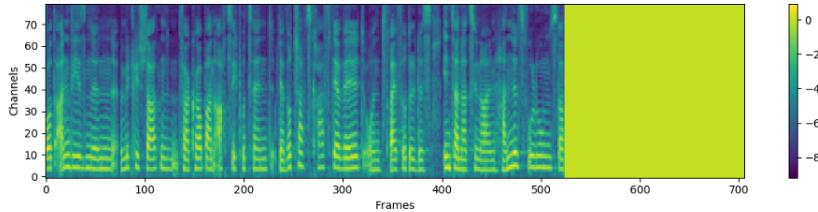
Dekoder adalah jaringan saraf rekuren autoregresif yang memprediksi spektrogram mel dari urutan input yang disandikan satu frame pada satu waktu. Prediksi dari langkah waktu sebelumnya pertama-tama dilewatkan melalui pra-net kecil yang berisi 2 lapisan yang terhubung penuh dari 256 unit ReLU tersembunyi. Output prenet dan vektor konteks perhatian digabungkan dan dilewatkan melalui tumpukan 2 lapisan LSTM uni-directional dengan 1024 unit. Akhirnya, spektrogram mel yang diprediksi dilewatkan melalui post-net convolutional 5-lapisan yang memprediksi residi untuk ditambahkan ke prediksi untuk meningkatkan rekonstruksi keseluruhan. Setiap lapisan pasca jaring terdiri dari 512 filter dengan bentuk 5×1 dengan normalisasi batch, diikuti oleh aktivasi tanh pada semua kecuali lapisan terakhir[11, 13].

6.1.2 Loss Function

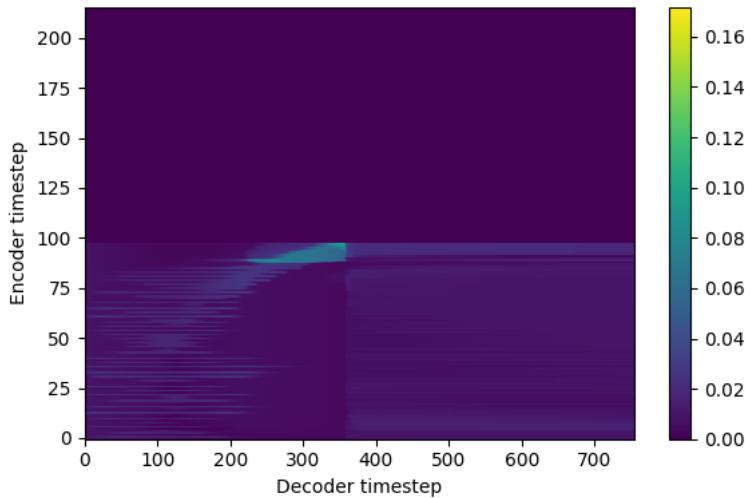
Summed mean squared error (MSE) Sejalan dengan prediksi bingkai spektrogram, rangkaian keluaran LSTM dekoder dan konteks perhatian diproyeksikan ke skalar dan melewati aktivasi sigmoid untuk memprediksi kemungkinan bahwa urutan keluaran telah selesai. Prediksi "token berhenti" ini digunakan selama inferensi untuk memungkinkan model menentukan secara dinamis kapan harus menghentikan pembangkitan alih-alih selalu menghasilkan untuk durasi yang tetap. Saya memutuskan untuk menggunakan pytorch untuk implementasi saya, melacak pelatihan dengan tensorboard , menggunakan GPU gcloud Tesla K80, terhubung ke port server dengan 'ssh -NfL', dan lab jupyter yang banyak digunakan selama pengembangan. [perlengkapan penyelamat hidup] Saya mereferensikan berbagai repositori github untuk memahami makalah, implementasi, mengoreksi bug dalam kode saya sendiri. Karena kompleksitas alami dari pernyataan masalah, saya tidak bisa mendapatkan hasil pidato yang menakjubkan seperti manusia tetapi saya belajar banyak hal tentang Text-to-speech dan itu adalah tujuan utama ketika saya mulai mengerjakan proyek ini. Beberapa hasil untuk referensi:



Gambar 6.2 Spektogram mel yang diprediksi : Seperti yang dapat Anda bandingkan dengan wilayah atas, ia memiliki banyak celah dan masih membutuhkan banyak pelatihan. Sisi kanan (hijau solid) hanya padding dalam satu batch.



Gambar 6.3 Target mel-spektogram



Gambar 6.4 Perhatian (Seperti yang Anda lihat di sisi kiri bawah, sepertinya sedang belajar untuk menyelaraskan tetapi masih membutuhkan sekitar satu minggu pelatihan untuk mendapatkan diagonal yang sempurna untuk perhatian)

Semua gambar yang dihasilkan di atas adalah setelah iterasi 50K (1 iterasi = 1 batch) yaitu 3 hari pelatihan. Model ini membutuhkan sekitar 300 ribu iterasi untuk mendekati seperti manusia. Teman-teman dapat melihat bahwa, spektrogram mel yang diprediksi terlihat cukup bagus bahkan ketika perhatian tidak dipelajari dengan benar. Selamatkan diri Anda dari jebakan dan pedulikan perhatian!

Menerapkan model dan pelatihan ternyata tidak semudah yang saya kira pada awalnya. Saya menemukan banyak masalah yang saya ingin kalian ketahui sebelumnya dan menghemat waktu di GPU.

1. Pelajari dataset yang digunakan. Ini adalah bagian terpenting dari proyek. Dengarkan sampel data, periksa panjang sampel teks, durasi sampel audio, dll. Teman-teman dapat menghemat banyak waktu selama pelatihan, jika teman-teman mengetahui data tersebut dengan baik. M-AILABS mengumumkan kumpulan data pidato besar mereka awal tahun ini. Mereka memiliki kumpulan data ucapan yang sangat banyak dalam berbagai bahasa. Data Angela Merkel dari bagian wanita Jerman, yang memiliki 12 jam pidato dari pidato publik dan wawancaranya. Dataset ini lebih rendah dibandingkan dengan LJSpeech (dataset bahasa Inggris paling populer, 24 jam bicara). Saya menemukan ini hanya ketika saya mulai berlatih dan menghabiskan berhari-hari mengamati hasilnya.
2. TTS sangat mahal secara komputasi . Sebagai mahasiswa, saya baru saja memiliki akses ke satu GPU (Nvidia Tesla K80) di Google cloud. Mengingat struktur dataset yang saya gunakan, GPU saya hanya mengizinkan ukuran batch 8 saat pelatihan. Google mengatakan, mereka melatihnya dengan ukuran batch 64. Saya pertama kali mencoba dengan ukuran batch 2 (karena memori GPU terbatas) dan ketika model gagal menunjukkan konvergensi setelah 2-3 hari pelatihan, saya mengurutkan data saya sesuai panjang teks dan durasi audio, dan memulai training dengan ukuran batch 8. Meskipun demikian, saya tidak dapat mengoptimalkan lebih banyak dengan dataset dan GPU yang saya miliki. Jadi, rencanakan dengan tepat.
3. Teacher-forcing ratio. Dalam teacher-forced training, model dibantu oleh label yang benar yaitu menggunakan kerangka Kebenaran Dasar saat ini untuk memprediksi langkah decoding berikutnya. Tidak jelas dalam makalah tentang rasio apa yang digunakan. Bahkan jika perhatiannya tidak dipelajari, model akan memprediksi kerangka yang baik untuk data pelatihan dalam mode paksaan guru tetapi dalam mode evaluasi itu tidak akan berhasil karena kita tidak memiliki kebenaran dasar (Mengira model itu bekerja sejak prediksi mungkin bagus terlepas dari keberpihakan yang buruk). Saya melakukan pelatihan dengan 1.0, 0.75 dan 0.5 untuk membuat model belajar keberpihakan. Selama mode evaluasi, pemaksaan guru harus dimatikan.
4. Dibutuhkan berhari-hari untuk melatih dan mendapatkan keberpihakan. Ini adalah proses yang sangat rumit untuk melatih sistem TTS. Mungkin perlu sekitar 7–10 hari untuk melatih model asalkan teman-teman memiliki dukungan GPU terbatas. Dan kemudian, men-debug kode dengan model seperti itu, adalah cerita lain.

5. Tuning hyperparameter adalah bagian yang sangat penting dari sistem Tacotron-2. Ukuran batch, tingkat pembelajaran, rasio guru-memaksa, panjang batch adalah beberapa parameter yang harus teman-teman perhatikan juga. Hal-hal bervariasi dengan dataset, jadi sangat sensitif!

Text-to-speech masih merupakan masalah penelitian yang sangat kompleks dan sangat menarik untuk dikerjakan. Pengalaman saya secara keseluruhan luar biasa dan saya belajar banyak hal tentang sistem TTS, bentuk gelombang audio, jaringan berulang, mel-spektrogram, mekanisme perhatian dan saya harap posting ini dapat membantu Anda dalam perjalanan Anda dengan sistem TTS. Di masa mendatang, saya ingin melihat versi model Tacotron-2 yang dioptimalkan, sesuatu yang lebih kuat di berbagai bahasa, lebih mudah dilatih, dan tidak terlalu berat secara komputasi. Jadi, saya hanya akan mengatakan, praproses data Anda dengan baik, sesuaikan hyperparameter Anda, catat semuanya di tensorboard dan mulai dari sekarang.

BAB 7

WAVENET

WaveNet adalah jaringan saraf dalam untuk menghasilkan audio mentah. Itu dibuat oleh para peneliti di perusahaan AI yang berbasis di London, DeepMind . Teknik yang diuraikan dalam makalah pada bulan September 2016, mampu menghasilkan suara mirip manusia yang terdengar relatif realistik dengan secara langsung memodelkan bentuk gelombang menggunakan metode jaringan saraf yang dilatih dengan rekaman ucapan nyata. Pengujian dengan bahasa Inggris dan Mandarin AS dilaporkan menunjukkan bahwa sistem tersebut mengungguli sistem text-to-speech (TTS) terbaik yang ada di Google, meskipun pada 2016 sintesis text-to-speechnya masih kurang meyakinkan daripada ucapan manusia yang sebenarnya. Kemampuan WaveNet untuk menghasilkan bentuk gelombang mentah berarti dapat memodelkan semua jenis audio, termasuk musik[14]. Menghasilkan ucapan dari teks adalah tugas yang semakin umum berkat popularitas perangkat lunak seperti Siri Apple, Cortana Microsoft, Amazon Alexa, dan Asisten Google.

Sebagian besar sistem tersebut menggunakan variasi teknik yang melibatkan fragmen suara yang digabungkan bersama untuk membentuk suara dan kata yang dapat dikenali, yang paling umum disebut TTS gabungan. Ini terdiri dari perpustakaan besar fragmen pidato, direkam dari satu pembicara yang kemudian digabungkan untuk menghasilkan kata-kata dan suara yang lengkap. Hasilnya terdengar tidak wajar,

dengan irama dan nada yang aneh. Ketergantungan pada perpustakaan yang direkam juga membuat sulit untuk memodifikasi atau mengubah suara.

Teknik lain, yang dikenal sebagai TTS parametrik, menggunakan model matematika untuk menciptakan kembali suara yang kemudian dirangkai menjadi kata dan kalimat. Informasi yang diperlukan untuk menghasilkan suara disimpan dalam parameter model. Karakteristik pidato keluaran dikendalikan melalui input ke model, sedangkan pidato biasanya dibuat menggunakan synthesizer suara yang dikenal sebagai vocoder. Ini juga dapat menghasilkan audio yang terdengar tidak wajar.

WaveNet adalah jenis feedforward neural network yang dikenal sebagai deep convolutional neural network (CNN). Di WaveNet, CNN mengambil sinyal mentah sebagai input dan mensintesis output satu sampel pada satu waktu. Ia melakukan dengan mengambil sampel dari distribusi softmax (yaitu kategorikal) dari nilai sinyal yang dikodekan menggunakan transformasi -law companding dan dikuantisasi ke 256 nilai yang mungkin.

Menurut makalah penelitian DeepMind September 2016 Original WaveNet: A Generative Model for Raw Audio, jaringan tersebut diberi makan bentuk gelombang nyata dari pidato dalam bahasa Inggris dan Mandarin. Saat ini melewati jaringan, ia mempelajari seperangkat aturan untuk menggambarkan bagaimana bentuk gelombang audio berkembang dari waktu ke waktu. Jaringan terlatih kemudian dapat digunakan untuk membuat bentuk gelombang seperti ucapan baru pada 16.000 sampel per detik[14].

WaveNet mampu memodelkan suara yang berbeda secara akurat, dengan aksen dan nada input yang berkorelasi dengan output. Misalnya, jika dilatih dengan bahasa Jerman, ia menghasilkan pidato bahasa Jerman. Kemampuan ini juga berarti bahwa jika WaveNet diberi masukan lain – seperti musik – keluarannya akan berupa musik. Pada saat dirilis, DeepMind menunjukkan bahwa WaveNet dapat menghasilkan bentuk gelombang yang terdengar seperti musik klasik.

Makalah Januari 2019 Pembelajaran representasi ucapan tanpa pengawasan menggunakan autoencoder WaveNet merinci metode untuk berhasil meningkatkan pengenalan otomatis yang tepat dan diskriminasi antara fitur dinamis dan statis untuk "pertukaran konten", terutama termasuk menukar suara pada rekaman audio yang ada, di untuk membuatnya lebih dapat diandalkan. Makalah tindak lanjut lainnya, Sample Efficient Adaptive Text-to-Speech, tertanggal September 2018 (revisi terbaru Januari 2019), menyatakan bahwa DeepMind telah berhasil mengurangi jumlah minimum rekaman kehidupan nyata yang diperlukan untuk mengambil sampel suara yang ada melalui WaveNet untuk "hanya beberapa menit data audio" sambil mempertahankan hasil berkualitas tinggi.

Kemampuannya untuk mengkloning suara telah menimbulkan kekhawatiran etis tentang kemampuan WaveNet untuk meniru suara orang hidup dan mati. Menurut artikel BBC 2016 , perusahaan yang mengerjakan teknologi kloning suara serupa (seperti Adobe Voco) bermaksud untuk memasukkan tanda air yang tidak terdengar oleh manusia untuk mencegah pemalsuan, sambil mempertahankan kloning suara yang memuaskan, misalnya, kebutuhan tujuan industri hiburan akan memiliki kompleksitas yang jauh lebih rendah dan menggunakan metode yang berbeda dari yang diperlukan untuk menipu metode pembuktian forensik dan perangkat ID elektronik,

sehingga suara alami dan suara yang dikloning untuk tujuan industri hiburan masih dapat dengan mudah dibedakan dengan analisis teknologi.

Pada saat peluncurannya, DeepMind mengatakan bahwa WaveNet membutuhkan terlalu banyak kekuatan pemrosesan komputasi untuk digunakan dalam aplikasi dunia nyata. Pada Oktober 2017, Google mengumumkan peningkatan kinerja 1.000 kali lipat bersama dengan kualitas suara yang lebih baik. WaveNet kemudian digunakan untuk menghasilkan suara Asisten Google untuk bahasa Inggris AS dan Jepang di semua platform Google. Pada bulan November 2017, peneliti DeepMind merilis makalah penelitian yang merinci metode yang diusulkan untuk "menghasilkan sampel ucapan dengan ketelitian tinggi lebih dari 20 kali lebih cepat daripada waktu nyata", yang disebut "Distilasi Kepadatan Probabilitas". Pada konferensi pengembangan I/O tahunan pada Mei 2018, diumumkan bahwa suara Asisten Google baru tersedia dan dimungkinkan oleh WaveNet; WaveNet sangat mengurangi jumlah rekaman audio yang diperlukan untuk membuat model suara dengan memodelkan audio mentah dari sampel aktor suara.

7.1 Talking Machine

Mengizinkan orang berkomunikasi dengan mesin adalah impian lama interaksi manusia-komputer. Kemampuan komputer untuk memahami ucapan alami telah mengalami revolusi dalam beberapa tahun terakhir dengan penerapan jaringan saraf dalam (misalnya, Google Voice Search). Namun, menghasilkan pidato dengan komputer — sebuah proses yang biasanya disebut sebagai sintesis ucapan atau text-to-speech (TTS) — sebagian besar masih didasarkan pada apa yang disebut TTS concatenative, di mana database yang sangat besar dari fragmen pidato pendek direkam dari satu penutur dan kemudian digabungkan kembali untuk membentuk ujaran yang lengkap. Hal ini membuat sulit untuk memodifikasi suara (misalnya beralih ke pembicara yang berbeda, atau mengubah penekanan atau emosi ucapan mereka) tanpa merekam database yang sama sekali baru.

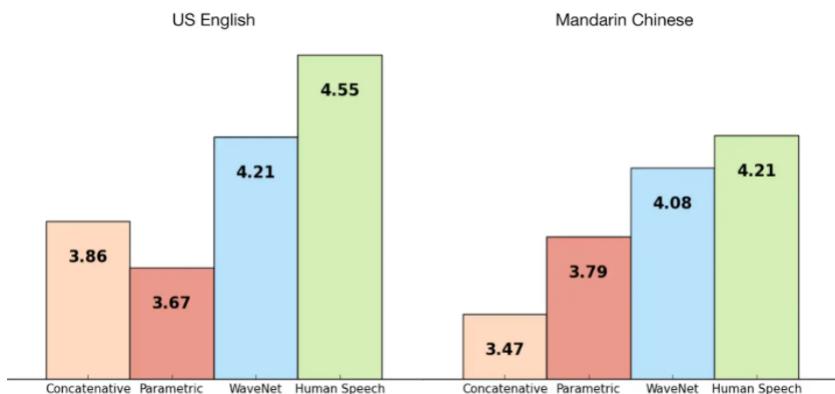
Hal ini menyebabkan permintaan yang besar untuk TTS parametrik, di mana semua informasi yang diperlukan untuk menghasilkan data disimpan dalam parameter model, dan isi serta karakteristik pidato dapat dikontrol melalui input ke model. Sejauh ini, bagaimanapun, TTS parametrik cenderung terdengar kurang alami dari pada concatenative. Model parametrik yang ada biasanya menghasilkan sinyal audio dengan melewatkannya melalui algoritma pemrosesan sinyal yang dikenal sebagai vocoder.

WaveNet mengubah paradigma ini dengan secara langsung memodelkan bentuk gelombang mentah dari sinyal audio, satu sampel pada satu waktu. Selain menghasilkan ucapan yang terdengar lebih alami, menggunakan bentuk gelombang mentah berarti WaveNet dapat memodelkan semua jenis audio, termasuk musik.

7.2 WaveNet Improving State of The Arts

Kami melatih WaveNet menggunakan beberapa kumpulan data TTS Google sehingga kami dapat mengevaluasi kinerjanya. Gambar berikut menunjukkan kualitas WaveNets pada skala 1 sampai 5, dibandingkan dengan sistem TTS terbaik Google saat ini (parametrik dan concatenative), dan dengan ucapan manusia menggunakan Mean Opinion Scores (MOS) . MOS adalah ukuran standar untuk tes kualitas suara subjektif, dan diperoleh dalam tes buta dengan subjek manusia (dari lebih dari 500 peringkat pada 100 kalimat tes). Seperti yang dapat kita lihat, WaveNets mengurangi kesenjangan antara keadaan seni dan kinerja tingkat manusia lebih dari 50% untuk bahasa Inggris AS dan Mandarin.

Untuk bahasa Cina dan Inggris, sistem TTS Google saat ini dianggap sebagai yang terbaik di seluruh dunia, jadi meningkatkan keduanya dengan satu model merupakan pencapaian besar.



Gambar 7.1 Perbandingan WaveNet dengan Model Lain

Untuk menarik perbandingan antara WaveNet dan pendekatan sintesis ucapan yang ada, tes Subjektif 5-skala Mean Opinion Score (MOS) dilakukan. Dalam tes MOS, subjek (manusia) disajikan dengan sampel ucapan yang dihasilkan dari salah satu sistem sintesis ucapan dan diminta untuk menilai kealamian sampel ucapan dalam skor skala lima poin (1: Buruk, 2: Buruk, 3 : Cukup, 4: Baik, 5: Sangat Baik).

Terlihat jelas dari gambar 7.1 bahwa WaveNet mencapai kealamian di atas 4,0 dalam tes MOS 5 skala, yang secara signifikan lebih baik daripada sistem dasar lainnya dan sangat dekat dengan ucapan manusia yang sebenarnya. Periksa contoh pidato di blog DeepMind untuk menyadari perbedaan dalam kealamian pidato yang disintesis dalam pendekatan ini! Selain mampu menghasilkan sampel audio sebagai output, WaveNet dapat dengan mudah dikondisikan pada berbagai karakteristik ucapan seperti teks, identitas pembicara, dll untuk menghasilkan ucapan yang sesuai dengan kebutuhan kita. Itulah yang membuatnya semakin seru.

7.3 Wavenet. The Generative Model

Model Generatif. Apa itu? Mengingat titik data tidak berlabel generik, model generatif mencoba mempelajari distribusi probabilitas apa yang menghasilkan titik data tersebut dengan tujuan menghasilkan titik data baru (mirip dengan titik data input) dengan memanfaatkan distribusi yang dipelajari. Model generatif dapat memodelkan distribusi probabilitas dengan cara yang berbeda, secara implisit (dengan kepadatan yang dapat dilacak atau mendekati) atau secara eksplisit. Ketika kita mengatakan bahwa model generatif memodelkan distribusi probabilitas secara eksplisit, yang kita maksudkan adalah kita mendefinisikan distribusi probabilitas secara eksplisit dan mencoba untuk mengadaptasi distribusi sesuai dengan input titik data yang tidak berlabel. Berbeda dengan ini, model generatif implisit mempelajari distribusi probabilitas yang dapat langsung digunakan untuk mengambil sampel titik data baru tanpa perlu mendefinisikannya secara eksplisit. GAN (Generative Adversarial Networks), cawan suci Deep Learning akhir-akhir ini, termasuk dalam kategori model generatif implisit. Sedangkan WaveNet dan sepupunya Pixel CNNs/RNNs adalah model generatif eksplisit.

Bagaimana model WaveNet distribusi probabilitas secara eksplisit? WaveNet mencoba untuk memodelkan distribusi probabilitas gabungan dari aliran data X sebagai produk dari distribusi bersyarat elemen-bijaksana untuk setiap elemen x_t dalam aliran. Jadi untuk bentuk gelombang audio mentah $X = X_1, X_2, X_3 \dots X_T$, probabilitas gabungan difaktorkan sebagai berikut:

$$p(\mathbf{x}) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1})$$

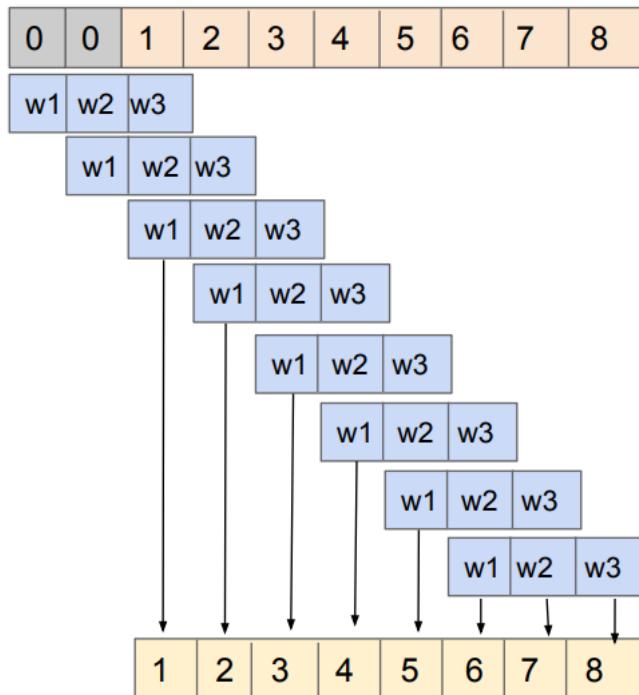
Gambar 7.2 Probabilitas Gabungan

Oleh karena itu, setiap sampel audio X_t dikondisikan pada sampel pada semua langkah waktu sebelumnya. Bukankah ini terlihat mirip dengan model peramalan deret waktu, di mana pengamatan pada langkah waktu bergantung pada pengamatan pada langkah waktu sebelumnya (Itulah yang kami coba modelkan menggunakan masing-masing istilah distribusi bersyarat ini)? Faktanya, WaveNet adalah model autoregresif. Bagaimana kita melanjutkan untuk memodelkan istilah distribusi bersyarat ini? RNN atau LSTM menjadi model sekuensial non-linier yang kuat adalah pilihan yang paling jelas. Sebenarnya, Pixel RNN menggunakan ide yang sama untuk menghasilkan gambar sintetis yang terlihat mirip dengan gambar input. Bisakah kita menggunakan ide ini untuk menghasilkan pidato sintetis? Pidato sampel pada frekuensi minimal 16KHz, yang berarti bahwa setidaknya ada 16.000 sampel dalam satu detik pidato. RNN atau LSTM tidak akan dapat memodelkan dependensi waktu yang begitu lama (dari pesanan 10.000 langkah waktu) karena mereka dapat memodelkan dependensi waktu paling banyak dari urutan 100 langkah waktu dan dengan demikian mereka tidak cocok untuk diucapkan perpaduan. Selain itu, melatih RNN,

LSTM akan sangat lambat karena tidak dapat diparalelkan karena sifatnya yang berurutan secara inheren. Bisakah kita menggunakan CNN untuk menangani ini? Tunggu, CNN? Bagaimana? Mirip dengan ide yang telah digunakan di Pixel CNN.

7.4 CNN

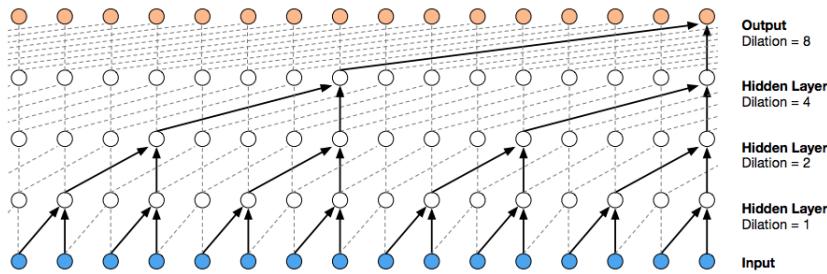
Mengapa mencoba CNN? CNN biasanya lebih cepat untuk dilatih dibandingkan dengan RNN atau LSTM, terutama bila diterapkan pada urutan 1-D yang panjang karena operasi yang terkait dengan setiap posisi topeng atau filter yang berbelit-belit dapat dilakukan secara paralel dan independen. Pelatihan lebih cepat. Kedengarannya bagus! Bagaimana dengan properti autoregressive (output dari time-step bergantung pada output dari time-step sebelumnya saja dan bukan pada output dari time-steps di masa depan)? Di situlah Konvolusi Kausal beraksi. Konvolusi Kausal 1-D dapat dengan mudah diimplementasikan dengan mengisi ke kiri urutan input 1-D, di mana konvolusi akan dilakukan, dengan jumlah nol yang sesuai dan kemudian melakukan konvolusi yang valid. Causal Convolutions akan memungkinkan kita untuk memodelkan lebih lama (memungkinkan kita untuk menentukan panjang look-back juga) dependensi waktu dibandingkan dengan RNN atau LSTM.



Gambar 7.3 Konvolusi Kausal untuk memastikan bahwa model tidak dapat melanggar urutan di mana kita memodelkan data.

Kita telah menangani pelanggaran autoregressive dengan mudah. Tetapi bagaimana dengan mengelola panjang lihat-belakang pesanan ribuan sampel (sehingga model kami melihat kembali setidaknya satu detik audio sebelum menarik kesimpulan tentang output pada langkah waktu saat ini)? Hal termudah yang terlintas dalam pikiran adalah meningkatkan ukuran filter yang cukup untuk melihat ke belakang dengan panjang yang sesuai, tetapi apakah ini benar-benar membantu? Dalam pandangan saya, melakukan hal itu akan membuat model kurang non-linier yang pada gilirannya akan mempersulit model kita untuk mempelajari dependensi temporal yang kompleks dan dengan demikian membatasi kinerja model kita. Hal berikutnya yang mungkin muncul di benak Anda adalah menambah jumlah lapisan dalam jaringan saraf. Ini mungkin membantu. Tapi itu akan menjadi komputasi yang tidak layak karena ukuran reseptif atau melihat ke belakang untuk langkah waktu dalam output meningkat secara linier dengan jumlah lapisan tersembunyi dalam model dan tidak bijaksana secara komputasi bagi kita untuk memiliki model dengan beberapa ribu lapisan tersembunyi. Sekarang kita dibatasi untuk membatasi jumlah lapisan tersembunyi serta ukuran filter namun meningkatkan panjang tampilan kembali? Bagaimana kita akan melakukan ini? Konvolusi yang melebar akan membantu kita di sini. Sekarang kita dibatasi untuk membatasi jumlah lapisan tersembunyi serta ukuran filter namun meningkatkan panjang tampilan kembali? Bagaimana kita akan melakukan ini? Konvolusi yang melebar akan membantu kita di sini.

Konvolusi melebar mencoba meningkatkan panjang tampilan atau ukuran bidang reseptif dengan menerapkan filter pada area yang lebih besar dari panjangnya dan melewatkannya nilai input dengan langkah tertentu. Ini setara dengan konvolusi dengan filter yang lebih besar yang berasal dari filter asli dengan melebarkannya dengan nol tetapi secara signifikan lebih efisien. Di WaveNet, beberapa lapisan konvolusi melebar ditumpuk satu sama lain untuk memiliki bidang reseptif yang sangat besar hanya dengan beberapa lapisan. Peningkatan eksponensial (dua kali lipat) faktor pelebaran menghasilkan pertumbuhan bidang reseptif eksponensial dengan kedalaman.



Gambar 7.4 Tumpukan lapisan konvolusi kausal yang melebar: Menggandakan faktor pelebaran dengan setiap lapisan menghasilkan pertumbuhan bidang reseptif $O(2^n)$.

7.5 Softmax Distribution. Mu-law Companding

Untuk memodelkan probabilitas bersyarat, WaveNet menggunakan distribusi softmax (kategorikal) daripada model campuran lainnya karena distribusi kategoris lebih fleksibel dan dapat lebih mudah memodelkan distribusi arbitrer karena tidak membuat asumsi tentang bentuknya. Audio mentah biasanya disimpan sebagai urutan nilai integer 16-bit (-32,768 hingga 32,767) dan menggunakan lapisan softmax untuk menampilkan distribusi probabilitas akan mengharuskan model kami untuk menghasilkan 65.535 nilai pada setiap langkah waktu. Bukankah ini akan memperlambat model kita? Itu pasti akan berhasil. Apa yang bisa kita lakukan tentang ini? Mengurangi kedalaman bit akan berhasil di sini. Jika kita terus menggunakan kedalaman bit linier (dibagi dengan 256) pengurangan akan memiliki dampak yang lebih besar pada sampel dengan amplitudo rendah daripada pada sampel dengan amplitudo tinggi. Pertimbangkan sampel 16-bit yang awalnya 32767, nilai positif tertinggi yang mungkin. Dikonversi menjadi 8 bit, nilai sampel menjadi 127 ($32767/256 = 127$ dengan sisa 255) dan error dari pembulatan adalah $255/32768$. Ini adalah kesalahan kuantisasi kurang dari 1%. Tetapi bandingkan ini dengan kesalahan untuk sampel 16-bit dengan magnitudo terendah, antara 0 dan 255. Intinya adalah bahwa dengan metode pengurangan kedalaman bit linier, pembulatan ke bawah memiliki dampak yang lebih besar pada sampel dengan amplitudo rendah daripada pada sampel dengan amplitudo tinggi. Jika kita dapat mendistribusikan kembali nilai sampel sehingga ada lebih banyak level kuantisasi pada amplitudo yang lebih rendah dan lebih sedikit level kuantisasi pada amplitudo yang lebih tinggi, kesalahan kuantisasi ini dapat dikurangi. Itu sebabnya, alih-alih menggunakan kuantisasi linier sederhana, pengomposisian Mu-law (kuantisasi non-linier) telah digunakan di WaveNet.

$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 + \mu |x_t|)}{\ln(1 + \mu)}$$

Gambar 7.5 Ekspresi untuk melakukan kompilasi Mu-law menghasilkan output yang direkonstruksi lebih baik (terdengar mirip dengan audio asli) daripada kuantisasi linier.

Dalam ekspresi di atas, $1 \downarrow X_t \downarrow 1$ (pengambilan sampel audio pada langkah waktu yang berbeda disampel ulang dari 32,768...32,767 hingga 1...1) dan $\mu = 255$. Ini akan membuat model hanya menghasilkan 256 nilai, bukan 65,535 nilai pada setiap langkah waktu, mempercepat pelatihan dan inferensi.

7.6 Gated Activations. Skip and Residual Connections.

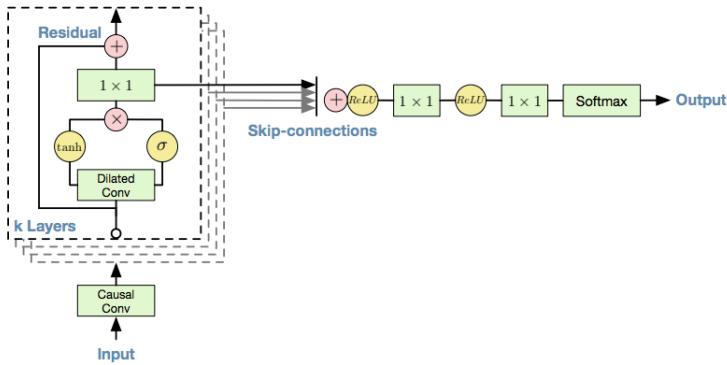
Fungsi aktivasi non-linier sangat penting dalam setiap model pembelajaran mendalam untuk mempelajari hubungan kompleks antara output dan input. RELU awalnya digunakan di WaveNet, tetapi setelah melakukan eksperimen, disadari bahwa

non-linearitas tan-hiperbolik gated dengan aktivasi sigmoid (terinspirasi dari Pixel CNN) bekerja lebih baik untuk WaveNet.

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x})$$

Gambar 7.6 Ekspresi untuk fungsi aktivasi terjaga keamanannya digunakan di WaveNet.

Dalam ekspresi di atas, W mewakili filter yang dapat dipelajari, $*$ mewakili operator konvolusi dan titik yang dilingkari mewakili perkalian elemen-bijaksana. Koneksi sisa, menambahkan output dari lapisan bawah ke output dari lapisan di atas, dan Lewati koneksi, menambahkan output dari lapisan bawah langsung ke lapisan output, telah terbukti berguna dalam mengurangi waktu konvergensi jaringan saraf dan melatih jaringan yang lebih dalam. . Dengan demikian, mereka telah digunakan dalam arsitektur WaveNet, seperti yang ditunjukkan pada gambar di bawah ini.



Gambar 7.7 Arsitektur WaveNet: menampilkan fungsi aktivasi yang terjaga keamanannya, melewatan koneksi dan koneksi residual.

7.7 Conditioning. Local. Global.

Kami belum berbicara tentang bagaimana kami mengkondisikan ucapan keluaran berdasarkan berbagai fitur seperti identitas pembicara, teks yang sesuai, dll. Suara keluaran WaveNet dapat dikondisikan dalam dua cara: Pengkondisian global, keluaran bias fitur tunggal dari semua langkah waktu seperti identitas pembicara atau pengkondisian lokal, memiliki banyak fitur, pada kenyataannya serangkaian fitur waktu yang berbeda, yang membiaskan keluaran pada langkah waktu yang berbeda seperti teks yang mendasari pidato. Jika kita mengungkapkan ini secara lebih formal, maka ini berarti pengenalan parameter baru dalam istilah distribusi bersyarat (H_t dalam pengkondisian lokal dan H dalam pengkondisian global) dalam model yang sebenarnya.

$$p(\mathbf{x} \mid \mathbf{h}) = \prod_{t=1}^T p(x_t \mid x_1, \dots, x_{t-1}, \mathbf{h})$$

Gambar 7.8 Modifikasi istilah distribusi bersyarat, setelah pengenalan input pengkondisian.

Dalam pengkondisian lokal, kemungkinan rangkaian waktu input pengkondisian mungkin memiliki panjang yang lebih rendah daripada audio dan untuk pengkondisian lokal, kedua rangkaian waktu harus memiliki panjang yang sama. Untuk men cocokkan panjangnya, kita dapat menggunakan CNN yang ditransposisikan (semacam skema upsampling yang dapat dipelajari) atau skema upsampling lainnya untuk menambah panjang input pengkondisian.

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k}^T \mathbf{h}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k}^T \mathbf{h})$$

Gambar 7.9 Ekspresi setelah pengenalan istilah bias h.

Dalam ekspresi di atas, V adalah proyeksi linier yang dapat dipelajari yang pada dasarnya melayani dua tujuan mengubah h ke ukuran yang benar dan mempelajari bobot yang benar untuk bias output.

7.8 Great Model. Fast Training. Slow Inference?

Arsitektur WaveNet, apa pun yang telah kita bicarakan sampai sekarang, menangkap ketergantungan dan pengkondisian temporal yang kompleks dengan sangat baik. Selain itu, karena sangat dapat diparalelkan, ini sangat cepat pada waktu pelatihan. Tapi bagaimana dengan kesimpulannya? Karena output pada langkah waktu bergantung pada output pada langkah waktu sebelumnya, pengambilan sampel audio baru akan secara inheren berurutan. Butuh sekitar 1 menit waktu GPU untuk menghasilkan 1 detik output. Jika Google akan menerapkan model ini di Asisten mereka, maka akan memakan waktu berjam-jam untuk memikirkan pertanyaan sederhana seperti “Hai Google! Bagaimana cuacanya?”. Bagaimana mereka meningkatkan waktu inferensi? IAF adalah jawabannya.

7.9 Normalising Flows. IAF.

Normalisasi aliran? Aliran normalisasi adalah urutan transformasi yang mempelajari pemetaan (bijeksi) dari kepadatan probabilitas sederhana (seperti Gauss) menjadi distribusi kompleks yang kaya. Misalkan Anda memiliki cukup titik sampel dari distribusi probabilitas $q(z)$ dan distribusi probabilitas $q(x)$, maka aliran normalisasi dapat digunakan untuk mempelajari transformasi yang akan memetakan titik sampel

dari $q(x)$ ke pemetaan yang sesuai dalam distribusi $q(z)$. Bagaimana ini dilakukan? Mari kita pertimbangkan transformasi f , pemetaan yang dapat dibalik dan halus. Jika kita menggunakan pemetaan ini untuk mengubah variabel acak z dengan distribusi $q(z)$, variabel acak yang dihasilkan $z' = f(z)$ memiliki distribusi $q(z')$:

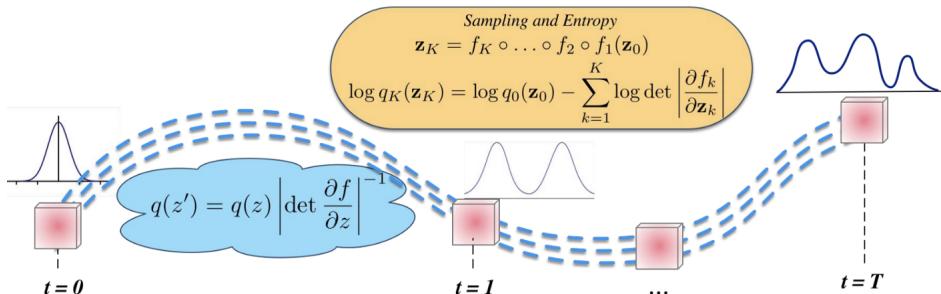
$$q(\mathbf{z}') = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}'} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$

Untuk mendapatkan intuisi tentang bagaimana kami mencapai ekspresi ini untuk distribusi variabel acak yang ditransformasikan, lihat posting blog ini oleh Eric Jang. Apakah satu transformasi f cukup? Faktanya, kita dapat membangun kepadatan kompleks yang sewenang-wenang dengan menyusun beberapa transformasi sederhana dan secara berturut-turut menerapkan ekspresi di atas. Kerapatan $q_K(z)$ yang diperoleh dengan mentransformasikan variabel acak z_0 secara berurutan dengan distribusi q_0 melalui rantai transformasi K f_k adalah:

$$\mathbf{z}_K = f_K \circ \dots \circ f_2 \circ f_1(\mathbf{z}_0)$$

$$\ln q_K(\mathbf{z}_K) = \ln q_0(\mathbf{z}_0) - \sum_{k=1}^K \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

Masing-masing transformasi ini dapat dengan mudah dimodelkan menggunakan perkalian matriks (dengan nilai yang dapat dipelajari), diikuti oleh non-linier seperti ReLU. Ideya kemudian adalah memperbarui parameter transformasi yang dapat dipelajari, dengan salah satu algoritme pengoptimalan favorit Anda dengan men-optimalkan kemungkinan (log-likelihood) titik sampel dari $q(x)$ di bawah distribusi probabilitas yang ditransformasikan $q_K(z)$. Ini akan membuat distribusi $q_K(z)$ sangat mirip dengan $q(x)$, dan dengan demikian mempelajari pemetaan yang sesuai dari $q(z)$ ke $q(x)$.



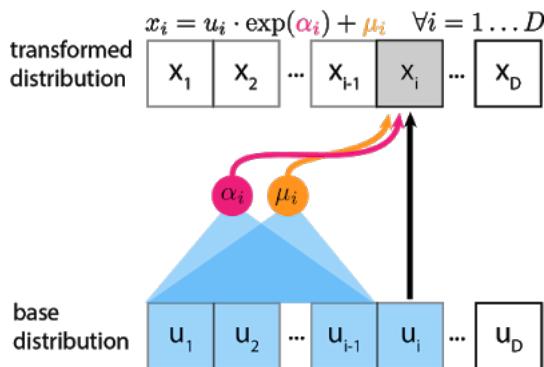
Gambar 7.10 Distribusi mengalir melalui urutan transformasi yang dapat dibalik.

Bagaimana ide normalisasi aliran dapat membantu kita dalam inferensi cepat? Ingat, WaveNet menjadi model generatif, tidak melakukan apa-apa selain mencoba mempelajari distribusi probabilitas yang akan menghasilkan data pelatihan dan karena ini adalah model generatif yang didefinisikan secara eksplisit (dengan kepadatan yang dapat dilacak), kita dapat dengan mudah mempelajari transformasi yang dapat memetakan titik dari titik sederhana. distribusi seperti Gaussian ke distribusi Kategoris kompleks yang dipelajari oleh WaveNet. Jika aliran normalisasi yang dipelajari yang memiliki skema inferensi cepat, masalah inferensi lambat kita di WaveNet dapat dengan mudah diurutkan. IAF (Inverse Autoregressive Flow) sangat cocok dengan ide ini. Di IAF, idenya adalah pertama-tama menggambar sampel acak dari $z \sim \text{Logistic}(0, 1)$ dan kemudian menerapkan transformasi berikut ke sampel yang diambil,

$$x_t = z_t \cdot s(z_{<t}, \theta) + \mu(z_{<t}, \theta)$$

Gambar 7.11 Transformasi skala dan pergeseran sederhana pada z_t di mana faktor penskalaan (s) dan faktor pergeseran (μ) dihitung dengan menggunakan parameter yang dapat dipelajari dan nilai dalam sampel input z dari langkah waktu sebelumnya.

Untuk menampilkan distribusi yang benar untuk langkah waktu x_t , aliran autoregresif terbalik dapat secara implisit menyimpulkan apa yang akan dihasilkannya pada langkah waktu sebelumnya x_1, \dots, x_{t-1} berdasarkan input noise z_1, \dots, z_{t-1} , yang memungkinkan untuk menampilkan semua x_t secara paralel diberikan z_t . Gambar di bawah akan membuat segalanya lebih jelas (perhatikan notasi yang diubah).



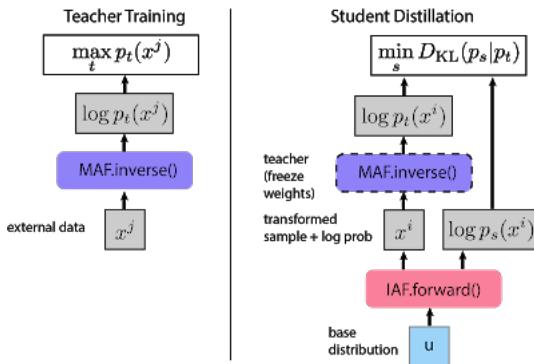
Gambar 7.12 Dalam Inverse Autoregressive Flow, output pada langkah waktu yang berbeda dapat dihitung secara paralel karena output dari langkah waktu tidak bergantung pada output dari langkah waktu sebelumnya.

IAF memiliki skema inferensi yang cepat (dan bahkan probabilitas bersyarat yang dapat dilacak dapat dihitung secara paralel), tetapi mereka lebih lambat untuk di-

latih. Mengapa? Karena jika Anda jika kami diberi titik data baru dan diminta untuk mengevaluasi kepadatan, kami perlu memulihkan Anda dan proses ini secara inheren berurutan lambat. Parallel WaveNet, memanfaatkan fakta ini dan muncul dengan konsep pelatihan IAF (Student WaveNet) menggunakan WaveNet sederhana (Guru WaveNet).

7.10 Parallel. Faster. WaveNet.

Di WaveNet paralel, idenya adalah untuk memanfaatkan fakta bahwa IAF memiliki skema inferensi cepat. Jadi, pada fase pertama kami melatih model WaveNet sederhana (yang kami sebut sebagai Pelatihan Guru). Pada fase kedua, kami membuka bobot Teacher WaveNet dan menggunakan untuk melatih IAF (Student Distillation). Idenya adalah pertama-tama mengambil sampel acak dari $\text{Logistic}(0, 1)$, melewatkinya melalui IAF secara paralel. Ini akan memberi kita titik dalam distribusi yang ditransformasikan dan probabilitas bersyarat yang terkait juga. Idenya adalah untuk melewati titik ini dalam distribusi yang ditransformasikan di seluruh Teacher WaveNet sederhana, yang akan menghasilkan probabilitas bersyarat sehubungan dengan Teacher WaveNet yang sudah terlatih. Kemudian kami mencoba untuk meminimalkan perbedaan KL antara probabilitas bersyarat yang diterima dari kedua model. Ini akan memungkinkan IAF (Student WaveNet) untuk mempelajari distribusi probabilitas yang hampir sama dengan pengajarnya, dan hasil memvalidasi fakta ini karena ada perbedaan yang hampir dapat diabaikan dalam skor MOS skala 5 antara keluaran yang diterima dari WaveNet Guru dan Siswa.



Gambar 7.13 Prosedur pelatihan Paralel WaveNet.

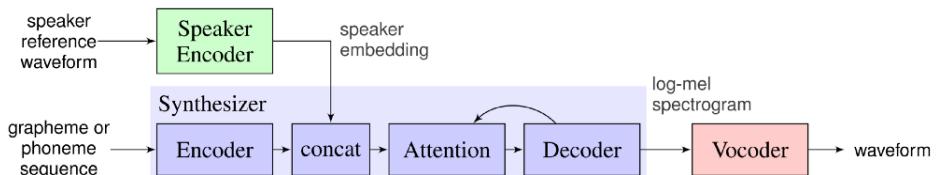
Apakah ini cukup cepat untuk disebarluaskan? Ya itu. Bahkan, ia mampu menghasilkan sampel ucapan lebih dari 20 kali lebih cepat daripada waktu nyata. Namun masih ada masalah, setiap kali kita perlu melatih kembali model kita, pertama-tama kita akan melatih WaveNet Guru dan kemudian WaveNet Siswa. Selain itu, kinerja Student WaveNet sangat bergantung pada seberapa baik Teacher WaveNet dilatih. Tapi secara keseluruhan bagus untuk pergi ke penyebaran.

BAB 8

MULTI-SPEAKER VOICE CLONING

8.1 Model SV2TTS

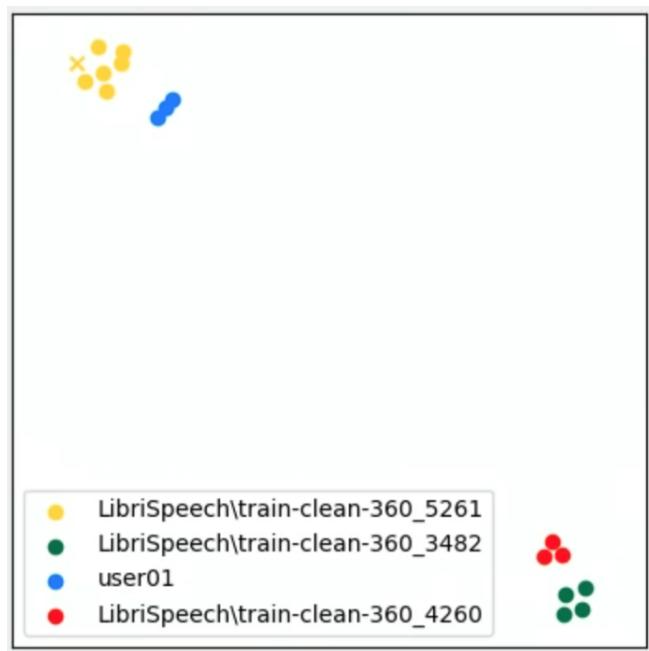
Model Speech Vector to TTS (SV2TTS) terdiri dari tiga bagian, masing-masing dilatih secara individual. Hal ini memungkinkan setiap bagian dilatih pada data independen, sehingga mengurangi kebutuhan untuk mendapatkan data multispeaker berkualitas tinggi.



Gambar 8.1 Arsitektur SV2TTS umum Sumber: Jia, Zhang, dan Weiss et al.

8.1.1 Speaker Encoder

Bagian pertama dari model SV2TTS adalah encoder speaker. Tugas speaker encoder adalah mengambil beberapa input audio (dikodekan sebagai bingkai spektrogram mel), dari speaker tertentu, dan mengeluarkan embedding yang menangkap “bagaimana suara speaker”. Encoder pembicara tidak peduli dengan kata-kata yang diucapkan pembicara, atau tentang kebisingan di latar belakang, yang dia pedulikan hanyalah suara pembicara, misalnya, suara bernada tinggi/rendah, aksen, nada, dll. Semua fitur ini digabungkan menjadi vektor berdimensi rendah, yang secara formal dikenal sebagai vektor-d, atau secara informal sebagai penyematan speaker. Akibatnya, ujaran yang diucapkan oleh penutur yang sama akan saling berdekatan pada penyematan penutur, sedangkan tuturan yang diucapkan oleh penutur yang berbeda akan berjauhan pada penyematan penutur[9, 15].



Gambar 8.2 Representasi visual dari embeddings, setiap warna adalah speaker yang berbeda[1]

Untuk belajar menghasilkan embeddings ini, penulis menjelaskan proses berikut:

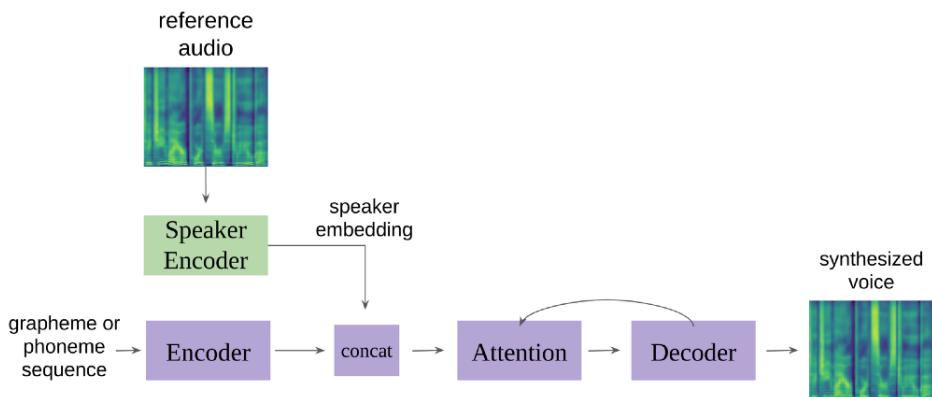
1. Pertama, contoh audio ucapan disegmentasi menjadi klip 1,6 detik tanpa transkip dan diubah menjadi spektrogram mel.
2. Kemudian speaker encoder dilatih untuk mengambil dua sampel audio dan memutuskan apakah speaker yang sama memproduksinya atau tidak. Sebagai produk

sampingan, ini memaksa encoder speaker untuk membuat embeddings yang mewakili suara speaker.

Proses pelatihan mirip dengan jaringan saraf siam jika Anda sudah familiar dengan mereka.

8.1.2 Synthesizer

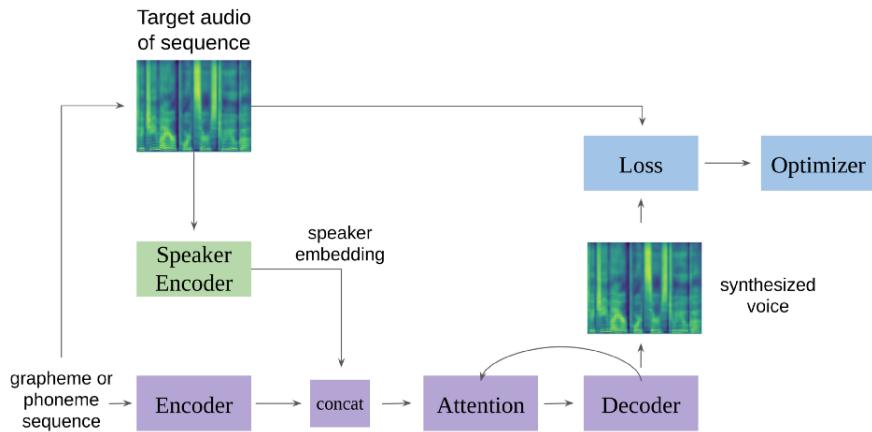
Synthesizer adalah bagian dari SV2TTS yang menganalisis input teks untuk membuat spektrogram mel, yang kemudian diubah oleh vocoder menjadi suara. Synthesizer mengambil urutan teks — dipetakan ke fonem (unit terkecil dari suara manusia, misalnya, suara yang Anda buat saat mengucapkan 'a'), bersama dengan embeddings yang dihasilkan oleh encoder speaker, dan menggunakan arsitektur Tacotron 2 untuk menghasilkan bingkai dari spektrogram mel secara berulang



Gambar 8.3 Model Arsitektur Multi-Speaker Voice Cloning [1]

Berikut Proses Training Synthesizer:

1. Pertama, kami mengumpulkan urutan fonem dan spektrogram mel dari pembicara yang mengucapkan kalimat itu.
2. Kemudian, spektrogram mel diteruskan ke encoder speaker untuk menghasilkan embedding speaker.
3. Selanjutnya, pembuat enkode penyintesis menggabungkan penyandian urutan fonemnya dengan penyematan speaker.
4. Spektrogram mel dihasilkan secara berulang oleh decoder dan bagian perhatian dari synthesizer
5. Terakhir, spektrogram mel dibandingkan dengan target awal untuk menghasilkan kerugian, yang kemudian dioptimalkan



Gambar 8.4 Training Synthesizer[1]

8.1.3 Vocoder

Pada titik ini, synthesizer telah membuat spektrogram mel, tetapi kami masih belum dapat mendengarkan apa pun. Untuk mengubah spektrogram mel menjadi gelombang audio mentah, penulis menggunakan vocoder. Vocoder khusus yang digunakan di sini didasarkan pada model WaveNet DeepMind , yang menghasilkan bentuk gelombang audio mentah dari teks, dan pada satu titik canggih untuk sistem TTS[14].



Gambar 8.5 Vocoder[1]

8.1.4 Alur Model SV2TTS

berikut adalah alur modelnya:

1. Encoder speaker mendengarkan sampel audio yang diberikan dan menghasilkan embedding
2. Synthesizer mengambil daftar fonem dan penyematan speaker, kemudian menghasilkan spektrogram mel
3. Vocoder saraf menguraikan spektrogram mel menjadi bentuk gelombang audio yang dapat kita Dengarkan

8.1.5 Mengukur Kualitas Hasil

Penulis menggunakan penilai manusia untuk mengukur kealamian dan kesamaan ucapan yang dihasilkan model. Kealamian mengukur seberapa "manusia" ucapan itu terdengar, sementara kesamaan mengukur seberapa mirip suara pidato yang disintesis dengan pembicara aslinya. Penilai berinteraksi melalui GUI yang terlihat seperti:

Instructions

In this task, your job is to evaluate if the two speech audio samples are from the same speaker. Please release this task if any of the following are true:

- You think you do not have good listening ability.
- There is considerable background noise (street noise, loud fan/air-conditioner, open TV/radio, people talking, etc).
- For any reason, you can't hear the audio samples.

Task

How are you listening to the speech samples?

- Headphones, with no noise in the background. I am listening to the speech sample using headphones and there is no noise around me (people talking, music playing, air-conditioners, fans, etc.).
 Headphones, with some low-level noise in the background. I am listening to the speech sample using headphones and there is some low-level noise around me (people talking, music playing, air-conditioners, fans, etc.).
 Audio speakers or other.

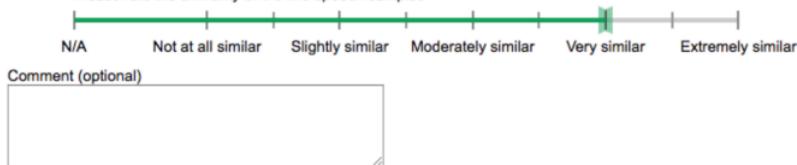
Speaker Similarity Between Two Speech Samples

Please listen to the two speech samples below (Sample A and Sample B) and rate how similar they are. Your rating should reflect your evaluation of how close the voices of the two speakers sound. You **should not judge the content, grammar, or audio quality** of the sentences; instead, just focus on the similarity of the speakers to one another.

Speech samples (please listen at least **two times each**)

Sample A	▶ 0:00 / 0:02	<input type="range"/>	🔊	<input type="range"/>
Sample B	▶ 0:00 / 0:03	<input type="range"/>	🔊	<input type="range"/>

Please rate the similarity of the two speech samples



Gambar 8.6 Pengukuran Kualitas Model[1]

Teknik penggunaan rating crowdsourced seperti ini sering disebut dengan Mean Opinion Score atau MOS. Penulis menemukan bahwa hasil akhir untuk kealamian akhirnya terlihat seperti:

System	VCTK Seen	VCTK Unseen	LibriSpeech Seen	LibriSpeech Unseen
Ground truth	4.43 ± 0.05	4.49 ± 0.05	4.49 ± 0.05	4.42 ± 0.07
Embedding table	4.12 ± 0.06	N/A	3.90 ± 0.06	N/A
Proposed model	4.07 ± 0.06	4.20 ± 0.06	3.89 ± 0.06	4.12 ± 0.05

Gambar 8.7 MOS [1]

Jika Anda ingin analisis lengkap ini, saya akan merekomendasikan membaca bagian 3.1 dari makalah asli. Namun, secara ringkas:

1. Model SV2TTS yang diusulkan mencapai sekitar 4,0 MOS di semua kumpulan data.
2. LibriSpeech (salah satu set data pelatihan) diklasifikasikan sebagai kurang alami, karena tidak ada tanda baca dalam transkrip, sehingga menyulitkan model untuk mempelajari jeda.
3. Sistem tabel penyematan menggunakan tabel pencarian penyematan speaker tetapi sebaliknya memiliki arsitektur yang sama.
4. Karena memiliki tabel pencarian, tidak dapat digeneralisasi, dan oleh karena itu tidak dapat dievaluasi pada kumpulan data yang tidak terlihat.
5. Kealamian dari contoh yang tidak terlihat dan yang terlihat sangat mirip, yang sangat mengesankan.

Dan untuk kesamaan ucapan, hasilnya terlihat seperti:

System	Speaker Set	VCTK	LibriSpeech
Ground truth	Same speaker	4.67 ± 0.04	4.33 ± 0.08
Ground truth	Same gender	2.25 ± 0.07	1.83 ± 0.07
Ground truth	Different gender	1.15 ± 0.04	1.04 ± 0.03
Embedding table	Seen	4.17 ± 0.06	3.70 ± 0.08
Proposed model	Seen	4.22 ± 0.06	3.28 ± 0.08
Proposed model	Unseen	3.28 ± 0.07	3.03 ± 0.09

Gambar 8.8 Speech Similarity[1]

Sekali lagi, Anda mungkin ingin membaca bagian 3.2 dari makalah asli untuk analisis lengkapnya. Tapi secara ringkas:

1. Skor lebih tinggi untuk VCTK (set data pelatihan lainnya), menunjukkan bentuk dataset yang lebih terstruktur.
2. Skor model SV2TTS antara "cukup mirip" dan "sangat mirip" pada skala evaluasi untuk pembicara yang tidak terlihat.
3. Meskipun ini sulit diukur secara matematis, model secara keseluruhan menangkap karakteristik pembicara secara akurat.

BAB 9

TUTORIAL PEMBUATAN MULTI-SPEAKER VOICE CLONING

9.1 Pengenalan Anaconda

Anaconda merupakan sebuah software yang mendistribusikan bahasa pemrograman python dan R untuk keperluan komputasi ilmiah seperti data science, machine learning, data processing skala-luas, analisis prediksi, dan lain sebaginya. Anaconda memiliki anaconda navigator yang didalamnya terdapat software-software yang dapat digunakan untuk membuat program python dan R. Salah satu software yang akan kita gunakan yaitu Spyder. Sebelum kita membuat program kita harus menginstall anaconda terlebih dahulu. Instalasi pada kali ini menggunakan 2 sistem operasi yaitu Windows 10 x64 dan Ubuntu 19.04. Hal yang dibutuhkan adalah laptop dengan os windows atau ubuntu dan internet.

9.1.1 Instalasi Anaconda 3 Windows 10 x64

Hal yang harus diperhatikan sebelum melakukan instalasi *Anaconda Python*

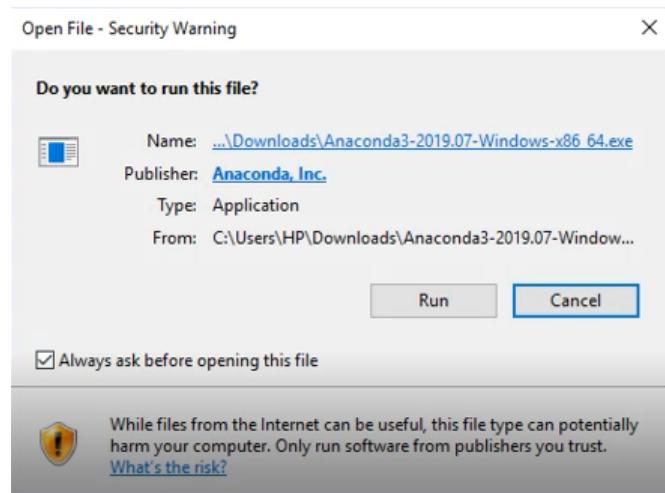
1. *Download Anaconda Python*

<https://www.anaconda.com/distribution/>

2. Perhatikan versi dari sistem operasi yang digunakan (versi 32bit atau 64bit)
3. Download file anaconda yang sesuai dengan versi sistem operasi (32bit atau 64bit)

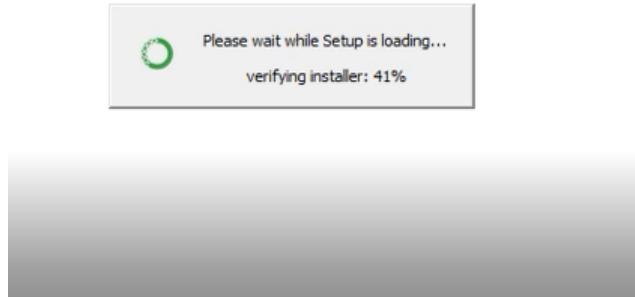
Berikut langkah-langkah instalasi anaconda.

1. Buka aplikasi *installer Anaconda* yang telah didownload lalu akan muncul gambar 9.1, lalu pilih run untuk menjalankan proses instalasi.



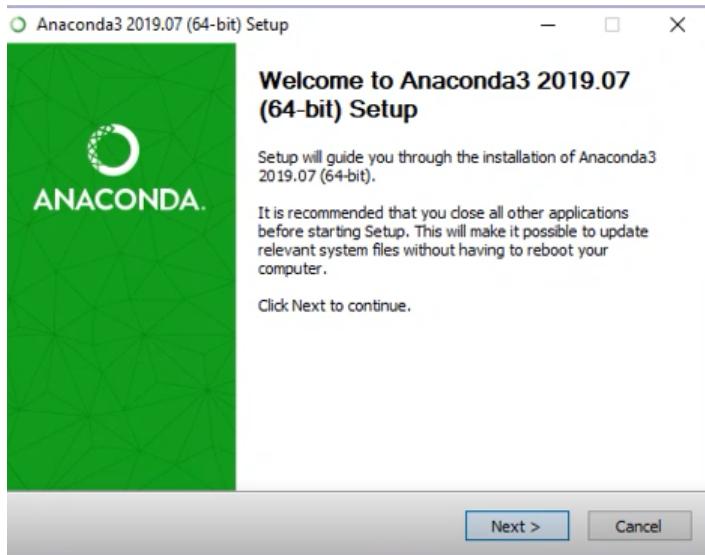
Gambar 9.1 Run Setup Anaconda

2. Tunggu hingga *setup loading* selesai seperti pada gambar 9.2.



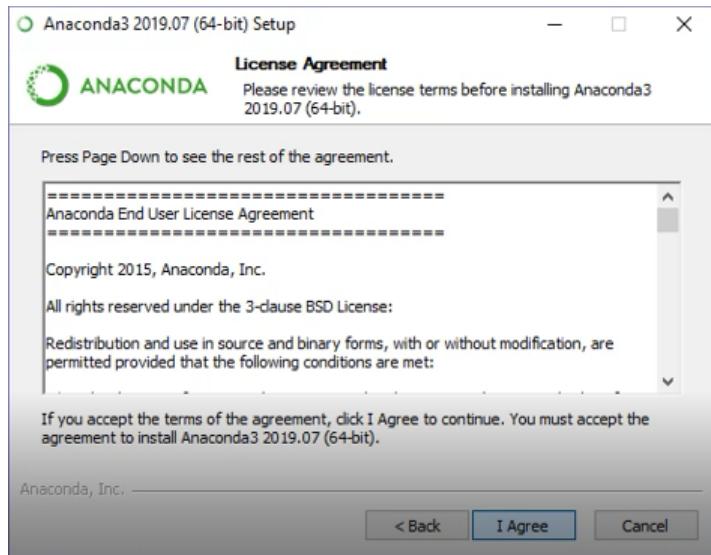
Gambar 9.2 Setup Loading

3. Jika *setup loading* telah selesai, maka klik *next* seperti pada gambar 9.3.



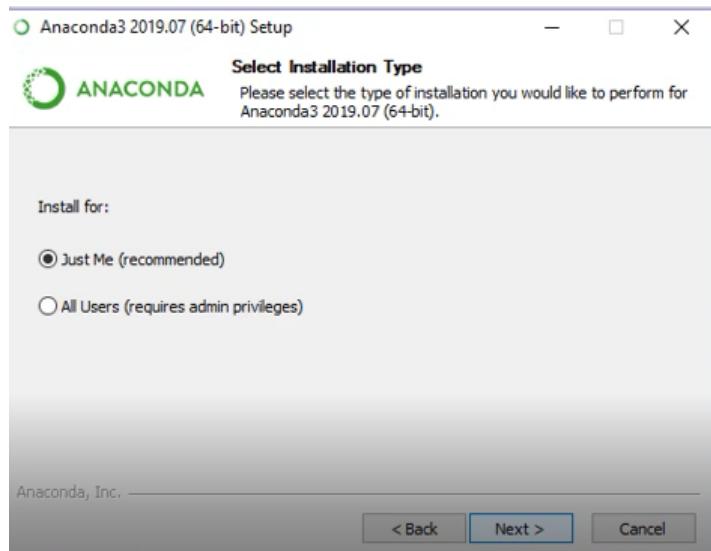
Gambar 9.3 Welcome to Anaconda Setup

4. Pada *License Agreement* klik *I Agree* karena jika teman-teman tidak menyetujui lisensi anaconda maka teman-teman tidak akan bisa melanjutkan proses instalasi. lakukan langkah ini seperti pada gambar 9.4



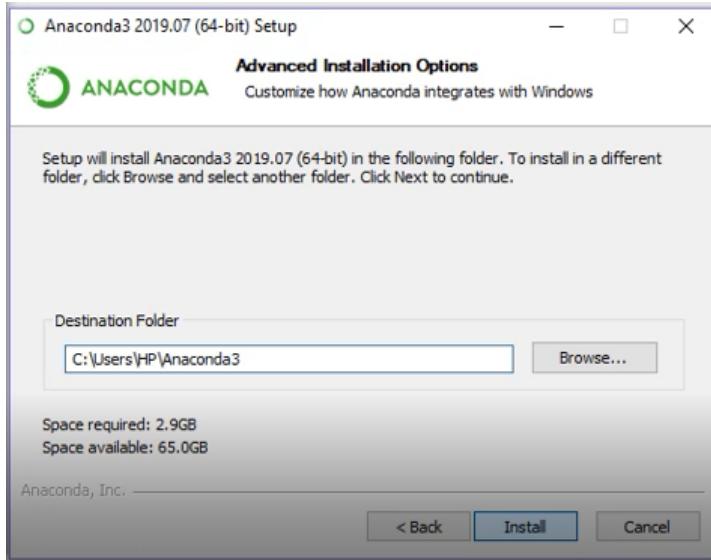
Gambar 9.4 License Agreement

5. Kemudian pilih *Just Me(Recomended)* agar sesuai dengan komputer yang digunakan, kemudian klik *next* seperti pada gambar 9.5.



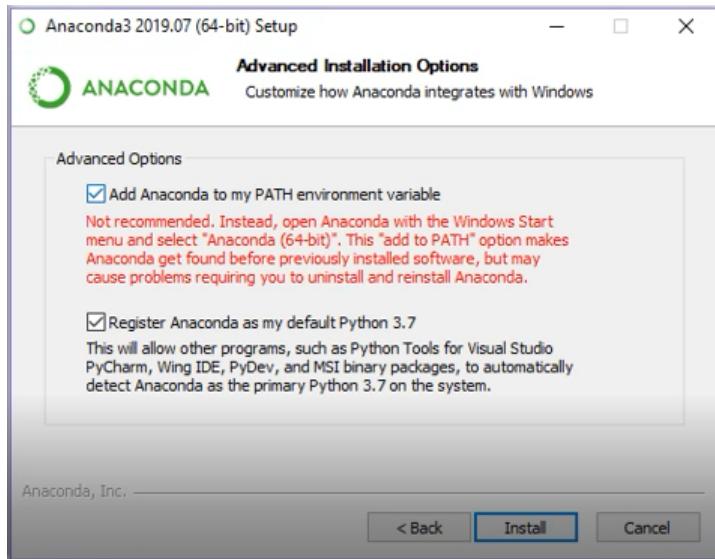
Gambar 9.5 Just Me(recomended)

6. Kemudian pilih direktori tempat kita akan *menginstall anaconda* seperti pada gambar 9.6



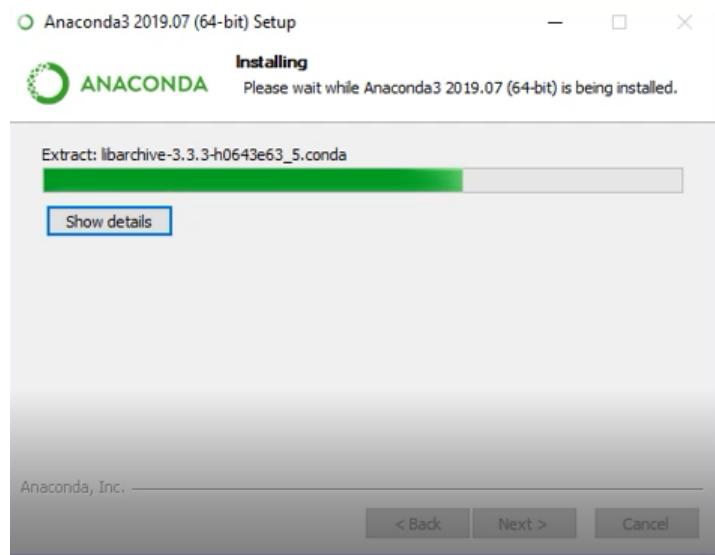
Gambar 9.6 Pilih lokasi

7. Kemudian centang *Add Anaconda to my Path environment variable*, agar saat melakukan instalasi *package anaconda*, *package* tersebut akan langsung tertuju ke *path anaconda* tidak ke aplikasi yang lain. kemudian Klik *install* seperti pada gambar 9.7.



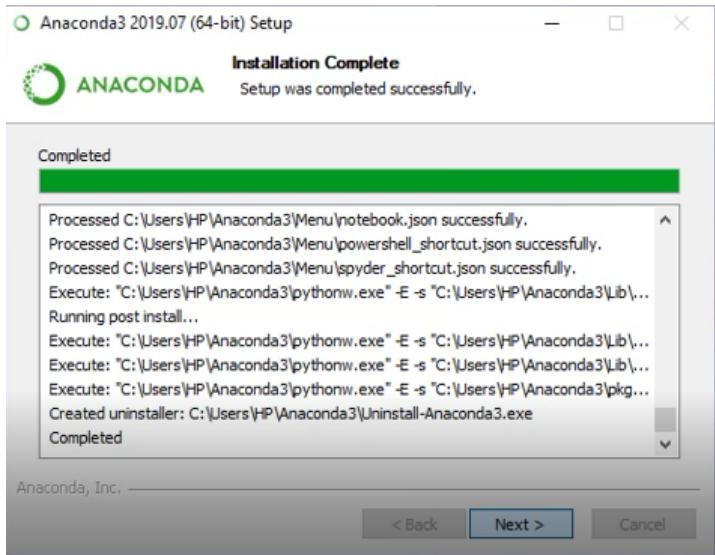
Gambar 9.7 Centang Anaconda to my PATH

8. Tunggu sampai proses *installasi* selesai seperti pada gambar 9.8.



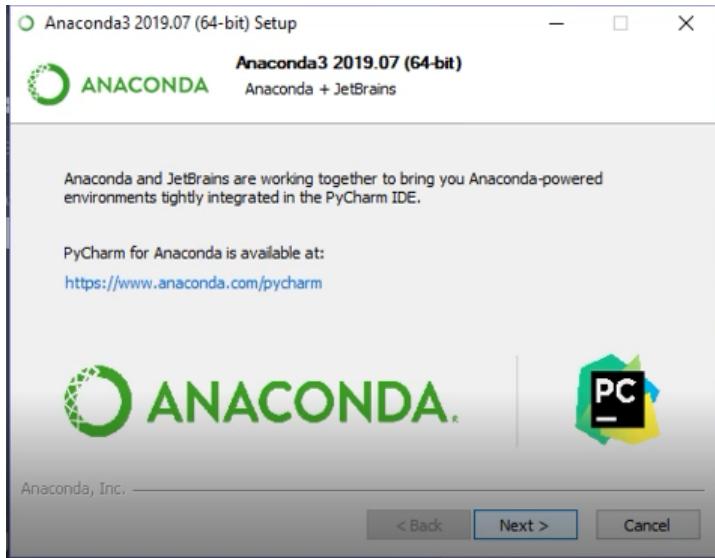
Gambar 9.8 Waiting Installation Complete

9. Apabila instalasi telah selesai maka akan terlihat seperti gambar 9.9, kemudian klik *next*



Gambar 9.9 Installation Complete

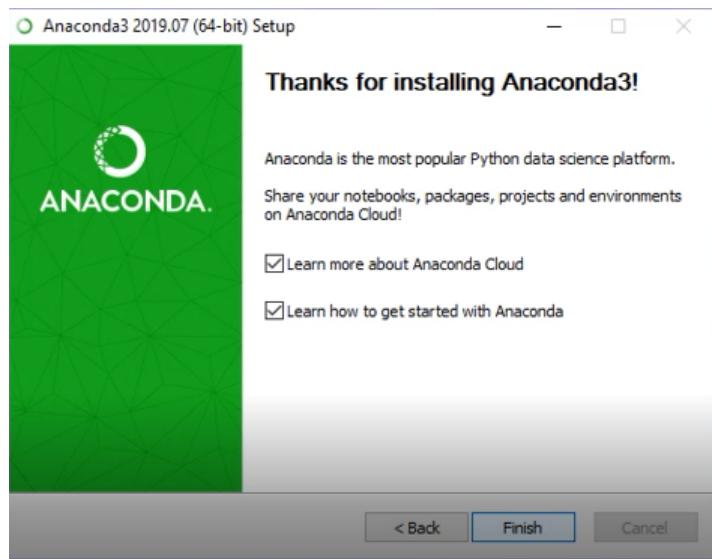
10. apabila muncul gambar 9.10, maka klik *next*



Gambar 9.10 Anaconda+JetBrains

11. Jika instalasi telah selesai maka akan ada ucapan terima kasih telah menginstall anaconda 3 seperti pada gambar 9.11, hal ini menandakan bahwa teman-teman

telah selesai dan berhasil melakukan instalasi anaconda. Kemudian klik *finish* untuk mengakhiri instalasi.



Gambar 9.11 *Thanks for install Anaconda*

9.1.2 Update Anaconda dan Spyder

Kenapa kita harus melakukan update anaconda dan spyder? melakukan update diperlukan agar software yang kita gunakan merupakan software yang terbaru, karena versi lama dan versi baru akan memiliki banyak perbedaan dan akan menjadi masalah nantinya ketika kita membuat program atau mengimportkan modul-modul python yang digunakan. Berikut cara mengupdate Spyder:

1. Buka anaconda prompt, lalu ketikkan perintah conda update spyder
2. Konfirmasi update dengan mengetikkan y, lalu tekan enter
3. Tunggu hingga installan selesai

Berikut cara mengupdate Anaconda:

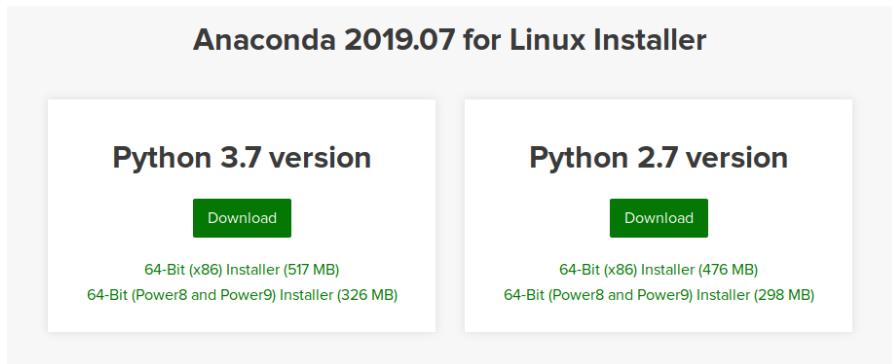
1. Buka anaconda prompt, lalu ketikkan perintah conda update anaconda
2. Konfirmasi update anaconda dengan mengetikkan y dan kemudian tekan enter
3. Tunggu hingga installan selesai

9.1.3 Instalasi Anaconda Ubuntu 19.04

Ikuti langkah berikut untuk instalasi python pada Ubuntu 19.04:

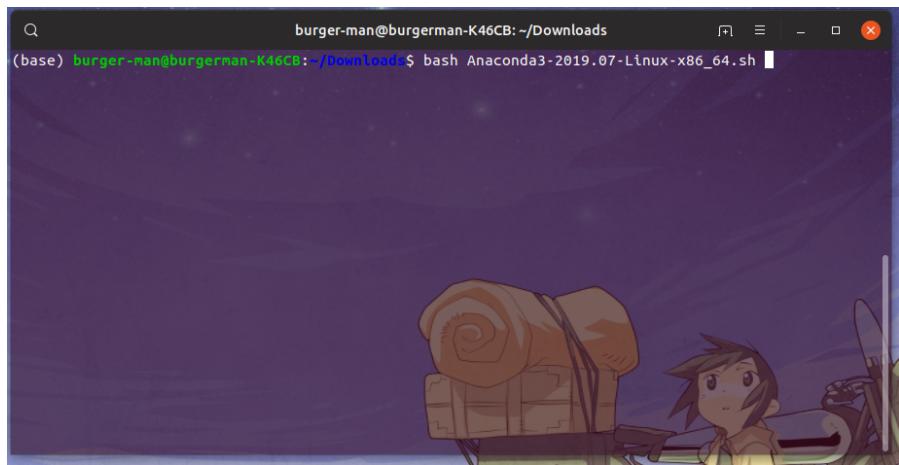
- Pertama kita kunjungi situs

<https://www.anaconda.com/distribution/#download-section> seperti gambar 9.12 dan pilih **64-Bit (x86) Installer (517 MB)**



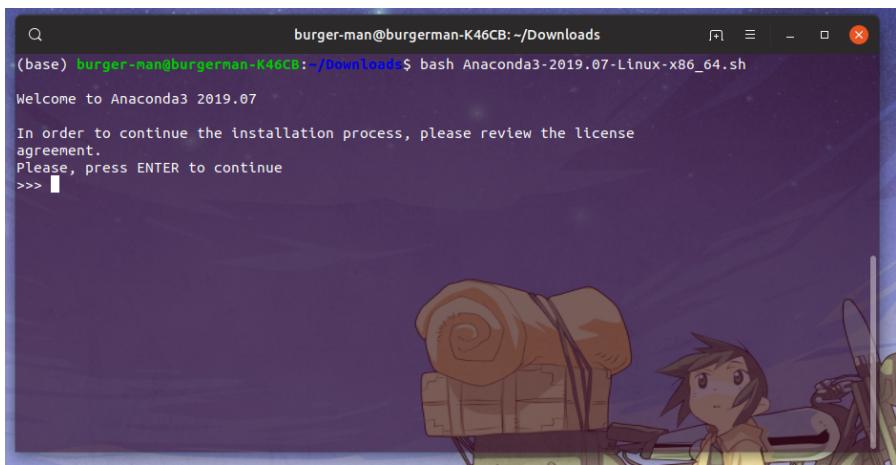
Gambar 9.12 Gambar halaman download

- Kedua kita buka **terminal** kita lalu arahkan ke direktori kita menyimpan file download anaconda
- Ketiga kita ketikkan sebagai berikut **bash namafileanaconda.sh** lalu enter, contoh seberti gambar 9.13



Gambar 9.13 Gambar install anaconda

- Setelah itu, tekan **ENTER** saja seperti gambar 9.14



Gambar 9.14 Gambar eksekusi anaconda

- Lalu akan muncul sebuah tulisan **End User License Agreement** seperti gambar 9.15, tekan **ENTER** dan tahan hingga seperti gambar



Gambar 9.15 Gambar anaconda license agreement



Gambar 9.16 Gambar perintah yes or no

6. Lalu setelah muncul perintah '**yes**' or '**no**' ketik **yes** lalu enter
 7. Setelah itu muncul path direktori instalasi anaconda kita seperti gambar 9.17 lalu tekan enter

```
Q                                     burger-man@burgerman-K46CB: ~/Downloads

Please answer 'yes' or 'no':'
>>>
Please answer 'yes' or 'no':'
>>> yes

Anaconda3 will now be installed into this location:
/home/burger-man/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

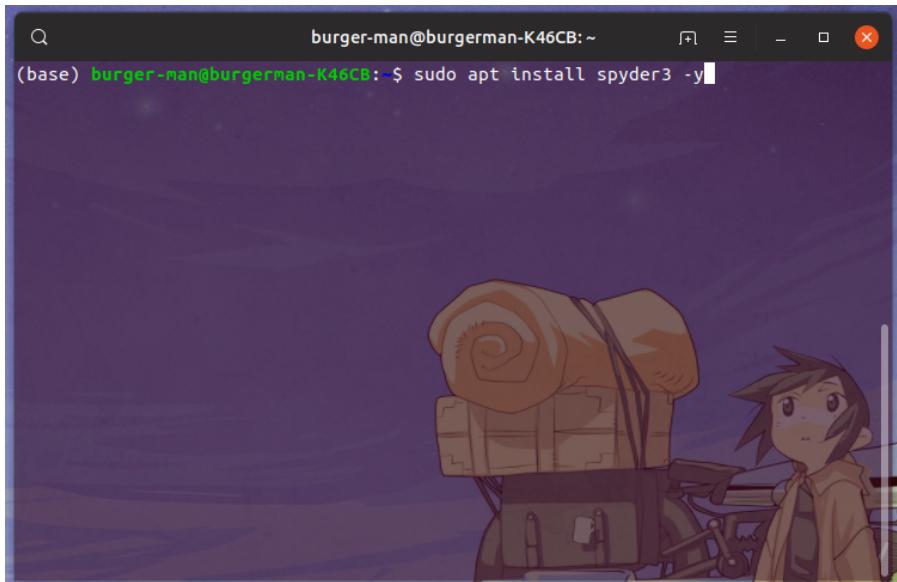
[~/home/burger-man/anaconda3] >>> █
```



Gambar 9.17 Gambar path anaconda

Setelah kita selesai instalasi anaconda jangan lupa juga untuk menginstal spyder ide, caranya seperti berikut:

- (a) ketikkan perintah `sudo apt install spyder3 -y` seperti gambar 9.18



Gambar 9.18 Gambar perintah install spyder3

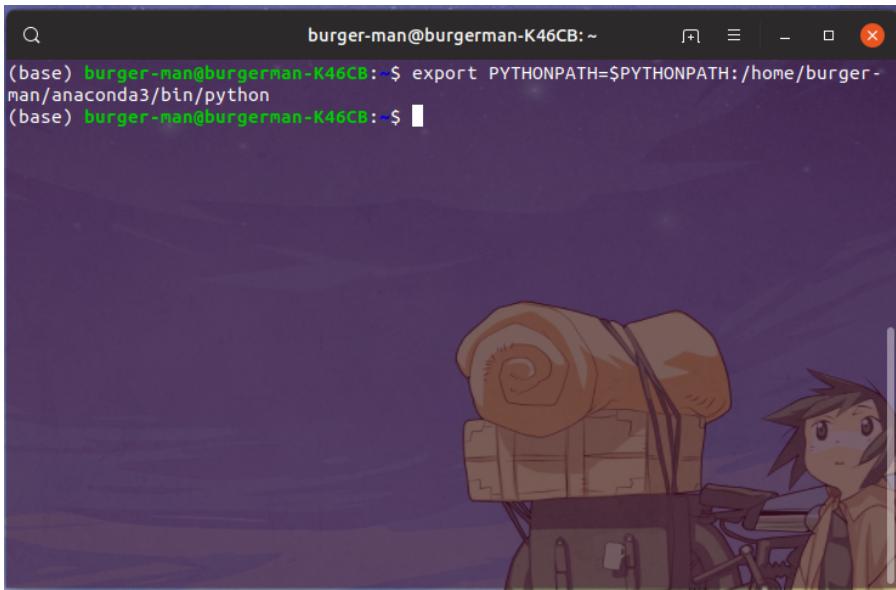
(b) lalu jalankan dengan perintah *spyder* atau *spyder3*

9.1.4 Konfigurasi *Python*

Setelah kita selesai instal Anaconda dan Spyder, selanjutnya kita akan mempelajari bagaimana cara setting environments python kita, caranya sebagai berikut:

1. pertama kita buka terminal kita lalu ketikkan perintah berikut, contoh seperti gambar 9.19, lalu tekan enter

```
| export PYTHONPATH=$PYTHONPATH: pathinstallasipythonkalian
```

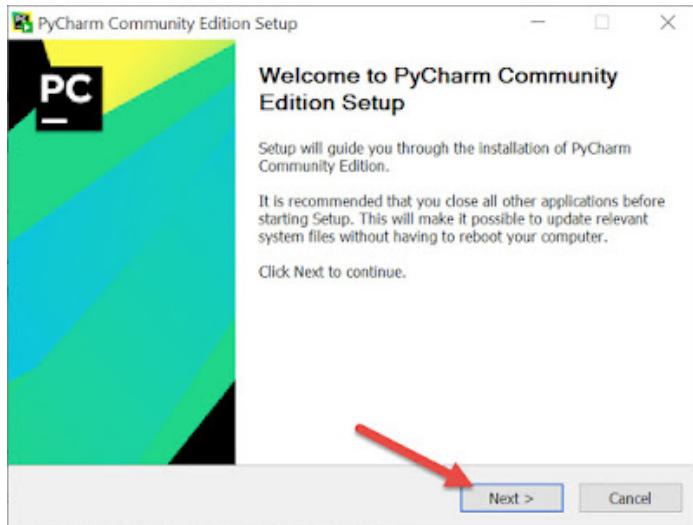


Gambar 9.19 Gambar setpath

9.2 Instalasi Pycharm

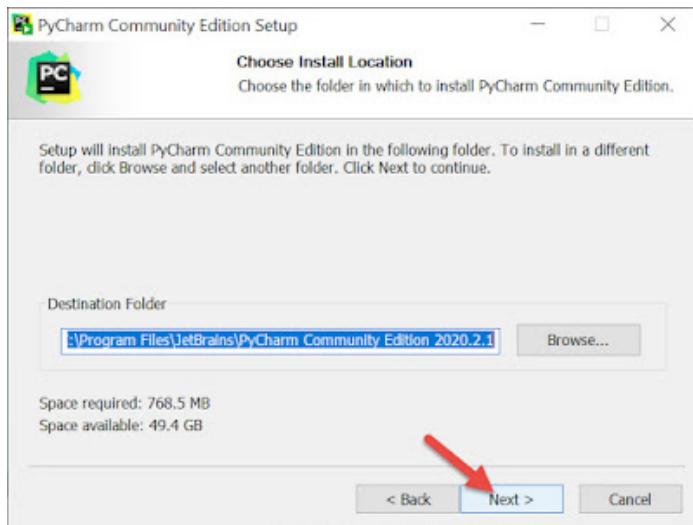
Ikuti langkah-langkah berikut untuk install pycharm di sistem operasi windows:

1. Download Installer Pycharm dari situs resminya yaitu <https://www.jetbrains.com/pycharm/download/>.
2. Jalankan Installer PyCharm yang telah selesai di download, kemudian klik Next untuk melanjutkan installasi seperti pada contoh gambar 9.20.



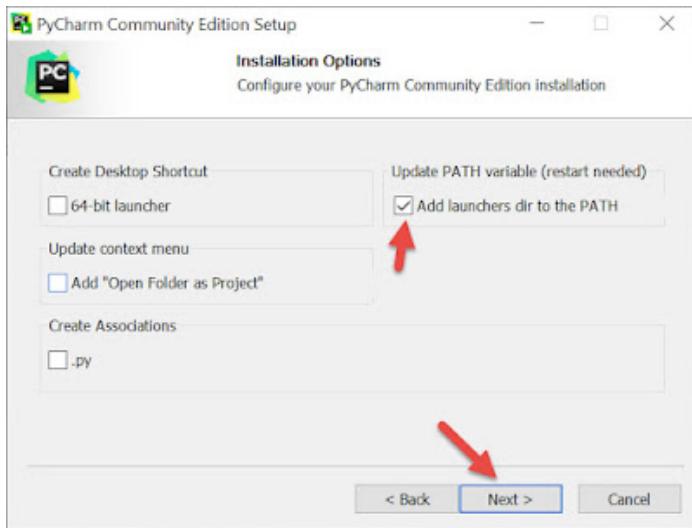
Gambar 9.20 Klik Next untuk Install Pycharm

3. Tentukan destination folder jika diperlukan, atau dibiarkan default kemudian klik Next.



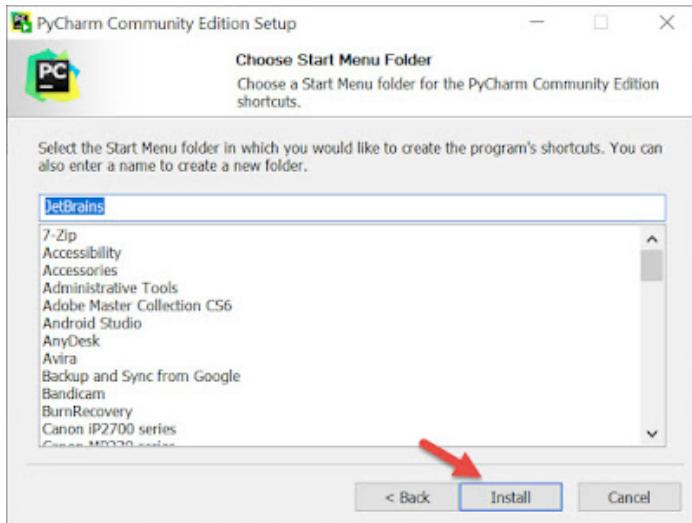
Gambar 9.21 Menentukan lokasi folder dan klik next

4. Checklist pada "Add launchers dir to PATH" kemudian klik Next.



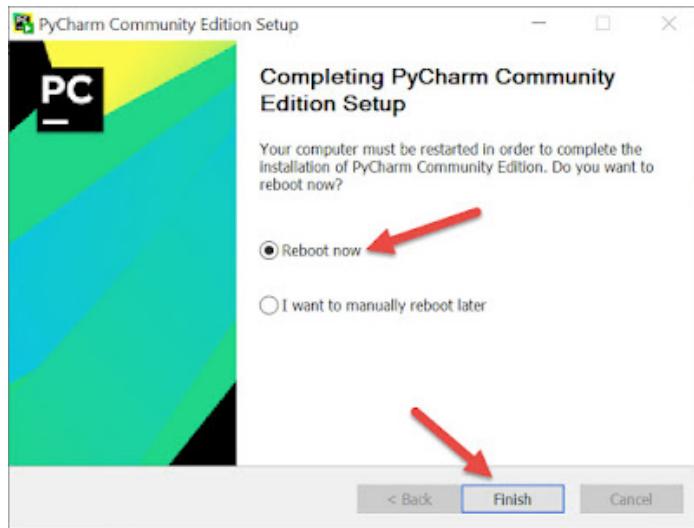
Gambar 9.22 Add launchers dir to PATH

5. Klik Install untuk melanjutkan



Gambar 9.23 Klik Install

6. Tunggu proses instalasi PyCharm sampai selesai.



Gambar 9.24 Reboot Now

- Setelah proses installasi PyCharm selesai, klik Reboot now, lalu klik Finish.

9.3 Membuat Akun Github

Github merupakan software developer untuk menyimpan dan sharing project-project yang bersifat opensource, karena bersifat opensource maka project tersebut dapat dikembangkan oleh programmer lain yang ingin berkontribusi pada project, sangat bagus untuk project yang dikembangkan oleh team maupun individu.

Github memiliki banyak keunggulan diantaranya sebagai tempat penyimpanan project, adanya free hosting, memudahkan developer, mendukung semua bahasa pemrograman dan github juga berguna sebagai tempat penyimpanan project secara gratis.

Berikut tata cara membuat akun github:

- buka browser kemudian kunjungi situs github.
- klik tombol sign up



Gambar 9.25 Sign Up | sumber: <https://omcyber.com/cara-membuat-akun-github/>

3. isi formulir data diri seperti username, email, password, dan verify your account. Pastikan data yang diisikan benar.

The image shows the GitHub sign-up form. It consists of several input fields and a note below them. The fields are labeled 'Username *', 'Email address *', and 'Password *'. Below the password field is a note: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#)'. Under 'Email preferences', there is a checkbox labeled 'Send me occasional product updates, announcements, and offers.' At the bottom, there is a section titled 'Verify your account' with a 'Create account' button. A watermark for 'Omcyber.com' is visible in the bottom right corner of the form area.

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#).

Email preferences

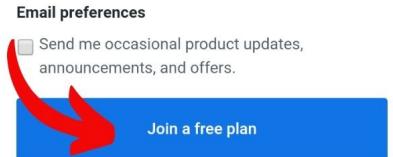
Send me occasional product updates, announcements, and offers.

Verify your account

[Create account](#)

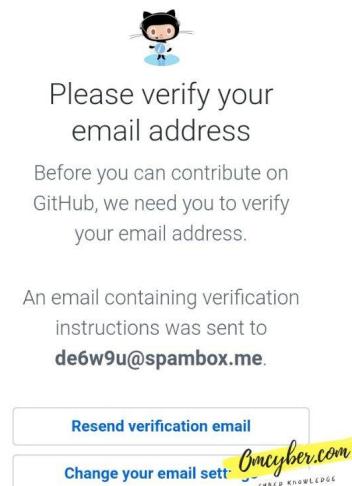
Gambar 9.26 Form Daftar

4. klik tombol create an account untuk membuat akun baru github.
5. silahkan pilih free plan dengan klik tombol join a free plan



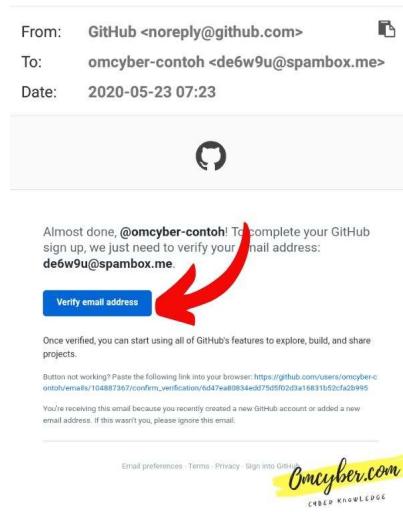
Gambar 9.27 Free Plan

6. verifikasi email agar akun github terverifikasi



Gambar 9.28 Verifikasi Email

7. buka inbox email yang didaftarkan, buka email dari github dan klik verifikasi email address.



Gambar 9.29 Verifikasi Email Address

- setelah berhasil verifikasi email maka akun github telah berhasil dibuat dan ter-verifikasi.



What do you want to do first?

Every developer needs to configure their environment, so let's get your GitHub experience optimized for you.



Gambar 9.30 Verifikasi Email Berhasil

9.4 Download dan Install Git

Sebelum melakukan instalasi git, kita memerlukan file instalasi git terlebih dahulu. File ini bisa teman-teman dapatkan dengan mengunduh file dari situs resmi git, berikut link yang dapat teman-teman kunjungi

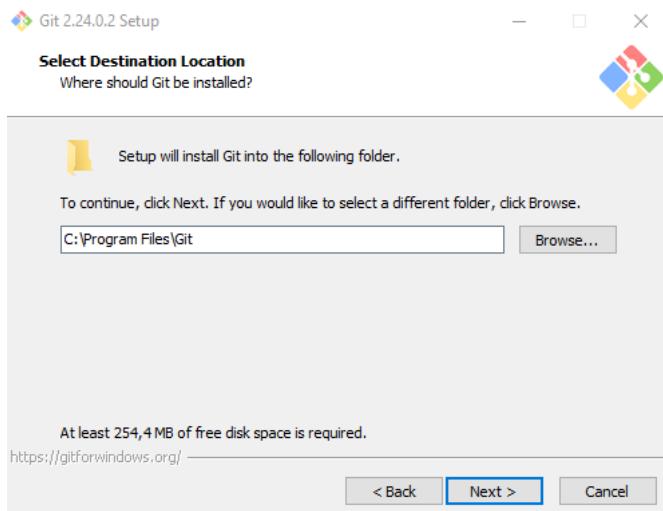
<https://git-scm.com/downloads>. Download file yang sesuai dengan sistem operasi pada komputer teman-teman. Jika komputer teman-teman memiliki sistem operasi Windows 64bit maka teman-teman harus mengunduh file instalasi git untuk windows 64bit. Ikuti tahapan berikut untuk melakukan instalasi git:

1. Buka file instalasi git yang telah di download, kemudian klik next.



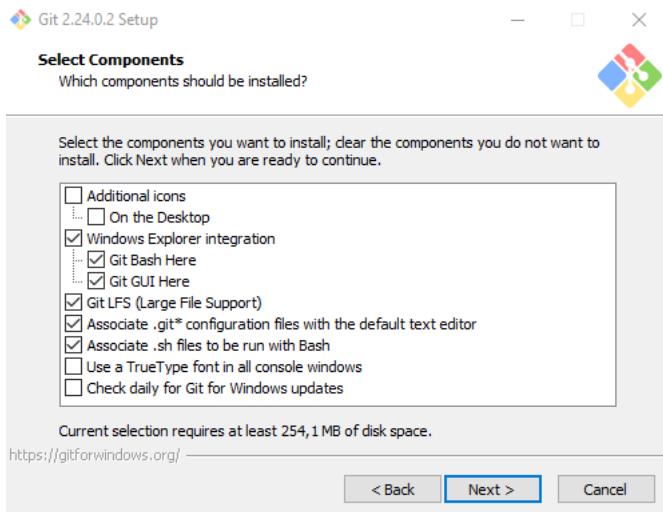
Gambar 9.31 Install Git

2. Pilih lokasi instalasi git, disarankan install pada lokasi seperti gambar



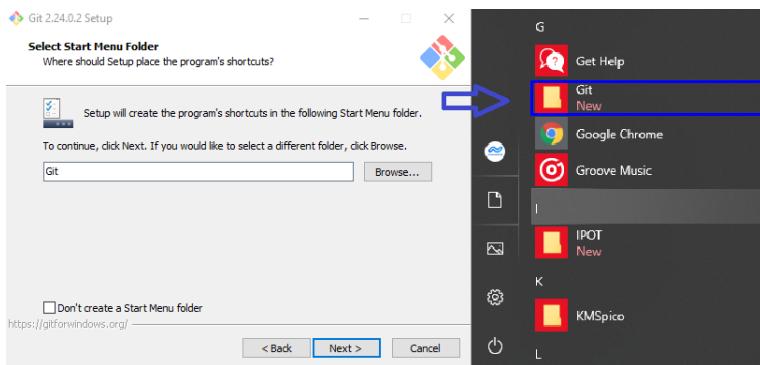
Gambar 9.32 Lokasi Instalasi Git

3. Selanjutnya pilih komponen tambahan untuk install git, sesuaikan dengan kebutuhan teman-teman. Jika sudah maka klik next untuk melanjutkan instalasi.



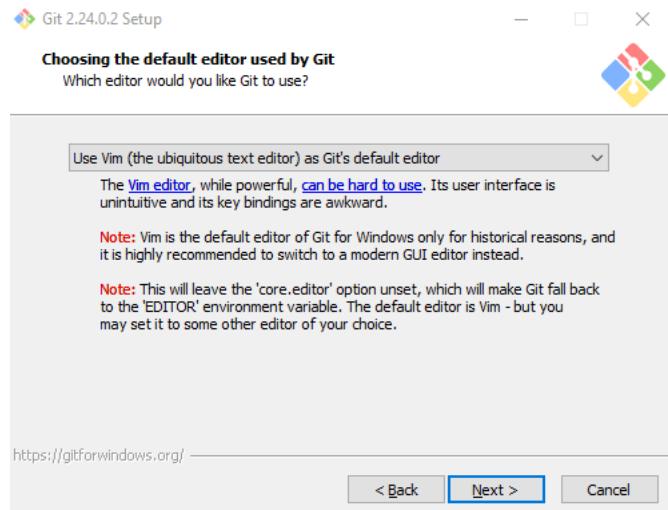
Gambar 9.33 Komponen Tambahan

4. Tentukan nama aplikasi git, untuk memudahkan pencarian aplikasi maka disarankan menggunakan nama Git saja.



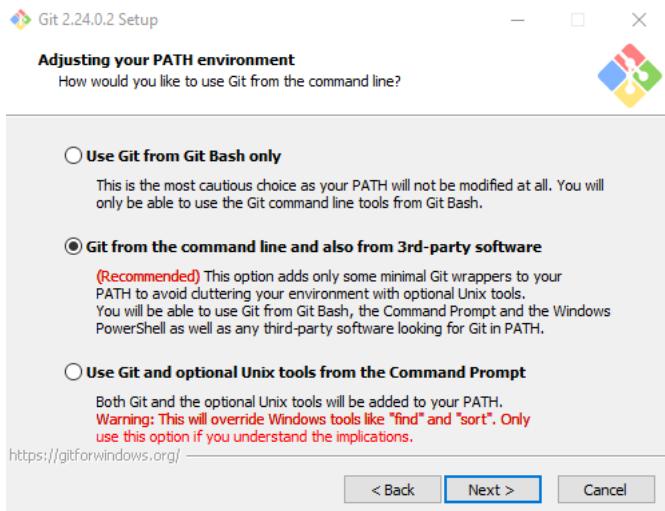
Gambar 9.34 Select Start Menu Folder

5. pilih file editor untuk dikombinasikan dengan git, pada tutorial ini saya menggunakan vim editor lalu pilih next.



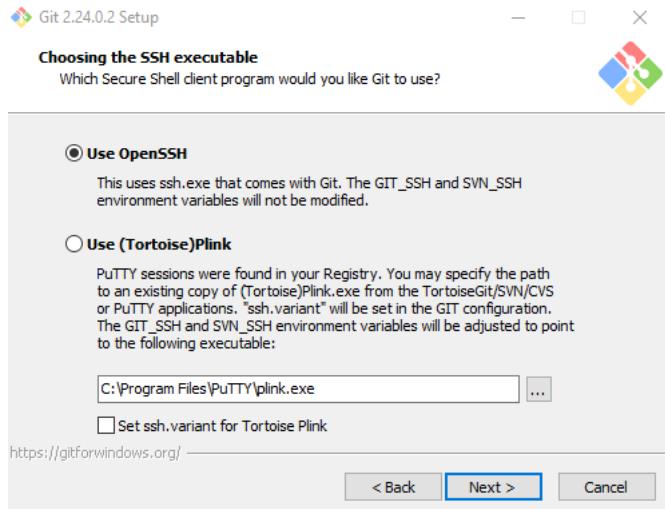
Gambar 9.35 Choose Default Editor

6. selanjutnya kita perlu mengatur path environment, Pilih Git from the command line and also from 3rd-party software agar saat menjalankan perintah Git dapat dikenali di Command Prompt (CMD) pada Windows.



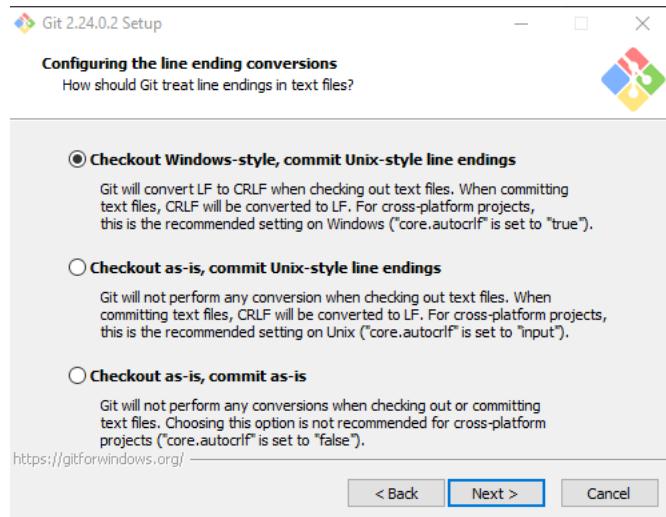
Gambar 9.36 PATH Environment

7. kemudian pilih aplikasi SSH, pada tutorial ini saya memilih Use OpenSSH yang merupakan aplikasi default SSH dari Git lalu klik next.



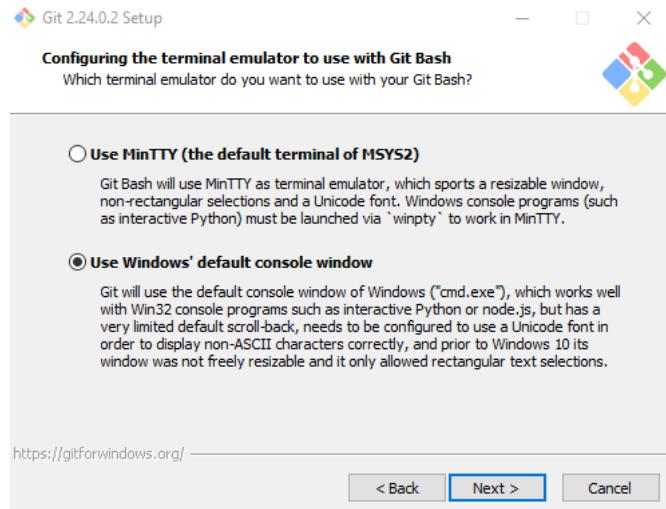
Gambar 9.37 Choose SSH

8. selanjutnya pilih Checkout Windows-style, commit Unix-style line endings dan klik next



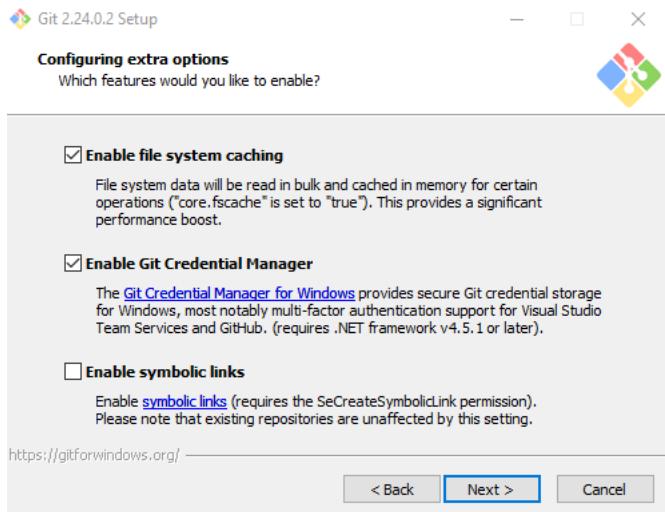
Gambar 9.38 Configuring the line ending

9. pilih Use Windows' default console windows kemudian klik next

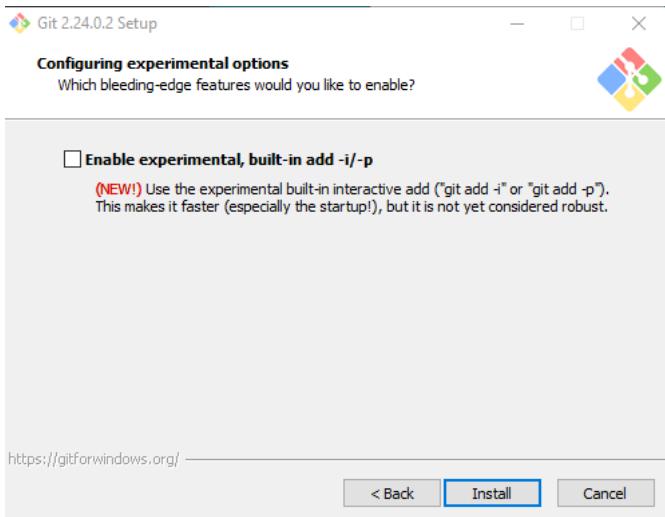


Gambar 9.39 Configuring the terminal emulator

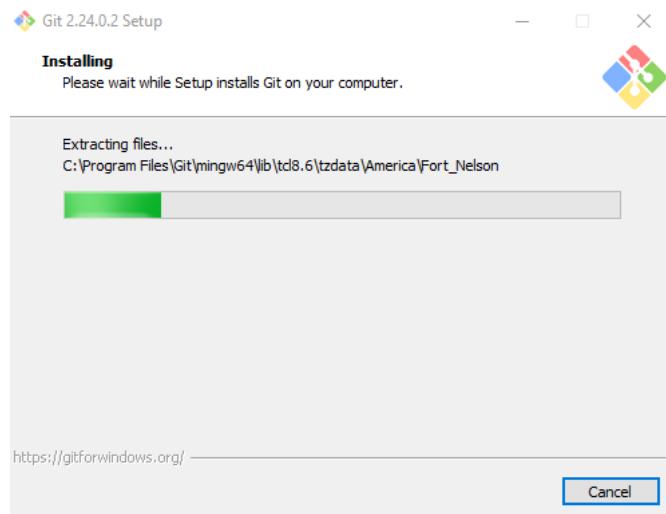
10. pilih opsi ekstra yaitu Enable File System Caching agar Git memiliki fungsi system caching dan Enable Git Credential Manager agar Git bisa dikombinasikan dengan aplikasi lain seperti Visual Studio, Android Studio, dan GitHub. Selanjutnya klik next.

**Gambar 9.40** Configuring extra options

11. setelah menambahkan konfigurasi ekstra maka teman-teman bisa melakukan instalasi git dengan klik install.

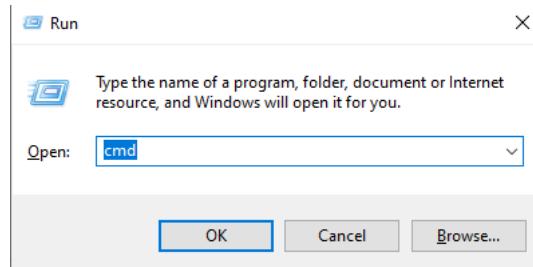
**Gambar 9.41** Install

12. Tunggu hingga proses instalasi selesai.



Gambar 9.42 Proses Install

13. Jika instalasi telah selesai maka teman-teman bisa cek versi git untuk memastikan apakah git telah berhasil terinstal di komputer teman-teman dengan membuka command prompt atau cmd.



Gambar 9.43 Command Prompt (CMD)

14. ketikkan perintah git –version lalu tekan enter untuk melihat versi git yang telah teman-teman install.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\dafit>git --version
git version 2.24.0.windows.2

C:\Users\dafit>
```

A screenshot of a Windows Command Prompt window. The title bar says "C:\WINDOWS\system32\cmd.exe". The command "git --version" is entered and the output "git version 2.24.0.windows.2" is displayed. The command prompt ends with "C:\Users\dafit>".

Gambar 9.44 Versi Git

9.5 Konfigurasi Git

Berikut hal yang perlu teman-teman lakukan apabila telah install git di komputer, teman-teman perlu melakukan konfigurasi email dan username akun git teman-teman. Ikuti tahapan berikut untuk melakukan konfigurasi git:

1. Konfigurasikan email dan username git yang telah teman-teman daftarkan pada github dengan mengetikkan perintah

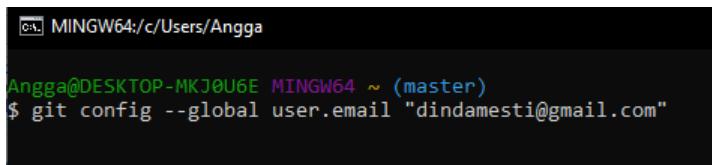
```
git config --global user.name "username anda"  
git config --global user.email "email_anda@gmail.com"
```



The screenshot shows a terminal window titled 'MINGW64:/c/Users/Angga'. The command \$ git config --global user.name "dindamajesty13" is entered and executed, changing the global user name.

```
Angga@DESKTOP-MKJ0U6E MINGW64 ~ (master)  
$ git config --global user.name "dindamajesty13"
```

Gambar 9.45 Konfigurasi Username Git



The screenshot shows a terminal window titled 'MINGW64:/c/Users/Angga'. The command \$ git config --global user.email "dindamesti@gmail.com" is entered and executed, changing the global user email.

```
Angga@DESKTOP-MKJ0U6E MINGW64 ~ (master)  
$ git config --global user.email "dindamesti@gmail.com"
```

Gambar 9.46 Konfigurasi Email Git

2. membuat ssh key dengan mengetikkan perintah

```
ssh-keygen -t rsa -b 4096 -C "email_anda@gmail.com"
```

lalu tekan enter sebanyak tiga kali, pertama untuk membuat direktori penyimpanan ssh, kedua dan ketiga untuk menambahkan passphrase.

```
saras@DESKTOP-DCDKHUR MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "sarascayaya@gmail.com" 1
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/saras/.ssh/id_rsa): 2
Created directory '/c/Users/saras/.ssh'.
Enter passphrase (empty for no passphrase): 3
Enter same passphrase again: 4
Your identification has been saved in /c/Users/saras/.ssh/id_rsa.
Your public key has been saved in /c/Users/saras/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:C68xLrRsWaVX681mt0BVE7tXUzjN0g8Ad9MtM1QCVcc sarascayaya@gmail.com
The key's randomart image is:
+---[RSA 4096]---+
|       .++oooo |
|       . +.+E |
|       o + * |
|       . . . o + |
|       .oS. . . + |
|       . oo... . +o+ |
|       o+.o.o. + ..%+ |
|       * . + . B. = |
|       . .o =. . . |
+---[SHA256]---+
```

Gambar 9.47 Generate SSH key

3. jika ssh key telah berhasil digenerate, ketikkan perintah

```
cd .ssh/
ls
```

perintah ini digunakan untuk berpindah direktori ke direktori ssh dan menampilkan list file yang ada didalam direktori tersebut.

```
c:\ MINGW64:/c/Users/Angga/.ssh

Angga@DESKTOP-MKJ0U6E MINGW64 ~ (master)
$ cd .ssh/
Angga@DESKTOP-MKJ0U6E MINGW64 ~/ssh (master)
$ ls
config  id_rsa  id_rsa.pub  id_rsa_angga  id_rsa_angga.pub  known_hosts  known_hosts.old
```

Gambar 9.48 Direktori SSH

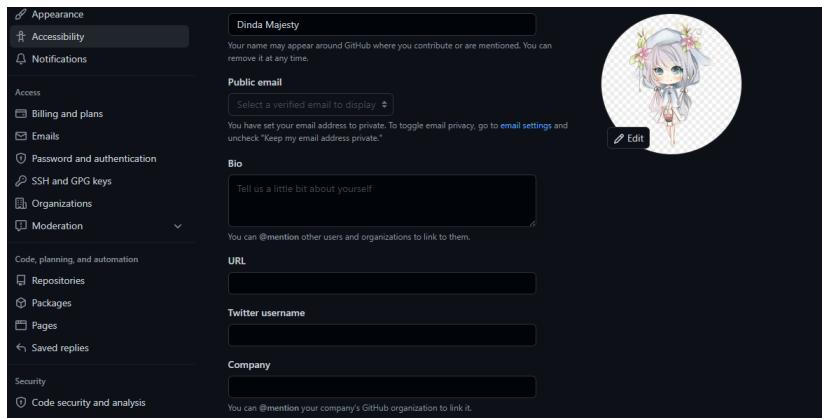
4. ketikkan perintah berikut untuk menampilkan ssh yang telah digenerate, kemudian copy ssh key yang tampil dilayar teman-teman.

```
cat id_rsa.pub
```

```
c:\ MINGW64:/c/Users/Angga/.ssh
Angga@DESKTOP-MKJ0U6E MINGW64 ~/ssh (master)
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQGQCNdL/CJD5z0MCWkWxN4Tz0k2oBy+kNwG9bchXe9lwrPQnb6tFZMLrsyRSz
fT6Hp0Kzq239Ldq8R2RsTpwlSoNkZ+91rrtDjXI+5ACKz9v9z2T5rjNR4tj44aq09Q3kQDq06tTrn+Gs0pbhwy6amknHb
u0qkq8pasIG2ow2d3onX14za/3cxI90mXX7upd3iF78SCVVbJX1THRgT352UdN651F0q0L1cID80MEEM44KKw7rSBa1xUe
KtcYCVeYSDSyDgqywCr31vbCMJa55WC10bkn+4JZS9M49mgZ00CeSoXffFRldpFY4x1+ixB8Sn4Eaq5VMa03W21Tkzs
DuqAztLe6WPNBwRF5cxYggxz3bj+yHakLF/7H9IxVdguxJPX2P7t5FOw1e6ziJgJMB5EGk= dindamesti@gmail.com
```

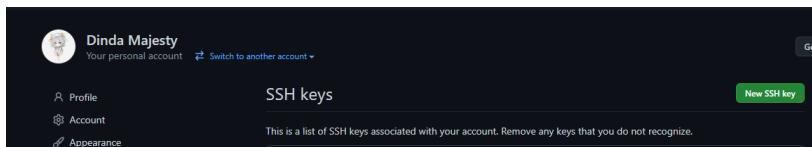
Gambar 9.49 SSH Key

5. pilih menu SSH dan GPG keys



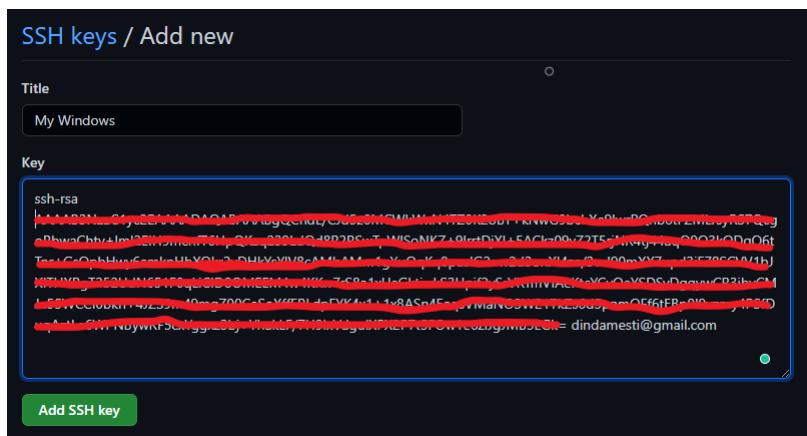
Gambar 9.50 Menu SSH dan GPG keys

6. klik tombol New SSH key



Gambar 9.51 New SSH key

7. isikan title, pada bagian key paste ssh key yang telah di copy sebelumnya dan klik add SSH key



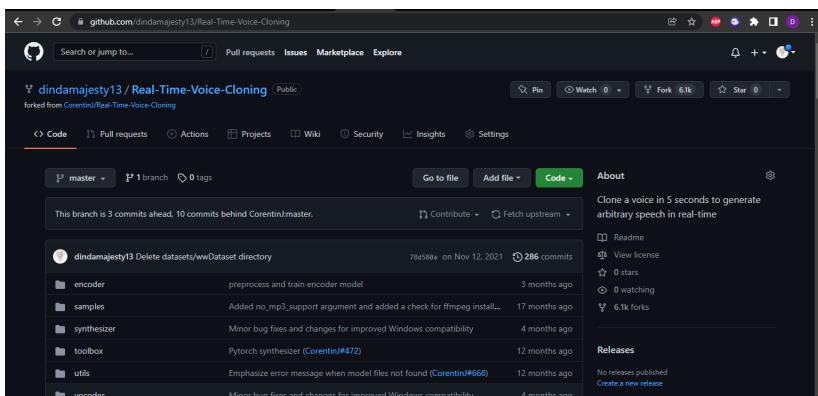
Gambar 9.52 Add SSH key

- sekarang teman-teman telah berhasil menambahkan ssh key pada akun github teman-teman. Selanjutnya teman-teman dapat mengklon Real-Time-Voice-Cloning project.

9.6 Fork dan Clone Voice Cloning Repository

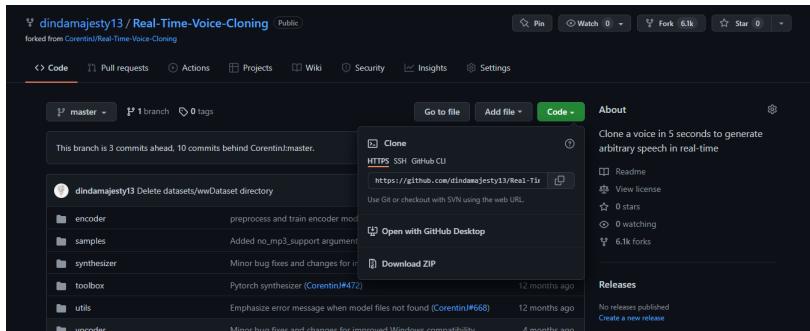
berikut tutorial clone repo voice cloning menggunakan github:

- Kunjungi link berikut
<https://github.com/dindamajesty13/Real-Time-Voice-Cloning>
- Klik fork, maka repo Real-Time-Voice-Cloning akan ada di github anda.



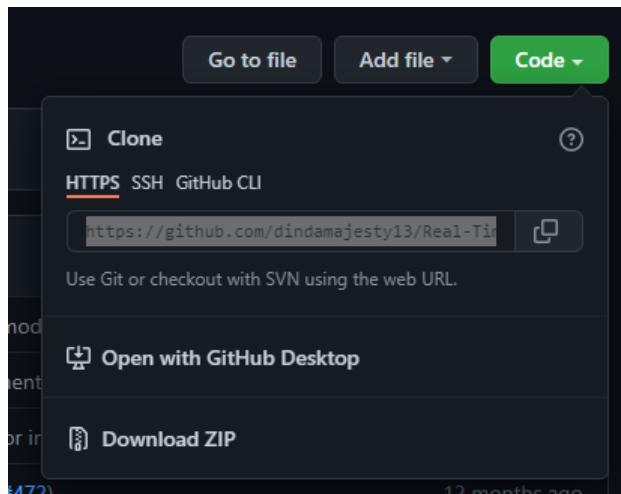
Gambar 9.53 Fork Repository

- buka project Real-Time-Voice-Cloning yang ada direpo anda lalu klik pada bagian code.



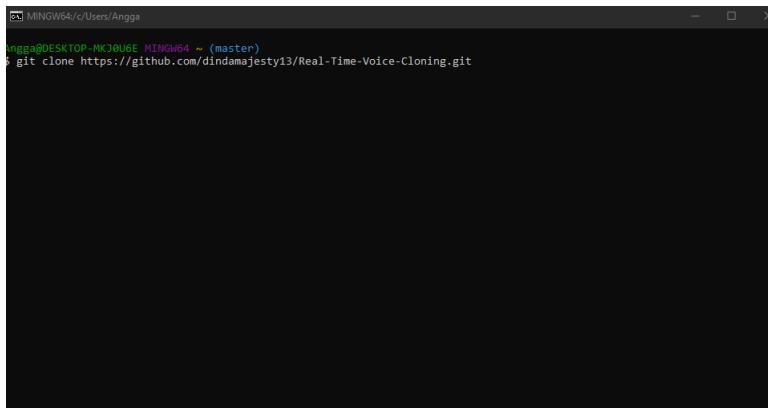
Gambar 9.54 Klik Tombol Code

4. pilih HTTPS dan salin url yang ada dibawah tulisan HTTPS



Gambar 9.55 Copy Url

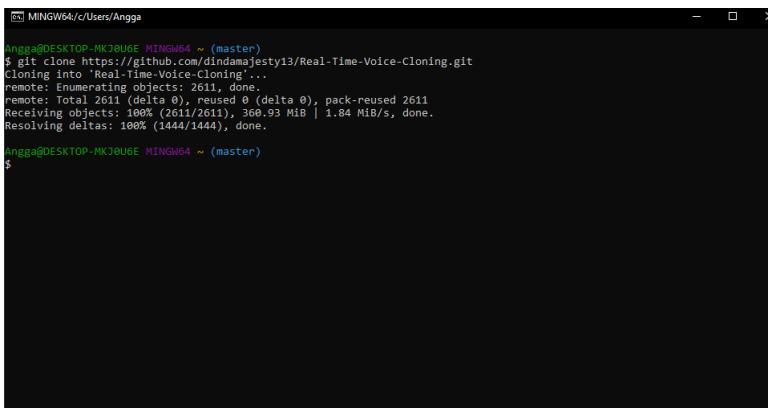
5. buka gitbash kemudian ketikkan git clone lalu paste url yang telah di copy sebelumnya.



```
MINGW64/c/Users/Angga
$ git clone https://github.com/dindamajesty13/Real-Time-Voice-Cloning.git
```

Gambar 9.56 Git Clone Repository

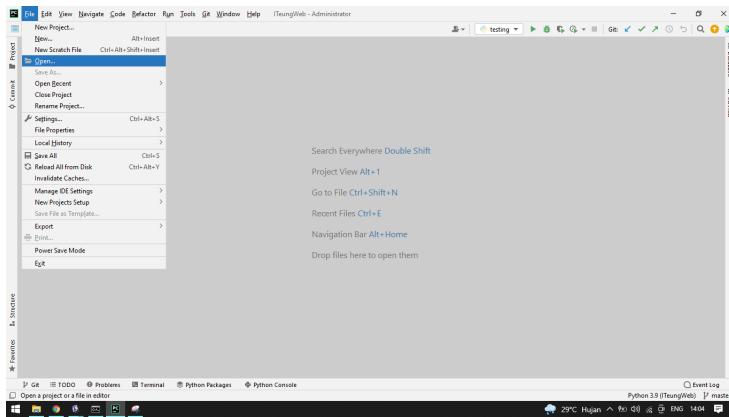
6. Lalu tekan enter pada keyboard, tunggu hingga proses clone selesai.



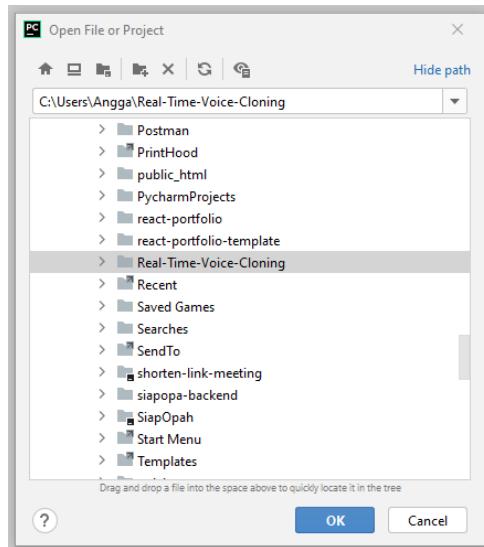
```
MINGW64/c/Users/Angga
$ git clone https://github.com/dindamajesty13/Real-Time-Voice-Cloning.git
Cloning into 'Real-Time-Voice-Cloning'...
remote: Enumerating objects: 2611, done.
remote: Total 2611 (delta 0), reused 0 (delta 0), pack-reused 2611
Receiving objects: 100% (2611/2611), 360.93 MiB | 1.84 MiB/s, done.
Resolving deltas: 100% (1444/1444), done.
$
```

Gambar 9.57 Proses Clone Repository

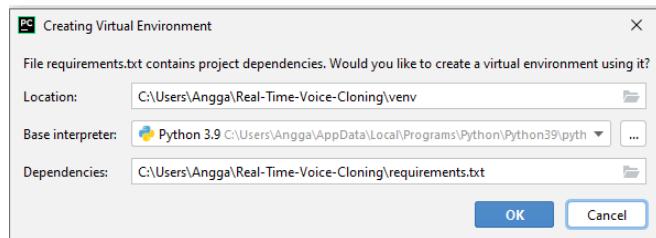
7. jika proses clone telah selesai, buka pycharm.
8. klik menu file, pilih open.

**Gambar 9.58** Pycharm

9. cari folder Real-Time-Voice-Cloning, kemudian klik ok

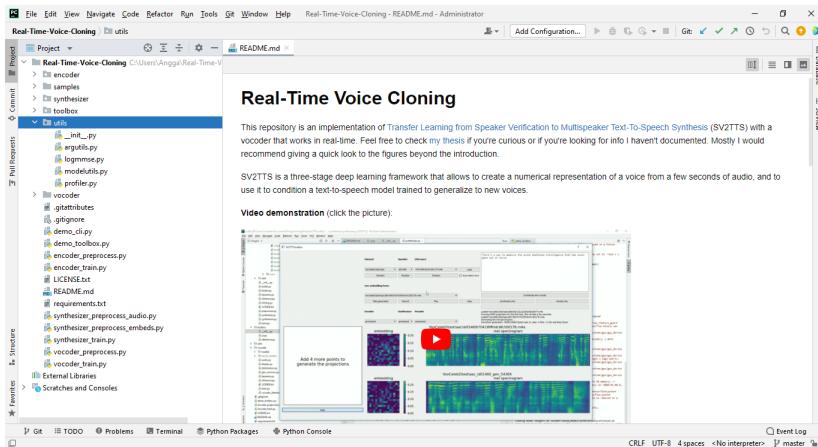
**Gambar 9.59** Lokasi Folder Repotori

10. pycharm akan membuka file project Real-Time-Voice-Cloning, jika terdapat pop up create virtual environment klik ok jika versi python sama dengan 3.7, jika versi python 3.9 seperti gambar 9.60 maka klik cancel dan ikuti tahapan Menambahkan Interpreter ke dalam Voice Cloning Project dibawah.



Gambar 9.60 Pop Up Create Virtual Environment

11. jika berhasil maka anda akan melihat tampilan seperti pada gambar 9.61



Gambar 9.61 Real-Time-Voice-Cloning project

9.7 Menambahkan Interpreter ke dalam Voice Cloning Project

Sebelum menjalankan project menggunakan Pycharm kita membutuhkan interpreter, saya menyarankan untuk menggunakan virtual environment atau conda environment jika anda pengguna anaconda. Dalam pembuatan interpreter penggunaan python 3.7 sangat direkomendasikan karena pada pembuatan voice cloning ini menggunakan tensorflow versi 1.x sehingga teman-teman harus mendownload python versi 3.7, ffmpeg, pytorch, dan requirements. Berikut tata pembuatan virtual environment untuk project voice cloning ini:

9.7.1 Download dan Install Python 3.7

1. download python versi 3.7 pada website resmi python atau kunjungi link berikut <https://www.python.org/downloads/release/python-371/>
2. Pilih file yang sesuai dengan sistem operasi dan processor pada komputer teman-teman.

Files

Version	Operating System	Description	MDS Sum	File Size	GP
Gzipped source tarball	Source release		99f78ecbf766ea449c4d9e7eda19e83	22802018	SIG
XZ compressed source tarball	Source release		0a57e9022c07fad3dadb2ee5f58568edb	16960060	SIG
macOS 64-bit/32-bit installer	macOS	for Mac OS X 10.6 and later	ac6330338b53b9e5dbb1bc2390a21e	34360623	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	b69d52f2e73e1fe3732337eb199a53	27725111	SIG
Windows help file	Windows		b5ca69aa44aa46cdb8cf2b527d699740	8534435	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	74f919be8add749e73d2d91eb6d1da5	6879900	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	4c9fd5b437ad393532e57f15ce832bc	26260496	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	6d665305db7e3d523a0eb252ebd9407	1333960	SIG
Windows x86 embeddable zip file	Windows		aa4189ea480a64a3ea87e72e09fc097	6377805	SIG
Windows x86 executable installer	Windows		da24511f28e4cc133c53f0638459993c	25537464	SIG
Windows x86 web-based installer	Windows		20b163041935862876433708819c97db	1297224	SIG

Gambar 9.62 Download Python 3.7

3. Centang add python 3.7 to PATH agar python yang diinstal ditambahkan ke environment variabel komputer teman-teman, kemudian klik install now.



Gambar 9.63 Add Python to PATH

4. Jika instalasi telah selesai maka teman-teman akan melihat tampilan seperti gambar 9.64



Gambar 9.64 Success Install Python 3.7

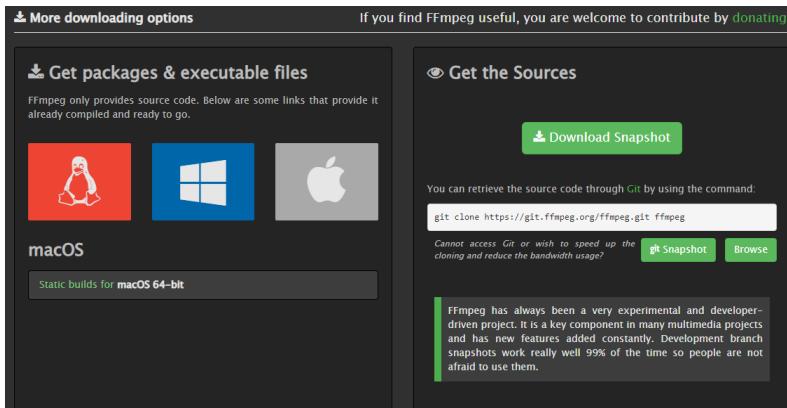
5. Untuk memastikan apakah python 3.7 telah benar-benar berhasil terinstal teman-teman dapat melakukan pengecekan melalui command prompt (CMD) dengan mengetikkan perintah python kemudian tekan enter.

A screenshot of a black command prompt window. The text inside shows the path "C:\Users\ACER>python", followed by the Python version information: "Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32", and the prompt ">>>".

Gambar 9.65 Check Python Version

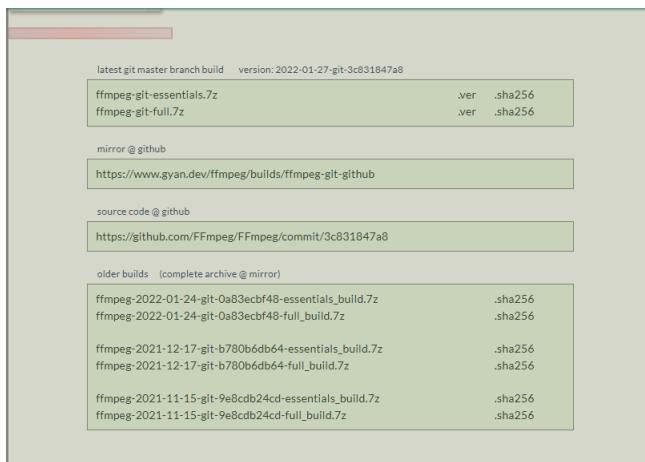
9.7.2 Download dan Install FFmpeg

1. Setelah menginstal python 3.7, kita harus menginstal ffmpeg, kunjungi link berikut untuk mendapatkan file instalasi ffmpeg <https://ffmpeg.org/download.html>. Sesuaikan file download dengan sistem operasi yang teman-teman gunakan. Pada tutorial ini saya menggunakan sistem operasi windows.



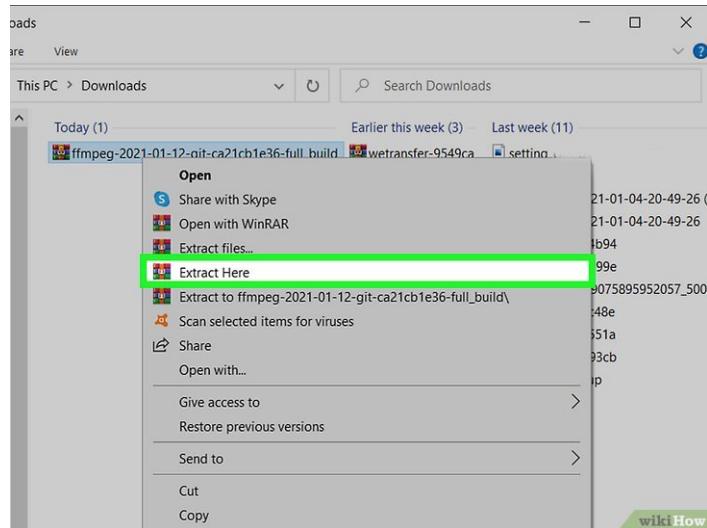
Gambar 9.66 Download FFmpeg

- Pilih Windows builds from gyan.dev. Kemudian teman-teman akan diarahkan ke halaman seperti pada gambar 9.67



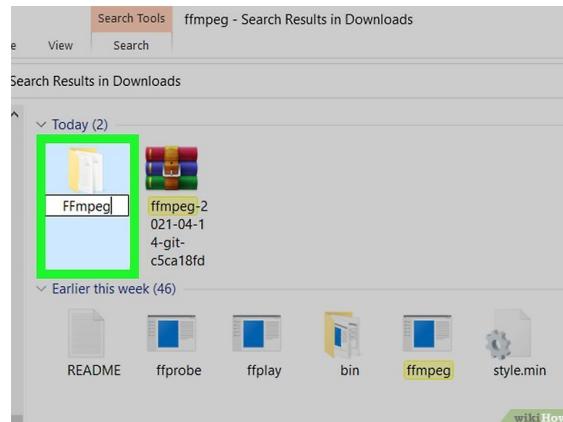
Gambar 9.67 Download FFmpeg for Windows

- Download file ffmpeg-git-full.7z kemudian extract file instalasi FFmpeg yang telah didownload.



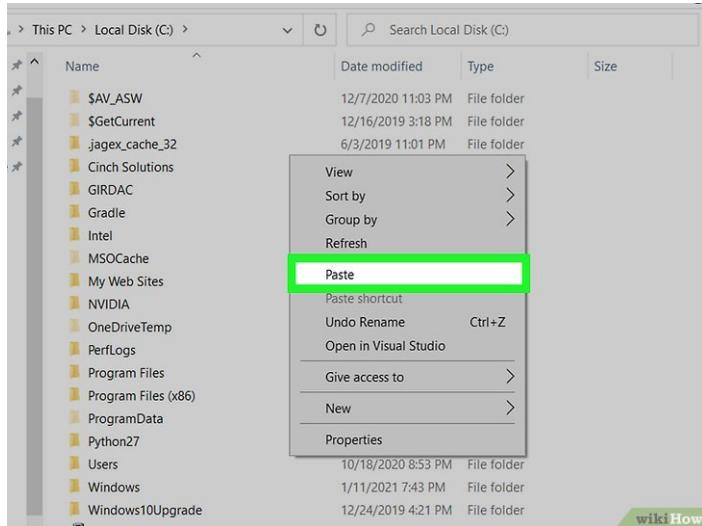
Gambar 9.68 Extract File Instalasi

4. Ganti nama direktori menjadi FFmpeg

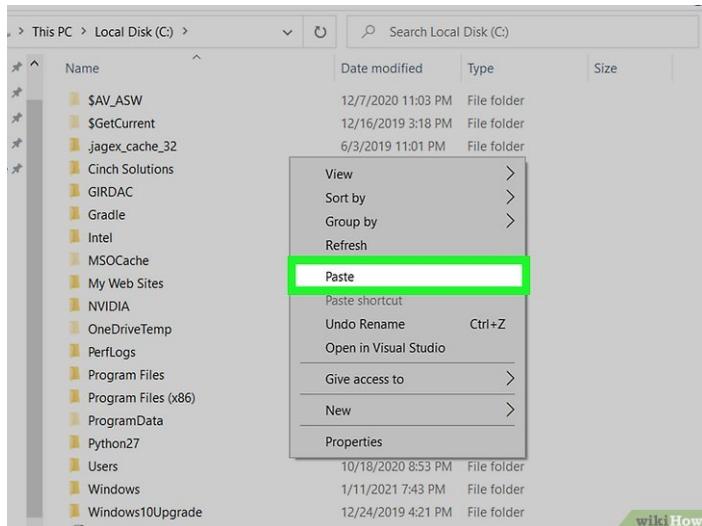


Gambar 9.69 Ganti Nama Folder

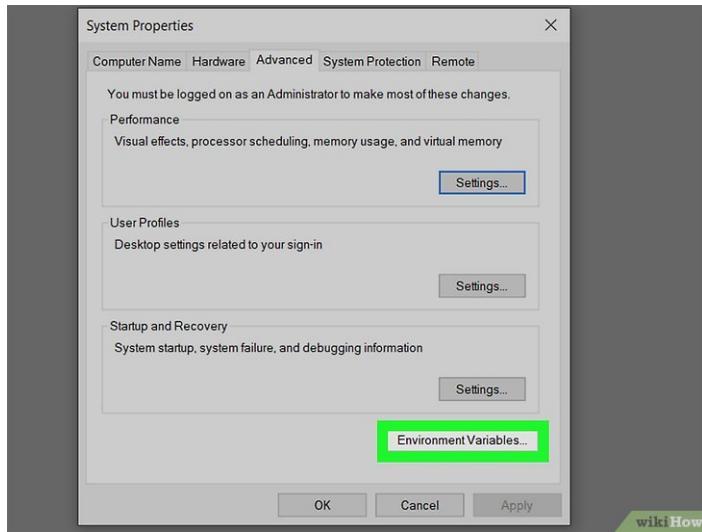
5. Copy folder dan Paste folder ke Local Disk C

**Gambar 9.70** Move Folder

- Klik start pada komputer anda lalu ketikkan environment variable. Klik edit environment variables for your account

**Gambar 9.71** Move Folder

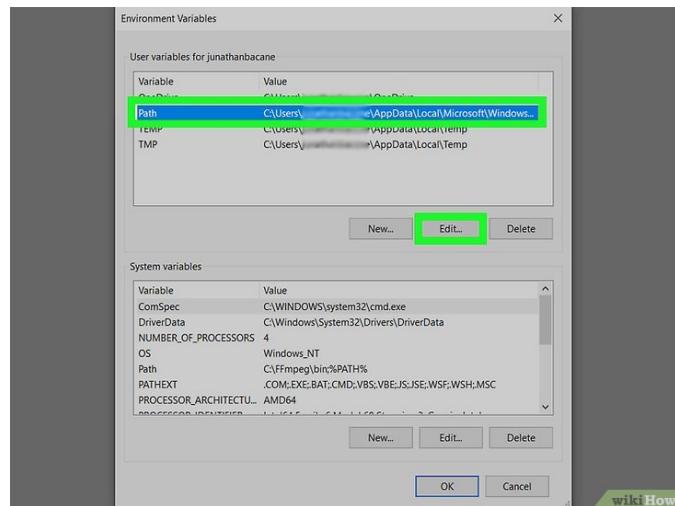
- Klik menu environment variables



Gambar 9.72 Environment Variables

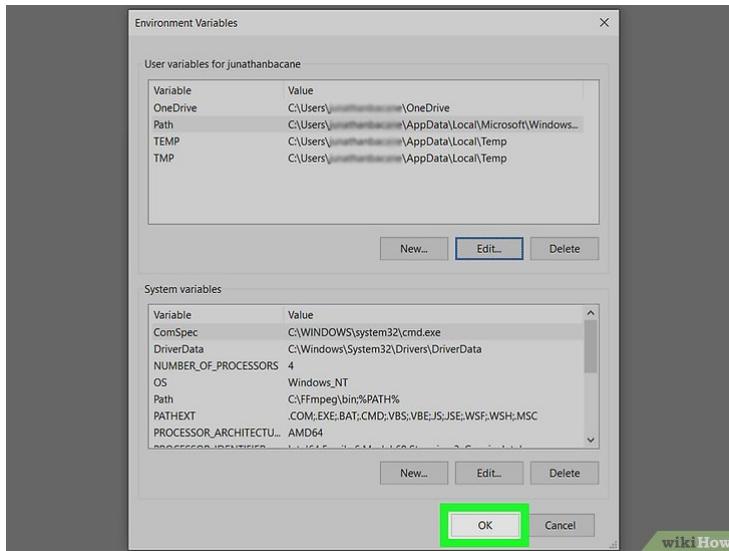
- Pada bagian PATH klik edit maka pop up window edit environment variable seperti pada gambar 9.73 akan muncul. Ketikkan lokasi folder bin FFmpeg lalu klik OK.

C:\ffmpeg\bin



Gambar 9.73 Add FFmpeg to PATH

- Klik OK untuk menyimpan perubahan pada environment variables

**Gambar 9.74** Save Changes

- Untuk melakukan pengecekan apakah FFmpeg telah berhasil di install pada komputer, teman-teman bisa melakukan pengecekan dengan mengetikkan `ffmpeg -version` pada command prompt.

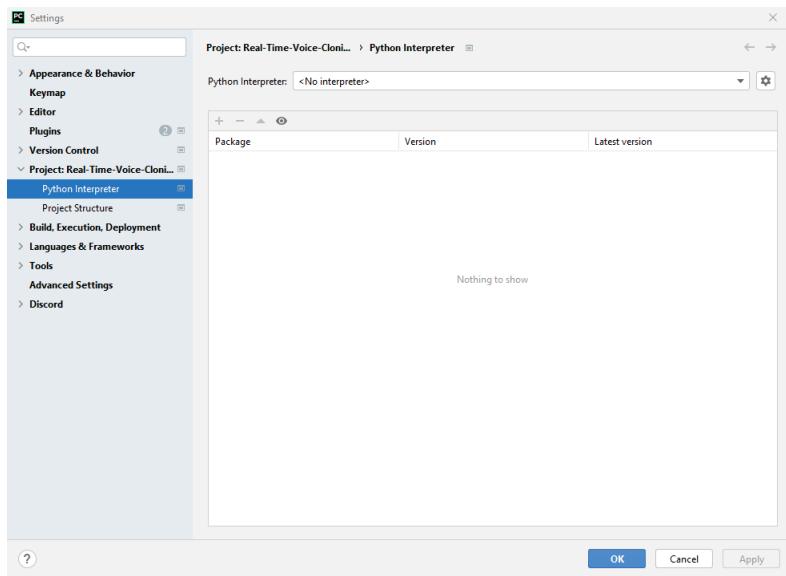
```
C:\Windows\system32>ffmpeg -version
ffmpeg version 2021-09-08-git-5e7e2e5031-full_build-www.gyan.dev Copyright (c) 2000-2021
built with gcc 10.3.0 (Rev5, Built by MSYS2 project)
configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --dis
fig --enable-iconv --enable-gnutls --enable-libxml2 --enable-gmp --enable-lzma --enable-
-librist --enable-libsrt --enable-libssh --enable-libzmq --enable-avisynth --enable-libb
sdl2 --enable-libdav1d --enable-libzvbi --enable-libravie --enable-libsvtav1 --enable-li
libx265 --enable-libxvid --enable-libaom --enable-libopenjpeg --enable-libvpx --enable-l
ibfreetype --enable-libfribidi --enable-libvidstab --enable-libvmaf --enable-libzimg --e
nable-cuvid --enable-ffnvcodec --enable-nvdec --enable-nvenc --enable-d3d11va --enable-d
bglslang --enable-vulkan --enable-opengl --enable-libcdio --enable-libgme --enable-libmo
le-libopencore-amrwb --enable-libmp3lame --enable-libshine --enable-libtheora --enable-l
c --enable-libilbc --enable-libgsm --enable-libopencore-amrnb --enable-libopus --enable-
able-ladspa --enable-libbs2b --enable-libflite --enable-libmysofa --enable-librubberband
aprint
libavutil      57. 4.101 / 57. 4.101
libavcodec     59. 7.102 / 59. 7.102
libavformat    59. 5.100 / 59. 5.100
libavdevice    59. 0.101 / 59. 0.101
libavfilter     8. 7.101 /  8. 7.101
libswscale      6. 1.100 /  6. 1.100
libswresample   4. 0.100 /  4. 0.100
libpostproc    56. 0.100 / 56. 0.100
C:\Windows\system32>
```

Gambar 9.75 Check FFmpeg Version

9.7.3 Tutorial Membuat Virtual Environment

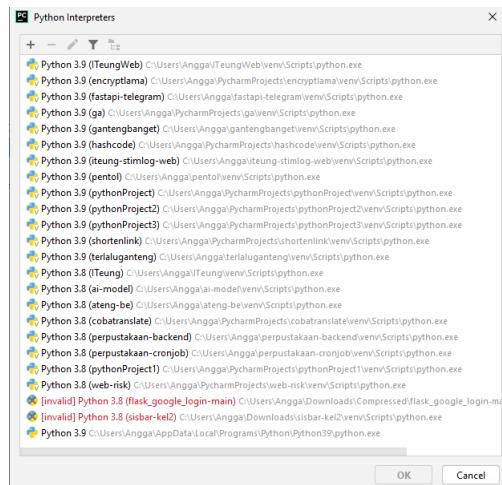
Berikut tutorial membuat virtual environment menggunakan pycharm dan python 3.7:

- Buka settings pycharm dengan mengklik menu file lalu pilih settings. Pada menu Project Real-Time-Voice-Cloning pilih menu python interpreter.



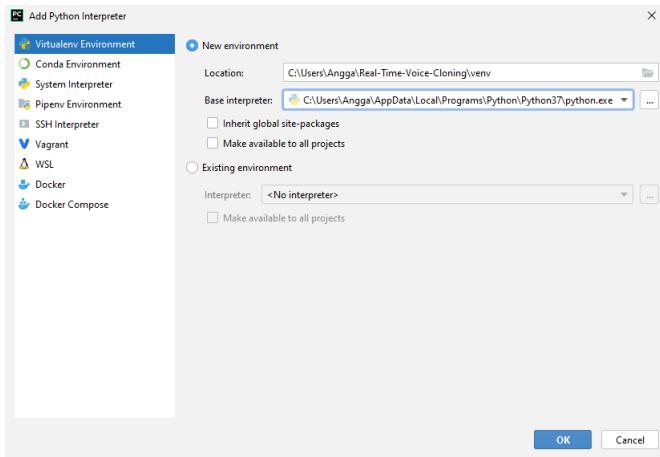
Gambar 9.76 Menu Settings Python Interpreter

- Klik No interpreter lalu pilih show all maka akan muncul python interpreter seperti pada gambar 9.77. Klik tanda tambah untuk membuat interpreter baru.



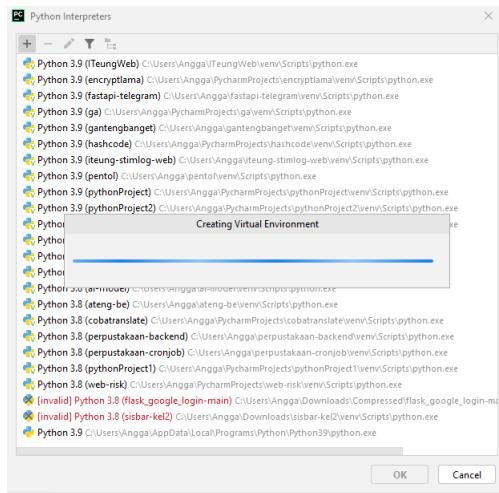
Gambar 9.77 Python Interpreter

3. Pilih Virtual Environment, lalu pilih New Environment, isikan lokasi tempat environment disimpan dan isikan file exe python 3.7 pada base interpreter. Lalu klik OK.



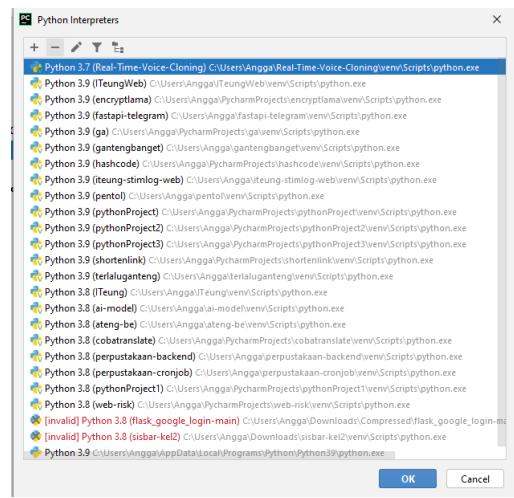
Gambar 9.78 Add Python Interpreter

4. Tunggu hingga proses creating virtual environment selesai.



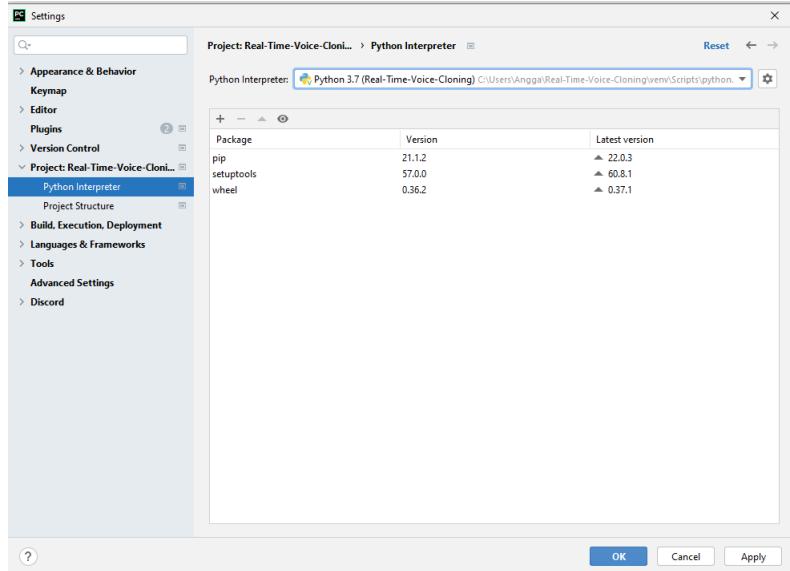
Gambar 9.79 Creating Virtual Environment

5. Pilih virtual environment yang telah kita buat lalu klik OK.



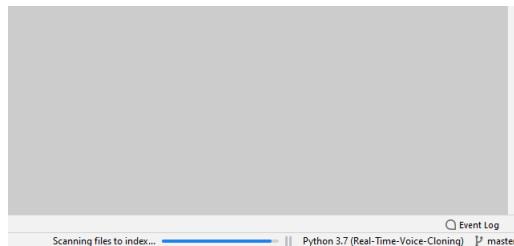
Gambar 9.80 Pilih Virtual Environment

6. Berikut tampilan virtual environment yang telah kita buat menggunakan python 3.7.



Gambar 9.81 Apply Virtual Environment

7. Pada kanan bawah pycharm teman-teman akan melihat progress bar yang menandakan pycharm sedang melakukan inisialisasi terhadap virtual environment teman-teman. Jalankan program apabila proses tersebut telah selesai.

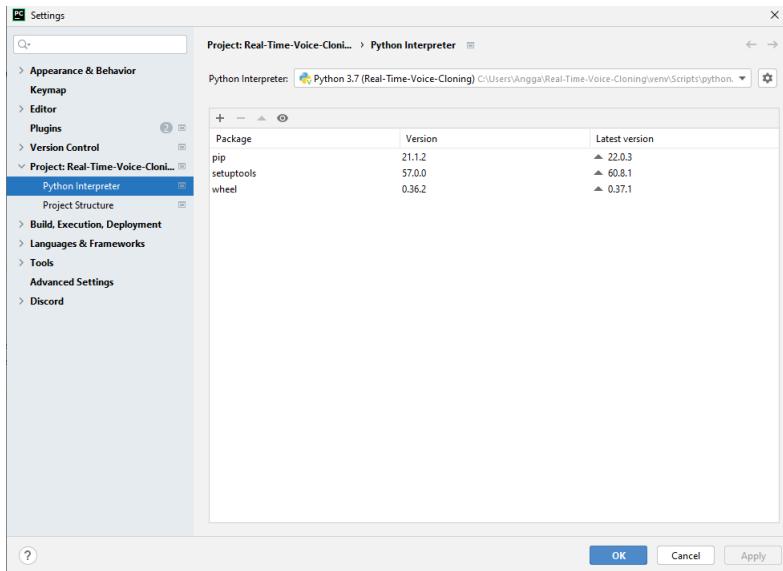


Gambar 9.82 Inisialisasi Virtual Environment

9.7.4 Tutorial Membuat Conda Environment

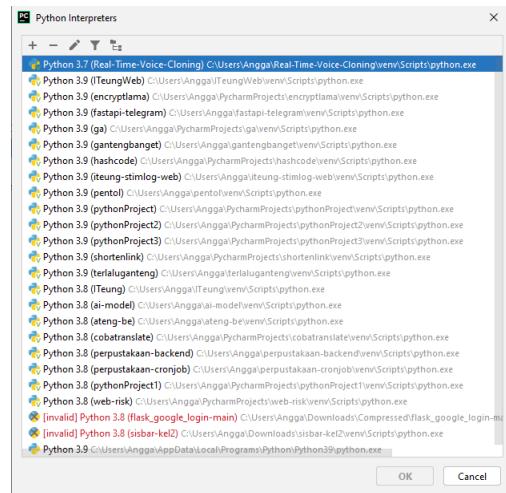
Berikut tutorial membuat conda environment menggunakan pycharm dan python 3.7:

1. Buka settings pycharm dengan mengklik menu file lalu pilih settings. Pada menu Project Real-Time-Voice-Cloning pilih menu python interpreter.



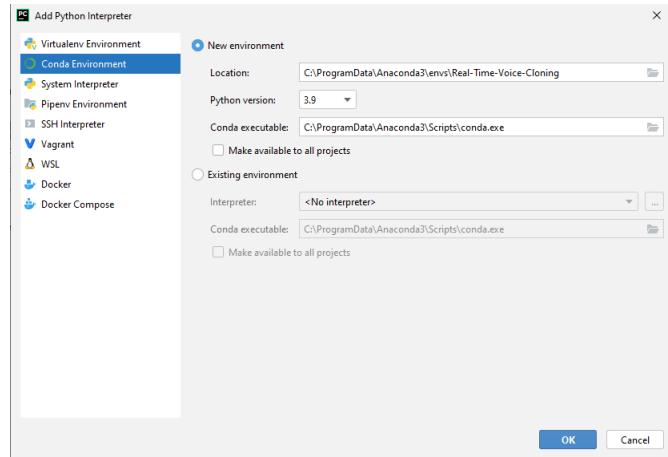
Gambar 9.83 Menu Settings Project

2. Klik No interpreter lalu pilih show all maka akan muncul python interpreter seperti pada gambar 9.84. Klik tanda tambah untuk membuat interpreter baru.



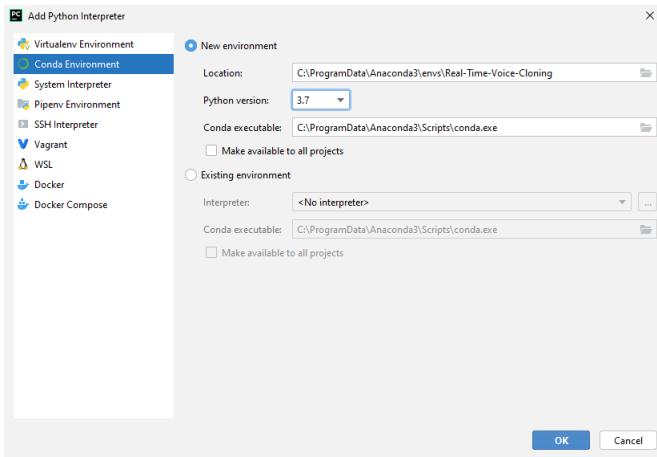
Gambar 9.84 Show All Interpreter

3. Pilih Conda Environment, lalu pilih New Environment.



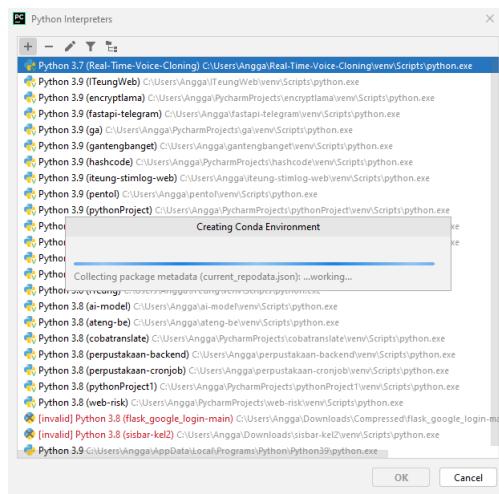
Gambar 9.85 Conda Environment

4. Isikan lokasi tempat conda environment disimpan, pilih python version 3.7 dan isikan lokasi file conda.exe pada conda executable. Lalu klik OK.



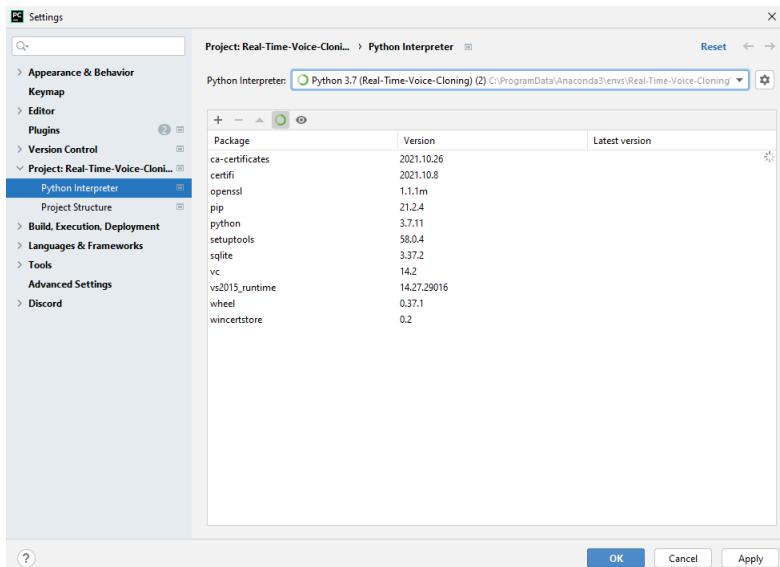
Gambar 9.86 Create New Conda Environment

5. Tunggu hingga proses pembuatan conda environment selesai.



Gambar 9.87 Creating Conda Environment

6. Berikut tampilan conda environment yang telah kita buat, klik Apply lalu klik OK.



Gambar 9.88 Apply Conda Environment

7. Tunggu hingga proses inisialisasi conda environment selesai, jika sudah teman-teman bisa menginstalkan requirements yang teman-teman gunakan dan menjalankan program yang ingin dijalankan.



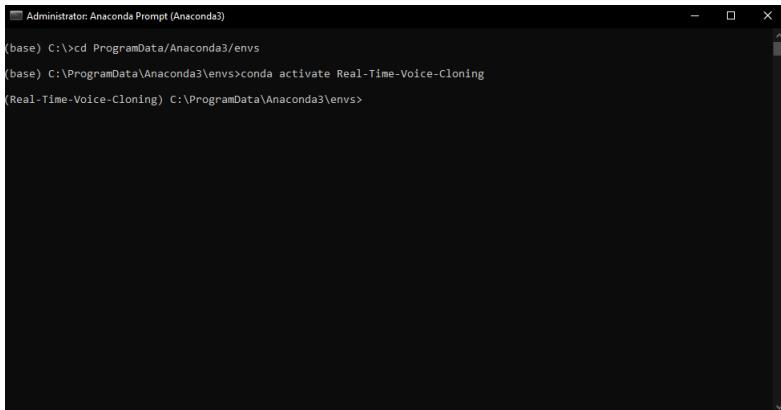
Gambar 9.89 Progress Bar Inisialisasi Conda Environment

9.7.5 Install Requirements

Perhatikan struktur folder project voice cloning teman-teman maka akan terlihat file dengan nama requirements.txt, file tersebut berisikan library-library yang digunakan dalam project ini. Library tersebut harus diinstall terlebih dahulu agar project dapat dijalankan dengan baik. Berikut tata cara instalasi requirements:

1. Jika teman-teman menggunakan conda environment maka buka anaconda prompt dan ketikkan perintah berikut untuk pindah directory dan mengaktifkan conda environment:

```
1 cd ProgramData / Anaconda3 / envs
2 conda activate your_environment
```

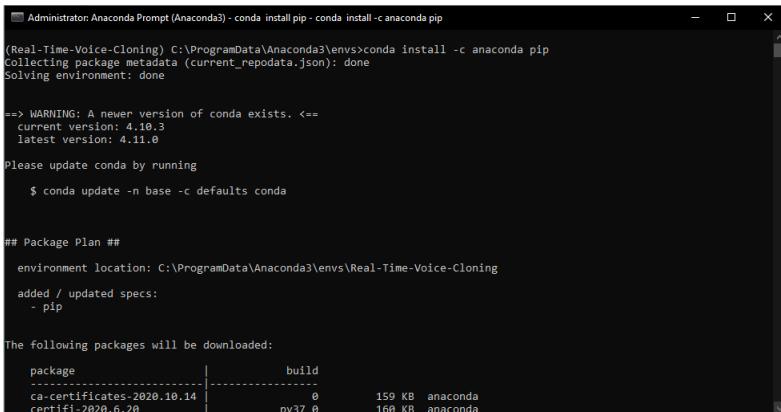


```
Administrator: Anaconda Prompt (Anaconda3)
(base) C:\>cd ProgramData\Anaconda3\envs
(base) C:\ProgramData\Anaconda3\envs>conda activate Real-Time-Voice-Cloning
(Real-Time-Voice-Cloning) C:\ProgramData\Anaconda3\envs>
```

Gambar 9.90 Activate Conda Environment

2. Ketikkan perintah berikut untuk menginstal pip, tunggu hingga proses instalasi selesai.

```
| conda install -c anaconda pip
```



```
Administrator: Anaconda Prompt (Anaconda3) - conda install pip - conda install -c anaconda pip
(Real-Time-Voice-Cloning) C:\ProgramData\Anaconda3\envs>conda install -c anaconda pip
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.10.3
  latest version: 4.11.0
Please update conda by running
  $ conda update -n base -c defaults conda

## Package Plan ##
environment location: C:\ProgramData\Anaconda3\envs\Real-Time-Voice-Cloning
added / updated specs:
- pip

The following packages will be downloaded:
  package          build
  -----          -----
ca-certificates-2020.10.14 |          0      159 KB  anaconda
certifi-2020.6.20    | py37 0      160 KB  anaconda
```

Gambar 9.91 Install Pip using Anaconda Prompt

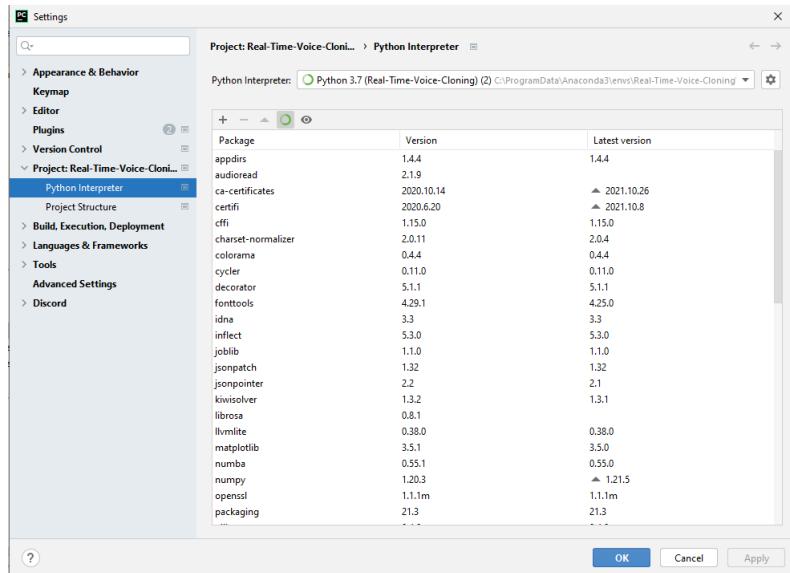
3. Ketikkan perintah berikut untuk menginstal requirements, tunggu hingga proses instalasi selesai.

```
| pip install -r C:\Users\Angga\Real-Time-Voice-Cloning\
               requirements.txt
```

```
(Real-Time-Voice-Cloning) C:\ProgramData\Anaconda3\envs>pip install -r C:\Users\Angga\Real-Time-Voice-Cloning\requirements.txt
Collecting inflect==5.3.0
  Downloading inflect-5.3.0-py3-none-any.whl (32 kB)
Collecting librosa==0.8.1-py3-none-any.whl (203 kB)
Using cached matplotlib==3.5.1
Collecting numpy==1.20.3
  Downloading numpy-1.20.3-cp37-cp37m-win_amd64.whl (7.2 MB)
Collecting Pillow==8.4.0
  Downloading Pillow-8.4.0-cp37-cp37m-win_amd64.whl (3.2 MB)
Collecting PyQt5==5.15.6
  Using cached PyQt5-5.15.6-cp36-abi3-win_amd64.whl (6.7 MB)
Collecting scikit-learn==0.22
  Using cached scikit_learn-0.22.1-cp37-cp37m-win_amd64.whl (7.1 MB)
Collecting scipy==1.7.3
  Using cached scipy-1.7.3-cp37-cp37m-win_amd64.whl (34.1 MB)
Collecting sounddevice==0.4.3
  Downloading sounddevice-0.4.3-py3-none-win_amd64.whl (195 kB)
Collecting SoundFile==0.10.3.post1
  Using cached SoundFile-0.10.3.post1-py2.py3.cp26.cp27.cp32.cp33.cp34.cp35.cp36.pp27.pp32.pp33-none-win_amd64.whl (689 kB)
Collecting tdmr==4.62.3
  Using cached tdmr-4.62.3-py2.py3-none-any.whl (76 kB)
Collecting umap-learn==0.5.2
```

Gambar 9.92 Install Requirements using Anaconda Prompt

- Pastikan requirements telah terinstall, buka python interpreter untuk melakukan pengecekan.



Gambar 9.93 Sukses Install Requirements pada Conda Environment

- Jika teman-teman menggunakan virtual environment maka buka command prompt dan ketikkan perintah berikut untuk pindah directory dan mengaktifkan virtual environment:

```
1 cd your-project
2 .\venv\Scripts\activate
```

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Angga>cd Real-Time-Voice-Cloning
C:\Users\Angga\Real-Time-Voice-Cloning>.\venv\Scripts\activate
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>
```

Gambar 9.94 Activate Virtual Environment with Command Prompt

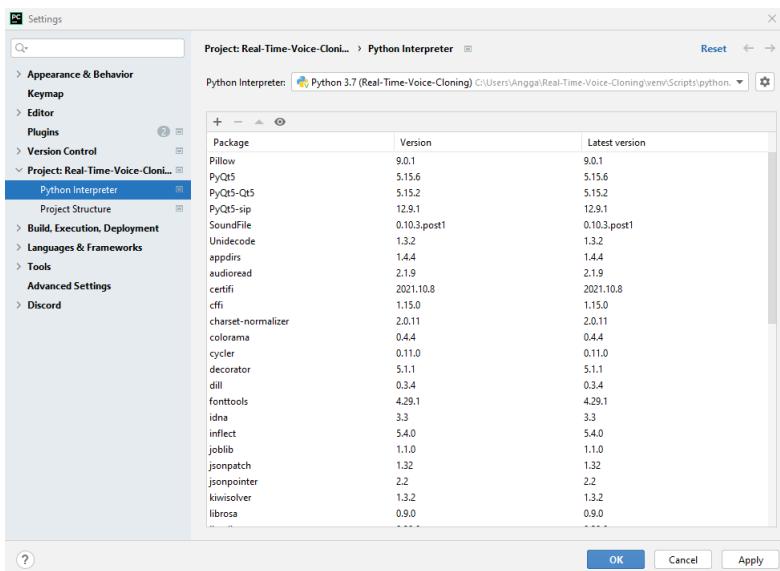
6. Ketikkan perintah berikut untuk menginstal requirements, tunggu hingga proses instalasi selesai.

```
1 pip install -r requirements.txt
```

```
Administrator: Command Prompt - pip install -r requirements.txt
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>pip install -r requirements.txt
Ignoring numpy: markers 'platform_system != "Windows"' don't match your environment
Ignoring webkitvad: markers 'platform_system != "Windows"' don't match your environment
Collecting umap-learn==0.3.10
  Downloading umap-learn-0.3.10.tar.gz (40 kB)
|██████████| 40 kB 2.5 MB/s
Collecting visdom
  Downloading visdom-0.1.8.9.tar.gz (676 kB)
|██████████| 676 kB 6.8 MB/s
Collecting librosa
  Downloading librosa-0.9.0-py3-none-any.whl (211 kB)
|██████████| 211 kB 6.4 MB/s
Collecting matplotlib>=3.3.0
  Downloading matplotlib-3.5.1-cp37-cp37m-win_amd64.whl (7.2 MB)
|██████████| 7.2 MB 1.7 MB/s
Collecting numpy==1.19.3
  Downloading numpy-1.19.3-cp37-cp37m-win_amd64.whl (13.2 MB)
|██████████| 13.2 MB 2.2 MB/s
Collecting scipy==1.0.0
  Downloading scipy-1.7.3-cp37-cp37m-win_amd64.whl (34.1 MB)
|██████████| 34.1 MB 2.2 MB/s
Collecting tqdm
  Downloading tqdm-4.62.3-py3-none-any.whl (76 kB)
|██████████| 76 kB 1.2 MB/s
Collecting sounddevice
  Downloading sounddevice-0.4.4-py3-none-win_amd64.whl (195 kB)
|██████████| 195 kB 2.2 MB/s
Collecting Soundfile
  Using cached Soundfile-0.10.3.post1-py2.py3.cp26.cp27.cp32.cp33.cp34.cp35.cp36.pp27.pp32.pp33-none-win_amd64.whl (689 kB)
```

Gambar 9.95 Install Requirements using Command Prompt

7. Pastikan semua library yang dibutuhkan telah terinstall, buka Settings lalu ke menu python interpreter maka teman-teman akan melihat library yang telah berhasil diinstall pada interpreter teman-teman seperti pada gambar 9.96



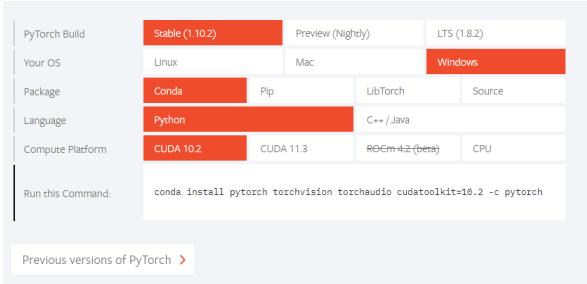
Gambar 9.96 Berhasil Install Requirement pada Virtual Environment

9.7.6 Download and install Pytorch dengan Anaconda

Pytorch merupakan library open source yang digunakan dalam pengembangan dan pelatihan neural network. Dalam project ini, apabila komputer teman-teman memiliki GPU yang mendukung maka saya sarankan untuk menggunakan GPU dalam proses training. Tidak masalah jika tidak memiliki GPU, teman-teman dapat melakukan proses training menggunakan CPU. Instalasi Pytorch dapat dilakukan dengan cuda ataupun Pytorch dan cpu saja.

Berikut cara instalasi Pytorch:

1. Kunjungi situs web pytorch, berikut link website <https://pytorch.org/>.
2. Perhatikan gambar 9.97. Pastikan teman-teman memilih stable build pytorch dan menyesuaikan sistem operasi dengan komputer teman-teman. Apabila teman-teman menggunakan conda environment maka pilih conda dan bahasa pemrograman python. Jika teman-teman menggunakan GPU maka select Cuda dengan versi yang teman-teman inginkan, pilih CPU apabila menggunakan CPU. Pada run this command teman-teman akan melihat perinta instalasi menggunakan conda, lalu copy perintah tersebut.

**Gambar 9.97** Select Pytorch with Conda

3. Jika teman-teman menggunakan conda environment, buka anaconda prompt, lalu ketikkan perintah berikut untuk pindah direktori.

```
cd ProgramData\Anaconda3\envs
```

perintah ini untuk mengaktifkan conda environment teman-teman. Ganti your-environment dengan nama conda environment yang sebelumnya telah teman-teman buat.

```
conda activate your-environment
```

4. Jika conda environment telah aktif maka teman-teman paste kan perintah yang telah di copy sebelumnya lalu tekan enter.

```
1 conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c
pytorch
```

```
(Real-Time-Voice-Cloning) C:\ProgramData\Anaconda3\envs>conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done

--> WARNING: A newer version of conda exists. <=#
current version: 4.10.3
latest version: 4.11.0

Please update conda by running

$ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\ProgramData\Anaconda3\envs\Real-Time-Voice-Cloning

added / updated specs:
- cudatoolkit=10.2
- pytorch
- torchaudio
- torchvision
```

Gambar 9.98 Install Pytorch with Anaconda Prompt

5. Ketikkan y lalu tekan enter. Tunggu hingga proses instalasi Pytorch selesai.

```

Administrator: Anaconda Prompt (Anaconda3) - conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
mkl          pkgs/main/win-64::mkl-2021.4.0-haa95532_640
mkl-service   pkgs/main/win-64::mkl-service-2.4.0-py37hbfffb1b_0
mkl_fft       pkgs/main/win-64::mkl_fft-1.3.1-py37hb77e83a_0
mkl_random    pkgs/main/win-64::mkl_random-1.2.2-py37hf1a4ad_0
numpy         pkgs/main/win-64::numpy-1.21.3-py37hfa4e547_0
numpy-base    pkgs/main/win-64::numpy-base-1.21.3-py37h3cdebe75_0
olefile       pkgs/main/win-64::olefile-0.46-py37_0
pillow        pkgs/main/win-64::pillow-8.4.0-py37hd45dc43_0
pytorch       pytorch/win-64::pytorch-1.10.2-py3.7_cudai0.2_cudnn7_0
pytorch-mutex pkgs/main/noarch::pytorch-mutex-1.10.2-py3.7_h06a3088_0
torchaudio    pkgs/main/win-64::torchaudio-0.10.2-py37_cu102
torchvision   pytorch/win-64::torchvision-0.11.3-py37_cu102
typing_extensions pkgs/main/noarch::typing_extensions-3.10.0.2-py06a43088_0
zlib          pkgs/main/win-64::zlib-1.2.11-hbcc25b3_4
zstd          pkgs/main/win-64::zstd-1.4.14-h19a0ad4_0

The following packages will be UPDATED:
ca-certificates anaconda::ca-certificates-2020.10.14.0 --> pkgs/main::ca-certificates-2021.10.26-haa95532_4
certifi        anaconda::certifi-2020.6.20-py37_0 --> pkgs/main::certifi-2021.10.8-py37haa95532_2

Proceed ([y]/n)? y

Downloading and Extracting Packages
pytorch-1.10.2          | 958.6 MB | ###### #################################### | 99%

```

Gambar 9.99 Proses Install Pytorch with Anaconda Prompt

- Apabila instalasi telah selesai maka teman-teman bisa melakukan pengecekan pada package pytorch yang telah di install.

```

Administrator: Anaconda Prompt (Anaconda3) - conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch
ca-certificates anaconda::ca-certificates-2020.10.14.0 --> pkgs/main::ca-certificates-2021.10.26-haa95532_4
certifi        anaconda::certifi-2020.6.20-py37_0 --> pkgs/main::certifi-2021.10.8-py37haa95532_2

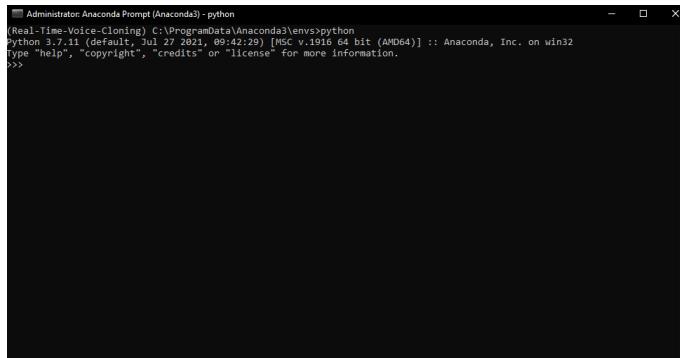
Proceed ([y]/n)? y

Downloading and Extracting Packages
pytorch-1.10.2          | 958.6 MB | ###### #################################### | 99%

```

Gambar 9.100 Instalasi Pytorch Selesai

- untuk memastikan apakah pytorch sudah berhasil di install maka teman-teman dapat melakukan pengecekan dengan cara mengetikkan perintah python lalu tekan enter.

**Gambar 9.101** Check Instalasi Pytorch

8. Selanjutnya teman-teman ketikkan perintah import torch lalu tekan enter.

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18)
Type "help", "copyright", "credits" or "license" for more information
>>> import torch
>>>
```

Gambar 9.102 Import Pytorch

9. Ketikkan perintah berikut untuk melihat versi pytorch yang telah berhasil di install.

```
print(torch.__version__)
```

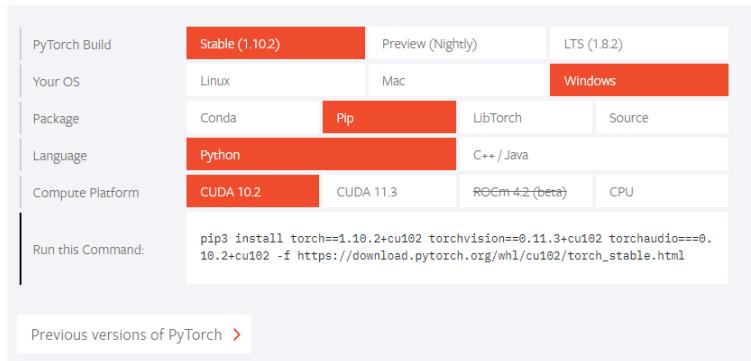
```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18)
Type "help", "copyright", "credits" or "license" for more information
>>> import torch
>>> print(torch.__version__)
1.10.2+cu102
>>>
```

Gambar 9.103 Print Pytorch Version

9.7.7 Download and install Pytorch dengan Pip

Berikut cara instalasi Pytorch menggunakan pip

1. Kunjungi situs web pytorch, berikut link website <https://pytorch.org/>.
2. Perhatikan gambar 9.104. Pastikan teman-teman memilih stable build pytorch dan menyesuaikan sistem operasi dengan komputer teman-teman. Apabila menggunakan virtual environment maka pilih Pip lalu copy perintah pada run this command.



Gambar 9.104 Select Pytorch with Pip

- Jika teman-teman menggunakan virtual environment, buka command prompt, lalu ketikkan perintah berikut untuk mengaktifkan virtual environment teman-teman. Ganti my-env dengan nama environment yang telah teman-teman buat sebelumnya.

```
.\my-env\Scripts\activate
```

- Jika virtual environment telah aktif maka teman-teman paste kan perintah yang telah di copy sebelumnya lalu tekan enter.

```
1 pip3 install torch==1.10.2+cu102 torchvision==0.11.3+cu102
  torchaudio==0.10.2+cu102 -f https://download.pytorch.org/whl/cu102/torch_stable.html
```

The screenshot shows a Command Prompt window titled 'Select Administrator: Command Prompt'. The user is running the command:

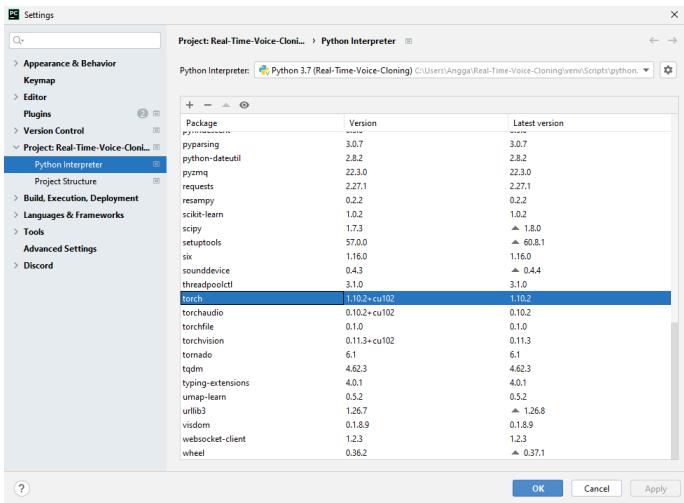
```
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>pip3 install torch==1.10.2+cu102 torchvision==0.11.3+cu102 torchaudio==0.10.2+cu102 -f https://download.pytorch.org/whl/cu102/torch_stable.html
```

The output shows the progress of the download and installation of Pytorch packages:

```
Looking in links: https://download.pytorch.org/whl/cu102/torch_stable.html
Collecting torch==1.10.2+cu102
  Downloading https://download.pytorch.org/whl/cu102/torch-1.10.2+cu102-cp37-cp37m-win_amd64.whl (1486.0 MB)
Collecting torchvision==0.11.3+cu102
  Downloading https://download.pytorch.org/whl/cu102/torchvision-0.11.3%2Bcu102-cp37-cp37m-win_amd64.whl (2.6 MB)
Collecting torchaudio==0.10.2+cu102
  Downloading https://download.pytorch.org/whl/cu102/torchaudio-0.10.2%2Bcu102-cp37-cp37m-win_amd64.whl (336 kB)
Collecting typing-extensions
  Using cached typing_extensions-4.0.1-py3-none-any.whl (22 kB)
Requirement already satisfied: numpy<1.21.0,>=1.19.2; python_version <= "3.6" and python_version >= "3.0" in c:\users\angga\real-time-voice-cloning\venv\lib\site-packages (from torchvision==0.11.3+cu102) (1.20.3)
Requirement already satisfied: numpy from c:\users\angga\real-time-voice-cloning\venv\lib\site-packages (from torchvision==0.11.3+cu102) (1.20.3)
Installing collected packages: typing-extensions, torch, torchvision, torchaudio
  Found existing installation: torch 1.10.2+cu102 torchvision 0.11.3+cu102 torchaudio 0.10.2+cu102
  WARNING: You are using pip version 21.1.3; however, version 21.0.3 is available.
  You should consider upgrading via the 'C:\Users\Angga\Real-Time-Voice-Cloning\venv\Scripts\python.exe -m pip install --upgrade pip' command.
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>
```

Gambar 9.105 Install Pytorch with Pip

- Tunggu hingga proses instalasi Pytorch selesai. Buka python interpreter dan lihat apakah package pytorch sudah ada atau belum.

**Gambar 9.106** Install Pytorch Berhasil

6. untuk memastikan apakah pytorch sudah berhasil di install maka teman-teman dapat melakukan pengecekan dengan cara mengetikkan perintah python lalu tekan enter.

```
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>python
python 3.7.9 (tags/v3.7.9:13cf9477c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 9.107 Check Version Pytorch

7. Selanjutnya teman-teman ketikkan perintah import torch lalu tekan enter.

```
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>python
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
```

Gambar 9.108 Import Torch

8. Ketikkan perintah berikut untuk melihat versi pytorch yang telah berhasil di install.

```
print(torch.__version__)
```

```
(venv) C:\Users\Angga\Real-Time-Voice-Cloning>python
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.__version__)
1.10.2+cu102
>>>
```

Gambar 9.109 Print Torch Version

9.8 Run Project Menggunakan Pycharm

Berikut tutorial menjalankan project Real-Time-Voice-Cloning menggunakan pycharm:

1. Buka file config.py yang terdapat didalam folder encoder, edit kode berikut sesuai dengan dataset yang teman-teman gunakan. Ganti value dari key clean sesuai dengan nama folder tempat teman-teman menyimpan dataset. Jika teman-teman hanya menggunakan satu dataset maka cukup abaikan salah satunya. Saya sarankan teman-teman mengganti nama variabel sesuai dengan dataset teman-teman.

```
1 #dataset 1
2 common_voice = {
3     "train": {
4         "clean": ["common_voice"]
5     },
6 }
```

```

6     "test": {
7         "clean": ["test_clean_data"]
8     },
9 }
10
11 #dataset 2
12 titml.datasets = {
13     "train": {
14         "clean": ["titml"]
15     },
16     "test": {
17         "clean": ["wwDataset"]
18     },
19 }
20
21 #contoh
22 nama_dataset = {
23     "train": {
24         "clean": ["nama_folder_dataset"]
25     },
26     "test": {
27         "clean": ["nama_folder_dataset"]
28     },
29 }
```

Listing 9.1 Config Dataset (Pycharm)

2. Buka file preprocess.py yang ada didalam folder encoder lalu edit kode berikut, sesuaikan dengan dataset yang teman-teman gunakan.

```

1 #import dataset sesuai dengan nama dataset pada config.py
2 #jika hanya satu dataset saja cukup importkan satu dataset , jika
3 #lebih dari dua maka importkan dan pisahkan dengan koma
4
5 from encoder.config import common_voice, titml_datasets
6
7 #ganti common_voice menjadi nama dataset pada config.py dan
8 #sesuaikan dengan yang diimportkan
9 #ganti wav sesuai dengan ekstensi file audio dataset teman-teman
10 #(mp3, flac, wav, m4a)
11
12 def preprocess_clean_dataset(datasets_root: Path, out_dir: Path,
13 skip_existing=False):
14     for dataset_name in common_voice["train"]["clean"]:
15         # Initialize the preprocessing
16         dataset_root, logger = _init_preprocess_dataset(
17             dataset_name, datasets_root, out_dir)
18         if not dataset_root:
19             return
20
21         # Preprocess all speakers
22         speaker_dirs = list(dataset_root.glob("*"))
23         _preprocess_speaker_dirs(speaker_dirs, dataset_name,
24             datasets_root, out_dir, "wav",
25                                     skip_existing, logger)
26
27 #sama dengan diatas .
```

```

22 #catatan: jika teman–teman hanya menggunakan satu dataset maka
23     # hapus function ini, abaikan, atau komen.
24 #jika menggunakan tiga dataset maka tambahkan function ini dan
25     # sesuaikan dengan dataset teman–teman seperti cara di atas.
26
27 def preprocess_speech_dataset(datasets_root: Path, out_dir: Path,
28     skip_existing=False):
29     for dataset_name in titml_datasets["train"]["clean"]:
30         # Initialize the preprocessing
31         dataset_root, logger = _init_preprocess_dataset(
32             dataset_name, datasets_root, out_dir)
33         if not dataset_root:
34             return
35
36         # Preprocess all speakers
37         speaker_dirs = list(dataset_root.glob("*"))
38         _preprocess_speaker_dirs(speaker_dirs, dataset_name,
39             datasets_root, out_dir, "wav",
40             skip_existing, logger)

```

Listing 9.2 Preprocessing Function (Pycharm)

3. Selanjutnya edit file encoder.preprocessing.py seperti pada kode 10.6.

```

1 from encoder.preprocess import preprocess_clean_dataset,
2     preprocess_speech_dataset
3 from utils.argutils import print_args
4 from pathlib import Path
5 import argparse
6
7 if __name__ == "__main__":
8     class MyFormatter(argparse.ArgumentDefaultsHelpFormatter,
9         argparse.RawDescriptionHelpFormatter):
10         pass
11
12     parser = argparse.ArgumentParser(
13         description="Preprocesses audio files from datasets,
14         encodes them as mel spectrograms and "
15         "writes them to the disk. This will allow you
16         to train the encoder. The "
17         "datasets required are at least one of
18         VoxCeleb1, VoxCeleb2 and LibriSpeech."
19         " Ideally, you should have all three. You
20         should extract them as they are "
21         " after having downloaded them and put them in
22         a same directory, e.g.:\n"
23         " -[datasets_root]\\n"
24         " -LibriSpeech\\n"
25         " -train-other-500\\n"
26         " -VoxCeleb1\\n"
27         " -wav\\n"
28         " -vox1.meta.csv\\n"
29         " -VoxCeleb2\\n"
30         " -dev",
31         formatter_class=MyFormatter
32     )

```

```
27     parser.add_argument("datasets_root", type=Path, help=\
28         "Path to the directory containing your LibriSpeech/TTS  
and VoxCeleb datasets.")
29     parser.add_argument("--o", "--out_dir", type=Path, default=
30         argparse.SUPPRESS, help=\
31             "Path to the output directory that will contain the mel  
spectrograms. If left out, "
32             "defaults to <datasets_root>/SV2TTS/encoder/")
33     parser.add_argument("--d", "--datasets", type=str,
34         default="titml,common_voice", help=\
35             "Comma-separated list of the name of the datasets you  
want to preprocess. Only the train"
36             "set of these datasets will be used. Possible names:  
librispeech_other, voxceleb1, "
37             "voxceleb2.")
38     parser.add_argument("-s", "--skip_existing", action="\
39         store_true", help=\
40             "Whether to skip existing output files with the same name  
. Useful if this script was  
"interrupted.")
41     parser.add_argument("--no_trim", action="store_true", help=\
42             "Preprocess audio without trimming silences (not  
recommended).")
43     args = parser.parse_args()
44
45     # Verify webrtcvad is available
46     if not args.no_trim:
47         try:
48             import webrtcvad
49         except:
50             raise ModuleNotFoundError("Package 'webrtcvad' not  
found. This package enables  
"noise removal and is recommended. Please install  
and try again. If installation fails,  
"use --no_trim to disable this error message.")
51     del args.no_trim
52
53     # Process the arguments
54     args.datasets = args.datasets.split(",")
55     if not hasattr(args, "out_dir"):
56         args.out_dir = args.datasets_root.joinpath("SV2TTS", "encoder")
57     assert args.datasets_root.exists()
58     args.out_dir.mkdir(exist_ok=True, parents=True)
59
60     # Preprocess the datasets
61     print_args(args, parser)
62     preprocess_func = {
63         "titml": preprocess_speech_dataset,
64         "common_voice": preprocess_clean_dataset,
65     }
66     args = vars(args)
67     for dataset in args.pop("datasets"):
68         print("Preprocessing %s" % dataset)
```

```
70 preprocess_func[dataset](**args)
```

Listing 9.3 Preprocessing Encoder Model (Pycharm)

Pada argument dataset masukkan nama dataset yang teman-teman gunakan, pada kode diatas saya menggunakan dataset titml dan common voice berbahasa indonesia

- Buka terminal yang ada pada pycharm lalu ketikkan perintah berikut

```
1 python encoder_preprocess.py datasets
```

Listing 9.4 Script to Run Preprocessing Speaker Encoder Model (Pycharm)



```
Terminal: Local + 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Angga\Real-Time-Voice-Cloning> python .\encoder_preprocess.py .\datasets\
```

Gambar 9.110 Preprocessing Speaker Encoder Model (Pycharm)

- Buka file encoder_train.py lalu edit argument –model_dir, pada bagian default isikan path direktori tempat model encoder akan disimpan.

```
1 from utils.argutils import print_args
2 from encoder.train import train
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Trains the speaker encoder. You must have
10        run encoder_preprocess.py first.",
11        formatter_class=argparse.ArgumentDefaultsHelpFormatter
12    )
13
14    parser.add_argument("run_id", type=str, help=
15        "Name for this model instance. If a model state from the
16        same run ID was previously "
17        "saved, the training will restart from there. Pass -f to
18        overwrite saved states and "
19        "restart from scratch.")
20    parser.add_argument("clean_data_root", type=Path, help=
21        "Path to the output directory of encoder_preprocess.py.
22        If you left the default "
23        "output directory when preprocessing, it should be <
24        datasets_root >/SV2TTS/encoder/.")
25    parser.add_argument("-m", "--models_dir", type=Path, default=
26        "/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-
27        Cloning/encoder/saved_models/", help=
28        "Path to the output directory that will contain the saved
29        model weights, as well as "
30        "backups of those weights and plots generated during
31        training.")
```

```

23     parser.add_argument("-v", "--vis_every", type=int, default
24         =10, help= \
25             "Number of steps between updates of the loss and the
26             plots.")
27     parser.add_argument("-u", "--umap_every", type=int, default
28         =100, help= \
29             "Number of steps between updates of the umap projection.
30             Set to 0 to never update the "
31             "projections.")
32     parser.add_argument("-s", "--save_every", type=int, default
33         =500, help= \
34             "Number of steps between updates of the model on the disk
35             . Set to 0 to never save the "
36             "model.")
37     parser.add_argument("-b", "--backup_every", type=int, default
38         =7500, help= \
39             "Number of steps between backups of the model. Set to 0
40             to never make backups of the "
41             "model.")
42     parser.add_argument("-f", "--force_restart", action="store_true",
43         help= \
44             "Do not load any saved model.")
45     parser.add_argument("--visdom_server", type=str, default="http://localhost")
46     parser.add_argument("--no_visdom", action="store_true", help=
        \
        "Disable visdom.")

args = parser.parse_args()

# Process the arguments
args.models_dir.mkdir(exist_ok=True)

# Run the training
print_args(args, parser)
train(**vars(args))

```

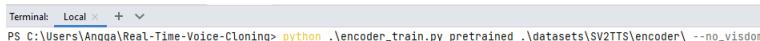
Listing 9.5 Training Speaker Encoder Model (Pycharm)

6. Selanjutnya, ketikkan kode program untuk menjalankan training speaker encoder model. Tambahkan argument –no_visdom apabila teman-teman tidak menggunakan visdom server untuk menampilkan visualisasi dari hasil training speaker encoder model.

```
1 python encoder_train.py pretrained datasets/SV2TTS/encoder/ --
  no_visdom
```

Listing 9.6 Script to Run Training Speaker Encoder Model

pretrained merupakan nama file model yang akan di save, teman-teman bisa mengubah-nya menjadi encoder atau apapun yang teman-teman inginkan dan pastikan path datasets nya benar.



```
Terminal: Local + ▾
PS C:\Users\Angga\Real-Time-Voice-Cloning> python .\encoder_train.py pretrained .\datasets\SV2TTS\encoder\ --no_visdom
```

Gambar 9.111 Training Speaker Encoder Model (Pycharm)

7. Buka file `synthesizer_preprocess_audio.py` lalu edit argument `--datasets_name`, pada bagian default isikan nama folder dataset teman-teman dan argumen `--subfolders` pada default isikan nama sub folder dari folder dataset. Jika terdapat lebih dari satu subfolder maka pisahkan subfolder 1 dan 2 menggunakan koma.

```
1 from synthesizer.preprocess import preprocess_dataset
2 from synthesizer.hparams import hparams
3 from utils.argutils import print_args
4 from pathlib import Path
5 import argparse
6
7
8 if __name__ == "__main__":
9     parser = argparse.ArgumentParser(
10         description="Preprocesses audio files from datasets, encodes them as mel spectrograms and writes them to the disk. Audio files are also saved, to be used by the \"vocoder for training.\"", formatter_class=argparse.ArgumentDefaultsHelpFormatter)
11     parser.add_argument("datasets_root", type=Path, help="Path to the directory containing your LibriSpeech/TTS datasets.")
12     parser.add_argument("-o", "--out_dir", type=Path, default=argparse.SUPPRESS, help="Path to the output directory that will contain the mel spectrograms, the audios and the \"embeds. Defaults to <datasets_root>/SV2TTS/synthesizer/")
13     parser.add_argument("-n", "--n_processes", type=int, default=4, help="Number of processes in parallel.")
14     parser.add_argument("-s", "--skip_existing", action="store_true", help="Whether to overwrite existing files with the same name. Useful if the preprocessing was \"interrupted.")
15     parser.add_argument("--hparams", type=str, default="", help="Hyperparameter overrides as a comma-separated list of name-value pairs")
16     parser.add_argument("--no_alignments", action="store_true", help="Use this option when dataset does not include alignments \
17         (these are used to split long audio files into sub-utterances.)")
18     parser.add_argument("--datasets_name", type=str, default="synthesizer_dataset", help="Name of the dataset directory to process.")
19     parser.add_argument("--subfolders", type=str, default="cv-corpus-7.0-2021-07-21", help="Comma-separated list of subfolders to process inside your dataset directory")
20     args = parser.parse_args()
```

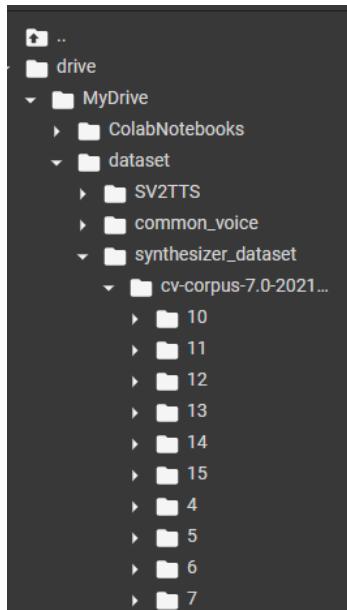
```

36 # Process the arguments
37 if not hasattr(args, "out_dir"):
38     args.out_dir = args.datasets_root.joinpath("SV2TTS", "synthesizer")
39
40 # Create directories
41 assert args.datasets_root.exists()
42 args.out_dir.mkdir(exist_ok=True, parents=True)
43
44 # Preprocess the dataset
45 print_args(args, parser)
46 args.hparams = hparams.parse(args.hparams)
47 preprocess_dataset(**vars(args))

```

Listing 9.7 Training Speaker Encoder Model (Pycharm)

Berikut struktur folder dataset yang saya gunakan:

**Gambar 9.112** Struktur Folder Dataset Synthesizer

- Selanjutnya, ketikkan kode program untuk menjalankan preprocessing audio. Tambahkan argumen `--no_alignments` jika dataset yang teman-teman gunakan tidak memiliki alignments.

```
1 python synthesizer_preprocess_audio.py datasets --no_alignments
```

Listing 9.8 Preprocessing Audio (Pycharm)

```
Terminal: Local × + ▾
PS C:\Users\Angga\Real-Time-Voice-Cloning> python .\synthesizer_preprocess_audio.py .\datasets\
```

Gambar 9.113 Preprocessing Audio (Pycharm)

- Buka file `synthesizer_preprocess_embed.py` lalu edit pada bagian argument `--encoder_model_fpath` dan isikan lokasi penyimpanan model speaker encoder yang telah di training sebelumnya pada bagian defaults.

```

1 from synthesizer.preprocess import create_embeddings
2 from utils.argutils import print_args
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Creates embeddings for the synthesizer from
10        the LibriSpeech utterances.",
11        formatter_class=argparse.ArgumentDefaultsHelpFormatter
12    )
13    parser.add_argument("synthesizer_root", type=Path, help=\
14        "Path to the synthesizer training data that contains the
15        audios and the train.txt file."
16        "If you let everything as default, it should be <
17        datasets_root>/SV2TTS/synthesizer/.")
18    parser.add_argument("-e", "--encoder_model_fpath", type=Path,
19        default="/content/drive/MyDrive/
20        ColabNotebooks/Real-Time-Voice-Cloning/encoder/saved_models/
21        pretrained.pt", help=\
22        "Path your trained encoder model.")
23    parser.add_argument("-n", "--n_processes", type=int, default
24        =4, help= \
25        "Number of parallel processes. An encoder is created for
26        each, so you may need to lower "
27        "this value on GPUs with low memory. Set it to 1 if CUDA
28        is unhappy.")
29    args = parser.parse_args()
30
31    # Preprocess the dataset
32    print_args(args, parser)
33    create_embeddings(**vars(args))

```

Listing 9.9 Training Speaker Encoder Model (Pycharm)

- Selanjutnya, ketikkan kode program untuk menjalankan preprocessing audio embeddings.

```

1 python synthesizer_preprocess_embeds.py datasets/SV2TTS/
2 synthesizer

```

Listing 9.10 Preprocessing Embeds (Pycharm)

Terminal: Local +
PS C:\Users\Angga\Real-Time-Voice-Cloning> python ./synthesizer_preprocess_embeds.py ./datasets\SV2TTS\synthesizer\

Gambar 9.114 Preprocessing Audio Embeddings (Pycharm)

11. Buka file synthesizer_train.py lalu edit argument –models_dir, pada bagian default isikan lokasi penyimpanan hasil training model synthesizer.

```

1 from pathlib import Path
2
3 from synthesizer.hparams import hparams
4 from synthesizer.train import train
5 from utils.argutils import print_args
6 import argparse
7
8
9 if __name__ == "__main__":
10     parser = argparse.ArgumentParser()
11     parser.add_argument("--run_id", type=str, help= \
12         "Name for this model. By default, training outputs will \
13         be stored to saved_models/<run_id>. If a model state " \
14         "from the same run ID was previously saved, the training \
15         will restart from there. Pass -f to overwrite saved " \
16         "states and restart from scratch.")
17     parser.add_argument("--syn_dir", type=Path, help= \
18         "Path to the synthesizer directory that contains the \
19         ground truth mel spectrograms, " \
20         "the wavs and the embeds.")
21     parser.add_argument("-m", "--models_dir", type=Path, default= \
22         "/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice- \
23         Cloning/synthesizer/saved.models", help= \
24         "Path to the output directory that will contain the saved \
25         model weights and the logs.")
26     parser.add_argument("-s", "--save_every", type=int, default= \
27         1000, help= \
28         "Number of steps between updates of the model on the disk \
29         . Set to 0 to never save the " \
30         "model.")
31     parser.add_argument("-b", "--backup_every", type=int, default= \
32         25000, help= \
33         "Number of steps between backups of the model. Set to 0 \
34         to never make backups of the " \
35         "model.")
36     parser.add_argument("-f", "--force_restart", action=" \
37         store_true", help= \
38         "Do not load any saved model and restart from scratch.")
39     parser.add_argument("--hparams", default="", help= \
40         "Hyperparameter overrides as a comma-separated list of \
41         name=value pairs")
42     args = parser.parse_args()
43     print_args(args, parser)
44
45     args.hparams = hparams.parse(args.hparams)
46
47     # Run the training

```

```
36     train(**vars(args))
```

Listing 9.11 Training Speaker Encoder Model (Pycharm)

12. Selanjutnya, ketikkan kode program untuk menjalankan proses training synthesizer model.

```
1 python synthesizer_train.py pretrained datasets\SV2TTS\
    synthesizer
```

Listing 9.12 Training Synthesizer Model (Pycharm)

```
Terminal: Local + ~
PS C:\Users\Angga\Real-Time-Voice-Cloning> python ./synthesizer_train.py pretrained .\datasets\SV2TTS\synthesizer\
Arguments:
  run_id:      pretrained
  syn_dir:     .\datasets\SV2TTS\synthesizer\
  models_dir:  synthesizer\saved_models/
  save_every:   1000
  backup_every: 25000
  force_restart: False
  hparams:

Checkpoint path: synthesizer\saved_models\pretrained\pretrained.pt
Loading training data from: datasets\SV2TTS\synthesizer\train.txt
Using model: Tacotron
```

Gambar 9.115 Training Synthesizer Model (Pycharm)

13. Buka file vocoder_preprocess.py lalu edit argument--syn_model_fpath, pada bagian default isikan lokasi penyimpanan hasil training model synthesizer.

```
1 import argparse
2 import os
3 from pathlib import Path
4
5 from synthesizer.hparams import hparams
6 from synthesizer.synthesize import run_synthesis
7 from utils.argutils import print_args
8
9
10 if __name__ == "__main__":
11     class MyFormatter(argparse.ArgumentDefaultsHelpFormatter,
12                         argparse.RawDescriptionHelpFormatter):
13         pass
14
15     parser = argparse.ArgumentParser(
16         description="Creates ground-truth aligned (GTA) spectrograms from the vocoder.",
17         formatter_class=MyFormatter)
18     parser.add_argument("datasets_root", type=Path, help=(
19         "Path to the directory containing your SV2TTS directory.\nIf you specify both --in_dir and "
20         "--out_dir, this argument won't be used."))
21     parser.add_argument("-s", "--syn_model_fpath", type=Path,
22                         default="/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer/saved_models/pretrained/synthesizer.pt",
```

```

23             help="Path to a saved synthesizer")
24     parser.add_argument("-i", "--in_dir", type=Path, default=
25                         argparse.SUPPRESS, help= \
26                             "Path to the synthesizer directory that contains the mel
27                             spectrograms, the wavs and the "
28                             "embeds. Defaults to <datasets_root>/SV2TTS/synthesizer
29                             /.")
30     parser.add_argument("-o", "--out_dir", type=Path, default=
31                         argparse.SUPPRESS, help= \
32                             "Path to the output vocoder directory that will contain
33                             the ground truth aligned mel "
34                             "spectrograms. Defaults to <datasets_root>/SV2TTS/vocoder
35                             /.")
36     parser.add_argument("--hparams", default="", help=\
37         "Hyperparameter overrides as a comma-separated list of
38         name=value pairs")
39     parser.add_argument("--cpu", action="store_true", help=\
40         "If True, processing is done on CPU, even when a GPU is
41         available.")
42     args = parser.parse_args()
43     print_args(args, parser)
44     modified_hp = hparams.parse(args.hparams)
45
46     if not hasattr(args, "in_dir"):
47         args.in_dir = args.datasets_root / "SV2TTS" / "synthesizer"
48     if not hasattr(args, "out_dir"):
49         args.out_dir = args.datasets_root / "SV2TTS" / "vocoder"
50
51     if args.cpu:
52         # Hide GPUs from Pytorch to force CPU processing
53         os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
54
55     run_synthesis(args.in_dir, args.out_dir, args.syn_model_fpath
56                 , modified_hp)

```

Listing 9.13 Vocoder Preprocess (Pycharm)

14. Selanjutnya, ketikkan kode program untuk menjalankan proses preprocessing vocoder model.

```
! python vocoder_preprocess.py datasets
```

Listing 9.14 Preprocessing Vococder Model (Pycharm)

```

Terminal: Local + ▾
PS C:\Users\Angga\Real-Time-Voice-Cloning> python .\vocoder_preprocess.py .\datasets\
Arguments:
  datasets_root: .\datasets\
  model_dir: synthesizer\saved_models\pretrained\
  hparams:
  no_trim:      False
  cpu:          False

```

Gambar 9.116 Preprocessing Vococder Model (Pycharm)

15. Buka file `vocoder.train.py` lalu edit argument `--models_dir`, pada bagian default isikan lokasi untuk menyimpan hasil training vocoder model.

```

1 from utils.argutils import print_args
2 from vocoder.train import train
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Trains the vocoder from the synthesizer
10        audios and the GTA synthesized mels, "
11        "or ground truth mels.",
12        formatter_class=argparse.ArgumentDefaultsHelpFormatter
13    )
14
15    parser.add_argument("run_id", type=str, help= \
16        "Name for this model instance. If a model state from the
17        same run ID was previously "
18        "saved, the training will restart from there. Pass -f to
19        overwrite saved states and "
20        "restart from scratch.")
21    parser.add_argument("datasets_root", type=str, help= \
22        "Path to the directory containing your SV2TTS directory.
23        Specifying --syn_dir or --voc_dir "
24        "will take priority over this argument.")
25    parser.add_argument("--syn_dir", type=str, default=argparse.
26        SUPPRESS, help= \
27        "Path to the synthesizer directory that contains the
28        ground truth mel spectrograms, "
29        "the wavs and the embeds. Defaults to <datasets_root>/
30        SV2TTS/synthesizer/")
31    parser.add_argument("--voc_dir", type=str, default=argparse.
32        SUPPRESS, help= \
33        "Path to the vocoder directory that contains the GTA
34        synthesized mel spectrograms. "
35        "Defaults to <datasets_root>/SV2TTS/vocoder/. Unused if
--ground_truth is passed.")
36    parser.add_argument("-m", "--models_dir", type=str, default="\
37        /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning
38        /vocoder/saved_models/", help=\
39        "Path to the directory that will contain the saved model
40        weights, as well as backups "
41        "of those weights and wavs generated during training.")
42    parser.add_argument("-g", "--ground_truth", action="\
43        store_true", help= \
44        "Train on ground truth spectrograms (<datasets_root>/
45        SV2TTS/synthesizer/mels).")
46    parser.add_argument("-s", "--save_every", type=int, default \
47        =1000, help= \
48        "Number of steps between updates of the model on the disk
49        . Set to 0 to never save the "
50        "model.")
51    parser.add_argument("-b", "--backup_every", type=int, default \
52        =25000, help= \
53        "

```

```

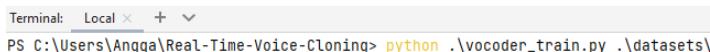
36     "Number of steps between backups of the model. Set to 0
37     to never make backups of the "
38     "model.")
39     parser.add_argument("-f", "--force_restart", action="
40     store_true", help= \
41         "Do not load any saved model and restart from scratch.")
42     args = parser.parse_args()
43
44     # Process the arguments
45     if not hasattr(args, "syn_dir"):
46         args.syn_dir = Path(args.datasets_root, "SV2TTS", "synthesizer")
47     args.syn_dir = Path(args.syn_dir)
48     if not hasattr(args, "voc_dir"):
49         args.voc_dir = Path(args.datasets_root, "SV2TTS", "vocoder")
50     args.voc_dir = Path(args.voc_dir)
51     del args.datasets_root
52     args.models_dir = Path(args.models_dir)
53     args.models_dir.mkdir(exist_ok=True)
54
55     # Run the training
56     print_args(args, parser)
57     train(**vars(args))

```

Listing 9.15 Vocoder Train (Pycharm)

16. Selanjutnya, ketikkan kode program untuk menjalankan proses training vocoder model. Tambahkan argument --ground-truth agar proses training dilakukan menggunakan mel spektrogram yang telah di preprocessing sebelumnya pada model synthesizer.

```
| python vocoder-train.py pretrained datasets --ground-truth
```

Listing 9.16 Training Vococder Model (Pycharm)


Terminal: Local + ▾
PS C:\Users\Angga\Real-Time-Voice-Cloning> python .\vocoder_train.py .\datasets\

Gambar 9.117 Training Vococder Model (Pycharm)

BAB 10

TUTORIAL PEMBUATAN PROJECT MENGGUNAKAN GOOGLE COLAB

Google colab atau Google Colaboratory merupakan solusi yang sangat baik apabila teman-teman memiliki masalah pada tutorial sebelumnya atau mungkin komputer teman-teman tidak memiliki aspek yang cukup memadai dalam pembuatan project ini. Google colab adalah sebuah software buatan Google yang digunakan untuk keperluan research di bidang data science dan machine learning. Hal yang lebih menakjubkannya lagi, google colab menyediakan free GPU yang akan sangat kita butuhkan dalam project ini. Selain GPU masih banyak keunggulan lainnya dari google colab yang akan membantu kita seperti fitur yang memungkinkan google colab terhubung dengan google drive sebagai media penyimpanan dataset, kodingan, hasil training dan model. Karena dapat di simpan di google drive tentu membuat project ini mudah untuk dibagikan kepada tim riset agar dapat mengetahui progres dan hasilnya.

Apabila teman-teman merupakan orang yang menyukai sesuatu hal yang gratis tetapi tidak praktis maka saya sarankan untuk menggunakan google colab untuk mengerjakan project ini. Tetapi jika teman-teman memiliki komputer dengan spesifikasi GPU yang sangat baik dan melebihi GPU gratis yang diberikan google colab maka akan lebih baik menggunakan komputer pribadi saja. Masing-masing cara memiliki kelebihan dan kekurangan. Menurut saya memanfaatkan google colab memang gratis namun sangat tidak praktis karena teman-teman akan membutuhkan akun google yang sangat banyak. Sementara itu menggunakan komputer pribadi mungkin membutuhkan lebih banyak biaya tetapi akan sangat praktis, tidak butuh mendaftar banyak akun google dan tidak terbatas waktu. Perlu teman-teman ketahui bahwa free GPU google colab tidak berlaku selamanya tetapi hanya 12 jam per akun google setelah

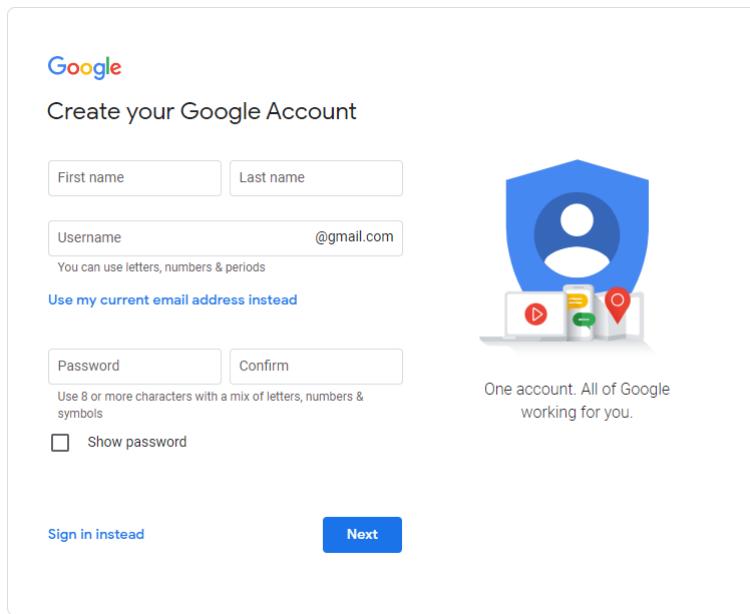
itu teman-teman harus mengganti akun google yang telah mencapai limit penggunaan GPU dengan akun google yang baru.

Berikut tata cara pembuatan Voice Cloning Project menggunakan google colab:

10.1 Membuat Akun Google

Sebelum kita memasuki tata cara penggunaan google colab, kita harus mengetahui cara untuk membuat akun google terlebih dahulu. Ikuti ya tahap-tahap berikut ini:

1. Buat sebuah akun google baru, kunjungi link berikut <https://accounts.google.com/signup/v2/webcreateaccount?hl=en&flowName=GlifWebSignIn&flowEntry=SignUp>.



Gambar 10.1 Create Google Account

2. Isikan data diri teman-teman pada form seperti nama, username, dan password. Kemudian klik next

Google

Create your Google Account

First name Last name

Username @gmail.com

You can use letters, numbers & periods

Available: abuat0854 fortraining28 buata7332

[Use my current email address instead](#)

Password Confirm

Use 8 or more characters with a mix of letters, numbers & symbols

Show password

[Sign in instead](#) [Next](#)



One account. All of Google working for you.

Gambar 10.2 Isi form pendaftaran

3. Isikan nomor handphone teman-teman yang aktif untuk verifikasi pembuatan akun google lalu klik next.

Google

Verify your phone number

For your security, Google wants to make sure it's really you. Google will send a text message with a 6-digit verification code. *Standard rates apply*

Phone number

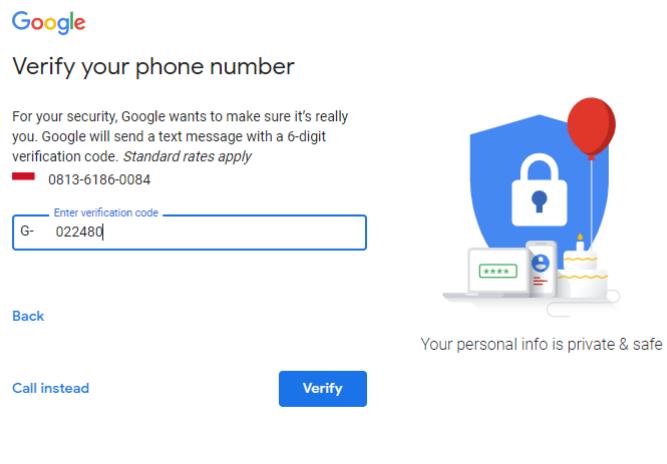
[Back](#) [Next](#)



Your personal info is private & safe

Gambar 10.3 Isikan Nomor Handphon

4. Cek handphone teman-teman apakah ada sms dari google yang berisikan kode verifikasi, jika ada masukkan 6 digit kode verifikasi tersebut lalu klik verify.



Gambar 10.4 Verify Phone Number

5. Isikan data-data tambahan seperti email pemulihan, tanggal lahir, dan jenis kelamin pada form. Klik Next.



akun, welcome to Google

akuntraining38@gmail.com

Phone number (optional)

Google will use this number only for account security. Your number won't be visible to others. You can choose later whether to use it for other purposes.

Recovery email address (optional)

We'll use it to keep your account secure

Month

Day

Year

Your birthday

Gender



Your personal info is private & safe

[Why we ask for this information](#)

[Back](#)

[Next](#)

Gambar 10.5 Form Personal Data

6. Pada tahap Get more from your number, teman-teman bisa klik skip atau klik yes, tergantung pada pilihan masing-masing. Jika mengklik Yes, I'm in maka teman-teman telah setuju bahwa nomor handphone teman-teman dapat digunakan diseluruh layanan google.



Get more from your number

If you like, you can add your phone number to your account for use across Google services. [Learn more](#)

For example, your number will be used to

Receive video calls & messages

G Make Google services, including ads, more relevant to you

[More options](#)



Your personal info is private & safe

[Back](#)

[Skip](#)

[Yes, I'm in](#)

Gambar 10.6 Get more from your number

- Pada tahap Privacy and Terms klik I agree untuk menyelesaikan pendaftaran akun google.

Combining data

We also combine this data among our services and across your devices for these purposes. For example, depending on your account settings, we show you ads based on information about your interests, which we can derive from your use of Search and YouTube, and we use data from trillions of search queries to build spell-correction models that we use across all of our services.

You're in control

Depending on your account settings, some of this data may be associated with your Google Account and we treat this data as personal information. You can control how we collect and use this data now by clicking "More Options" below. You can always adjust your controls later or withdraw your consent for the future by visiting My Account ([myaccount.google.com](#)).

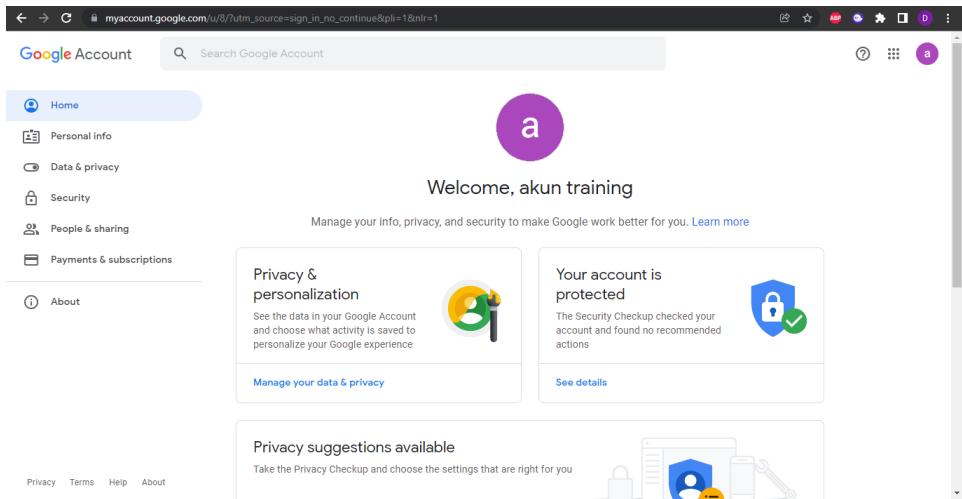
[More options ▾](#)

[Cancel](#)

[I agree](#)

Gambar 10.7 Create Google Account

- Jika tampilan website seperti gambar 10.8 maka selamat, teman-teman telah berhasil membuat akun google teman-teman.

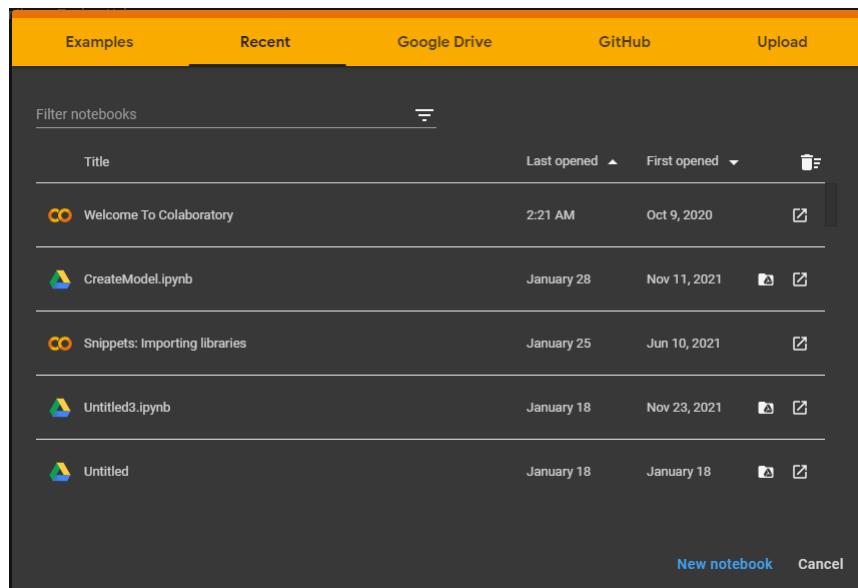


Gambar 10.8 *Create Google Account*

10.2 Membuat New Project Pada Google Colab

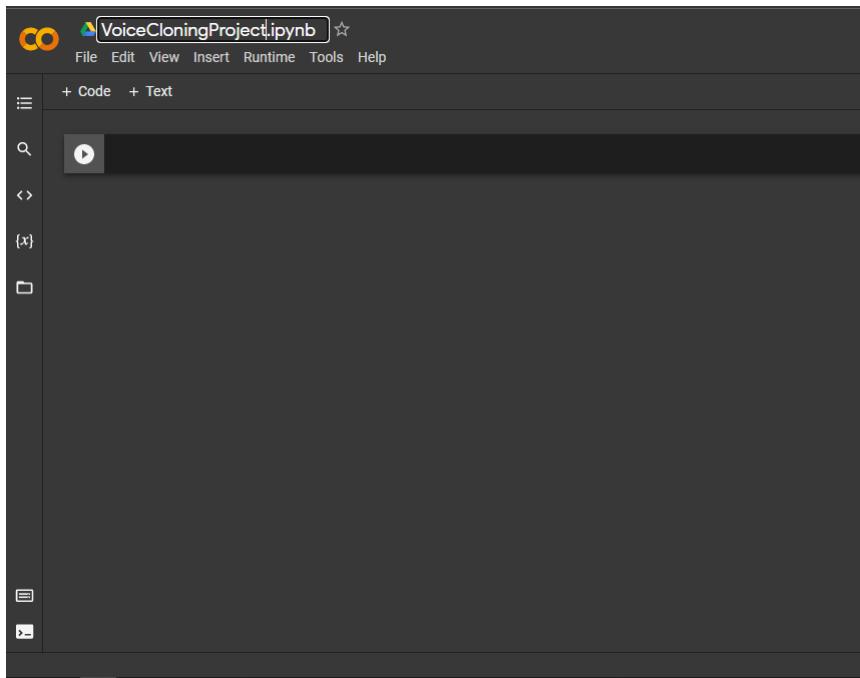
Saya akan mengajarkan kepada teman-teman cara membuat project baru di google colab dan menuliskan script code program yang akan kita gunakan untuk preprocessing dataset dan training tiga model yang akan kita butuhkan pada project ini. Berikut langkah-langkah pembuatannya:

1. Kunjungi website google colab atau akses link berikut <https://colab.research.google.com/>. Klik new project untuk membuat project google colab.



Gambar 10.9 Create New Project

2. Rename nama project menjadi VoiceCloningProject untuk memudahkan teman-teman melakukan pencarian file project ini pada google drive.

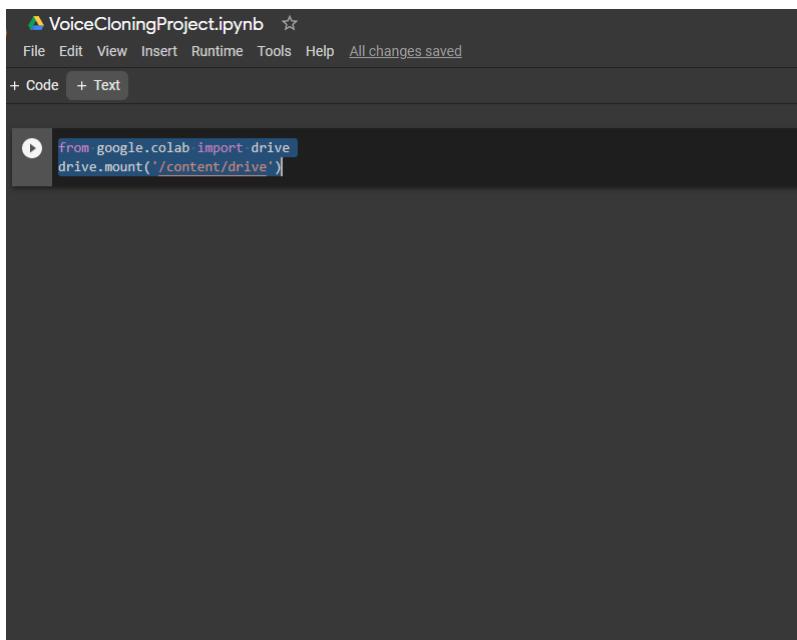


Gambar 10.10 Rename File

3. Ketikkan script berikut untuk melakukan mounting google colab ke google drive agar teman-teman dapat mengakses penyimpanan pada google drive.

```
1 from google.colab import drive  
2 drive.mount('/content/drive')
```

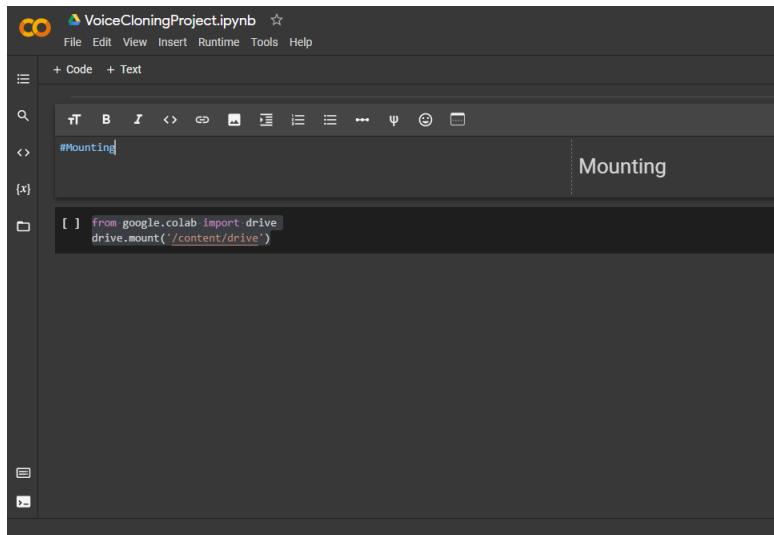
Listing 10.1 Mounting Google Drive



The screenshot shows the Google Colab interface with a dark theme. At the top, it displays the file name "VoiceCloningProject.ipynb" and a star icon. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". Underneath the menu is a toolbar with two buttons: "+ Code" and "+ Text". The main area is a code editor containing a single line of Python code: "from google.colab import drive; drive.mount('/content/drive')". A play button icon is positioned to the left of the code line.

Gambar 10.11 Script Mounting Google Drive

4. Klik text untuk mempercantik tampilan project teman-teman lalu ketikkan #Mounting.



The screenshot shows the Google Colab interface with a dark theme. At the top, it displays the file name "VoiceCloningProject.ipynb" and a star icon. Below the title bar is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help. Underneath the menu is a toolbar with various text styling icons (bold, italic, etc.) and a search bar. The main area is a text editor containing the placeholder text "#Mounting". To the right of the text, the word "Mounting" is displayed in a larger, bold font. The bottom of the screen shows a code editor with the same Python mounting script: "from google.colab import drive; drive.mount('/content/drive')". On the far left, there is a sidebar with a tree view of the project structure.

Gambar 10.12 Add Text to Project

- 5.Tambahkan kode untuk berpindah direktori ke folder Colab Notebooks yang ada didalam penyimpanan google drive teman-teman.

```
1 %cd /content/drive/MyDrive/ColabNotebooks
```

Listing 10.2 Change Directory

The screenshot shows a Google Colab interface. At the top, it says 'VoiceCloningProject.ipynb' with a star icon. Below the title bar are standard menu options: File, Edit, View, Insert, Runtime, Tools, Help, and 'All changes saved'. Underneath the menu is a toolbar with '+ Code' and '+ Text' buttons. The main workspace is titled 'Mounting'. A code cell is visible with the following content:

```
[ ] from google.colab import drive
drive.mount('/content/drive')
```

Below the code cell, another cell is partially visible with the command:

```
[ ] %cd /content/drive/MyDrive/ColabNotebooks
```

Gambar 10.13 Change Directory

6. Tambahkan lagi text dan ketikkan #Clone Repository lalu klik code untuk menambahkan kode berikut ini untuk clone repository dari github dan menginstall semua library yang ada di file requirements.txt.

```
1 %tensorflow_version 1.x
2 import os
3 from os.path import exists, join, basename, splitext
4
5 git_repo_url = 'https://github.com/dindamajesty13/Real-Time-Voice
   -Cloning.git'
6 project_name = splitext(basename(git_repo_url))[0]
7 if not exists(project_name):
8     # clone and install
9     !git clone -q --recursive {git_repo_url}
10    # install dependencies
11    !cd {project_name} && pip install -q -r requirements.txt
12    !pip install -q gdown
13    !apt-get install -qq libportaudio2
```

```
14 ! pip install -q https://github.com/tugstugi/dl-colab-notebooks/
      archive/colab_utils.zip
```

Listing 10.3 Clone Repository

Silahkan ganti git_repo_url dengan link repository teman-teman.

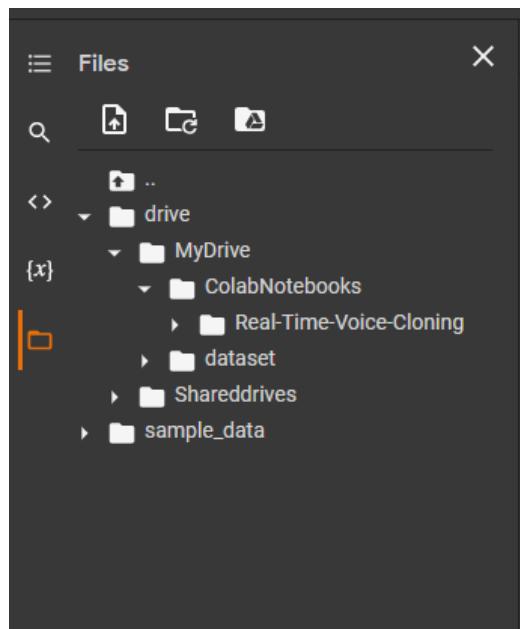
The screenshot shows a Google Colab notebook interface. The title bar says 'VoiceCloningProject.ipynb'. The main area has a sidebar on the left with sections like '+ Code' and '+ Text'. The main content area shows a code cell with the following Python script:

```
[ ] %tensorflow_version 1.x
import os
from os.path import exists, join, basename, splitext

git_repo_url = 'https://github.com/dindamajesty13/Real-Time-Voice-Cloning.git'
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
    # clone_and_install
    !git clone -q --recursive {git_repo_url}
    # install_dependencies
    !cd {project_name} && pip install -q -r requirements.txt
    !pip install -q gdown
    !apt-get install -qq libportaudio2
    !pip install -q https://github.com/tugstugi/dl-colab-notebooks/archive/colab_utils.zip
```

Gambar 10.14 *Script Clone Repository*

7. Apabila Clone Project telah selesai maka klik icon folder di kiri tampilan google colab teman-teman lalu cari folder Real-Time-Voice-Cloning untuk memastikan bahwa repositori telah berhasil di clone.



Gambar 10.15 Check Folder

8. Buka file config.py yang terdapat didalam folder encoder, edit kode berikut sesuai dengan dataset yang teman-teman gunakan. Ganti value dari key clean sesuai dengan nama folder tempat teman-teman menyimpan dataset. Jika teman-teman hanya menggunakan satu dataset maka cukup abaikan salah satunya. Saya sarankan teman-teman mengganti nama variabel sesuai dengan dataset teman-teman.

```
1 #dataset 1
2 common_voice = {
3     "train": {
4         "clean": ["common_voice"]
5     },
6     "test": {
7         "clean": ["test_clean_data"]
8     },
9 }
10
11 #dataset 2
12 titml.datasets = {
13     "train": {
14         "clean": ["titml"]
15     },
16     "test": {
17         "clean": ["wwDataset"]
18     },
19 }
20
21 #contoh
22 nama_dataset = {
```

```

23     "train": {
24         "clean": ["nama_folder_dataset"]
25     },
26     "test": {
27         "clean": ["nama_folder_dataset"]
28     },
29 }
```

Listing 10.4 Config Dataset

9. Buka file preprocess.py yang ada didalam folder encoder lalu edit kode berikut, sesuaikan dengan dataset yang teman-teman gunakan.

```

1 #import dataset sesuai dengan nama dataset pada config.py
2 #jika hanya satu dataset saja cukup importkan satu dataset , jika
3 #lebih dari dua maka importkan dan pisahkan dengan koma
4
5
6 #ganti common_voice menjadi nama dataset pada config.py dan
7 #sesuaikan dengan yang diimportkan
8 #ganti wav sesuai dengan ekstensi file audio dataset teman-teman
9 # (mp3, flac, wav, m4a)
10
11 def preprocess_clean_dataset(datasets_root: Path, out_dir: Path,
12     skip_existing=False):
13     for dataset_name in common_voice["train"]["clean"]:
14         # Initialize the preprocessing
15         dataset_root, logger = _init_preprocess_dataset(
16             dataset_name, datasets_root, out_dir)
17         if not dataset_root:
18             return
19
20         # Preprocess all speakers
21         speaker_dirs = list(dataset_root.glob("*"))
22         _preprocess_speaker_dirs(speaker_dirs, dataset_name,
23             datasets_root, out_dir, "wav",
24             skip_existing, logger)
25
26 #sama dengan diatas .
27 #catatan: jika teman-teman hanya menggunakan satu dataset maka
28 #hapus function ini , abaikan , atau komen.
29 #jika menggunakan tiga dataset maka tambahkan function ini dan
30 #sesuaikan dengan dataset teman-teman seperti cara di atas.
31
32 def preprocess_speech_dataset(datasets_root: Path, out_dir: Path,
33     skip_existing=False):
34     for dataset_name in titml_datasets["train"]["clean"]:
35         # Initialize the preprocessing
36         dataset_root, logger = _init_preprocess_dataset(
37             dataset_name, datasets_root, out_dir)
38         if not dataset_root:
39             return
40
41         # Preprocess all speakers
42         speaker_dirs = list(dataset_root.glob("*"))
```

```

34     _preprocess_speaker_dirs(speaker_dirs , dataset_name ,
35                               datasets_root , out_dir , "wav",
36                               skip_existing , logger)

```

Listing 10.5 Preprocessing Function

10. Selanjutnya edit file encoder.preprocessing.py seperti pada kode 10.6.

```

1  from encoder.preprocessing import preprocess_clean_dataset ,
2      preprocess_speech_dataset
3  from utils.argutils import print_args
4  from pathlib import Path
5  import argparse
6
7  if __name__ == "__main__":
8      class MyFormatter(argparse.ArgumentDefaultsHelpFormatter ,
9                      argparse.RawDescriptionHelpFormatter):
10         pass
11
12     parser = argparse.ArgumentParser(
13         description="Preprocesses audio files from datasets ,
14         encodes them as mel spectrograms and "
15             "writes them to the disk. This will allow you
16             to train the encoder. The "
17             "datasets required are at least one of
18             VoxCeleb1 , VoxCeleb2 and LibriSpeech . "
19             "Ideally , you should have all three. You
20             should extract them as they are "
21             "after having downloaded them and put them in
22             a same directory , e.g.: \n"
23             "-[datasets_root]\n"
24             "-LibriSpeech\n"
25             "-train-other-500\n"
26             "-VoxCeleb1\n"
27             "-wav\n"
28             "-vox1.meta.csv\n"
29             "-VoxCeleb2\n"
30             "-dev",
31             formatter_class=MyFormatter
32     )
33     parser.add_argument("datasets_root" , type=Path , help=\
34         "Path to the directory containing your LibriSpeech/TTS
35         and VoxCeleb datasets .")
36     parser.add_argument("-o" , "--out_dir" , type=Path , default=
37         argparse.SUPPRESS , help=\
38             "Path to the output directory that will contain the mel
39             spectrograms. If left out , "
40             "defaults to <datasets_root>/SV2TTS/encoder/")
41     parser.add_argument("-d" , "--datasets" , type=str ,
42                     default="titml,common_voice" , help=\
43             "Comma-separated list of the name of the datasets you
44             want to preprocess. Only the train "
45             "set of these datasets will be used. Possible names:
46             librispeech_other , voxceleb1 , "
47             "voxceleb2 .")

```

```

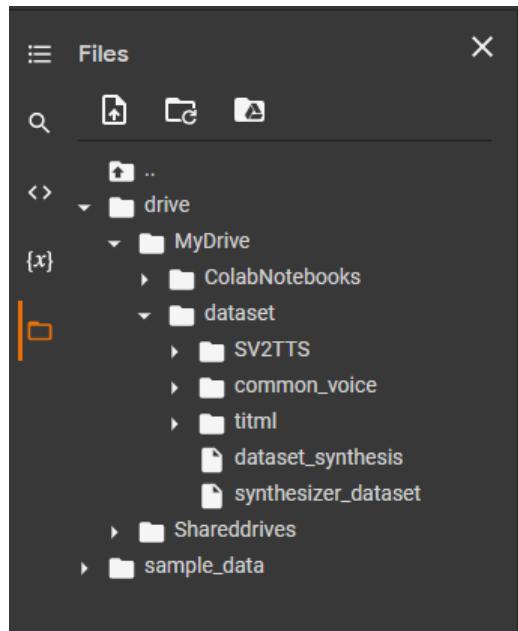
37     parser.add_argument("-s", "--skip_existing", action=""
38         store_true", help=\
39             "Whether to skip existing output files with the same name
40             . Useful if this script was "
41             "interrupted.")
42     parser.add_argument("--no_trim", action="store_true", help=\
43             "Preprocess audio without trimming silences (not
44             recommended).")
45     args = parser.parse_args()
46
47     # Verify webrtcvad is available
48     if not args.no_trim:
49         try:
50             import webrtcvad
51         except:
52             raise ModuleNotFoundError("Package 'webrtcvad' not
53             found. This package enables "
54             "noise removal and is recommended. Please install
55             and try again. If installation fails, "
56             "use --no_trim to disable this error message.")
57     del args.no_trim
58
59     # Process the arguments
60     args.datasets = args.datasets.split(",")
61     if not hasattr(args, "out_dir"):
62         args.out_dir = args.datasets_root.joinpath("SV2TTS", "encoder")
63     assert args.datasets_root.exists()
64     args.out_dir.mkdir(exist_ok=True, parents=True)
65
66     # Preprocess the datasets
67     print_args(args, parser)
68     preprocess_func = {
69         "titml": preprocess_speech_dataset,
70         "common_voice": preprocess_clean_dataset,
71     }
72     args = vars(args)
73     for dataset in args.pop("datasets"):
74         print("Preprocessing %s" % dataset)
75         preprocess_func[dataset](**args)

```

Listing 10.6 Preprocessing Encoder Model

Pada argument dataset masukkan nama dataset yang teman-teman gunakan, pada kode diatas saya menggunakan dataset titml dan common voice berbahasa indonesia.

11. Upload dataset teman-teman ke google drive dengan struktur direktori seperti gambar 10.16

**Gambar 10.16** Struktur Dataset

12. Tambahkan text dan ketikkan #Preprocessing encoder lalu tambahkan kode dan ketikkan kode program untuk menjalankan preprocessing speaker encoder model.

```
! python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder_preprocess.py /content/drive/MyDrive/dataset
```

Listing 10.7 Script to Run Preprocessing Speaker Encoder Model

```
[ ] ⏎ python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder_preprocess.py /content/drive/MyDrive/dataset
```

Gambar 10.17 Preprocessing Speaker Encoder Model

13. Buka file encoder_train.py lalu edit argument –model_dir, pada bagian default isikan path direktori tempat model encoder akan disimpan.

```
1 from utils.argutils import print_args
2 from encoder.train import train
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Trains the speaker encoder. You must have run encoder_preprocess.py first.",
10        formatter_class=argparse.ArgumentDefaultsHelpFormatter
11    )
12
13     parser.add_argument("run_id", type=str, help= \
14         "Name for this model instance. If a model state from the same run ID was previously "
15         "saved, the training will restart from there. Pass -f to overwrite saved states and "
16         "restart from scratch.")
17     parser.add_argument("clean_data_root", type=Path, help= \
18         "Path to the output directory of encoder_preprocess.py. If you left the default "
19         "output directory when preprocessing, it should be <datasets_root>/SV2TTS/encoder/.")
20     parser.add_argument("-m", "--models_dir", type=Path, default= \
21         "/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder/saved_models/", help=\
22         "Path to the output directory that will contain the saved model weights, as well as "
23         "backups of those weights and plots generated during training.")
24     parser.add_argument("-v", "--vis_every", type=int, default= \
25         10, help= \
26         "Number of steps between updates of the loss and the plots.")
27     parser.add_argument("-u", "--umap_every", type=int, default= \
28         100, help= \
29         "Number of steps between updates of the umap projection. Set to 0 to never update the "
30         "projections.")
31     parser.add_argument("-s", "--save_every", type=int, default= \
32         500, help= \
33         "Number of steps between updates of the model on the disk. Set to 0 to never save the "
34         "model.")
35     parser.add_argument("-b", "--backup_every", type=int, default= \
36         7500, help= \
37         "Number of steps between backups of the model. Set to 0 to never make backups of the "
38         "model.")
39     parser.add_argument("-f", "--force_restart", action="store_true", help= \
40         "Do not load any saved model.")
```

```

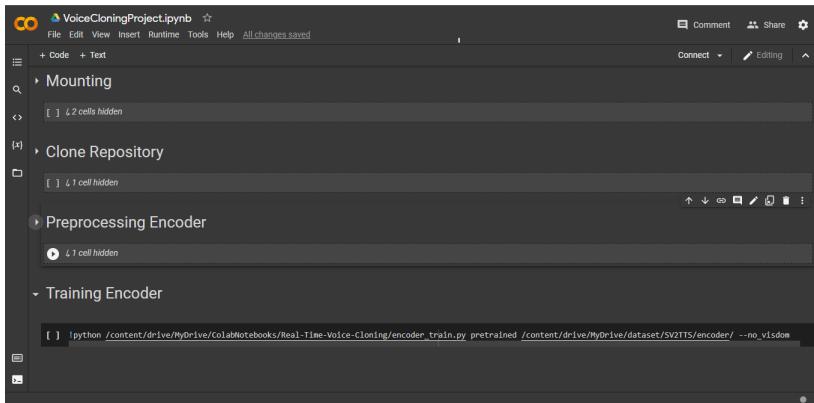
36     parser.add_argument("--visdom_server", type=str, default="http://localhost")
37     parser.add_argument("--no_visdom", action="store_true", help=
38         "\n        Disable visdom.")
39     args = parser.parse_args()
40
41     # Process the arguments
42     args.models_dir.mkdir(exist_ok=True)
43
44     # Run the training
45     print_args(args, parser)
46     train(**vars(args))

```

Listing 10.8 Training Speaker Encoder Model

14. Selanjutnya, klik tambahkan text dan ketikkan #Training encoder lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan training speaker encoder model. Tambahkan argument –no_visdom apabila teman-teman tidak menggunakan visdom server untuk menampilkan visualisasi dari hasil training speaker encoder model.

```
!python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder_train.py pretrained /content/drive/MyDrive/dataset/SV2TTS/encoder/ --no_visdom
```

Listing 10.9 Script to Run Training Speaker Encoder Model**Gambar 10.18** Training Speaker Encoder Model

15. Buka file synthesizer_preprocess_audio.py lalu edit argument –datasets_name, pada bagian default isikan nama folder dataset teman-teman dan argumen –subfolders pada default isikan nama sub folder dari folder dataset. Jika terdapat lebih dari satu subfolder maka pisahkan subfolder 1 dan 2 menggunakan koma.

```

1 from synthesizer.preprocess import preprocess_dataset
2 from synthesizer.hparams import hparams
3 from utils.argutils import print_args
4 from pathlib import Path

```

```
5 import argparse
6
7
8 if __name__ == "__main__":
9     parser = argparse.ArgumentParser(
10         description="Preprocesses audio files from datasets,
11             encodes them as mel spectrograms
12                 "and writes them to the disk. Audio files
13             are also saved, to be used by the "
14                 "vocoder for training.",
15             formatter_class=argparse.ArgumentDefaultsHelpFormatter
16     )
17     parser.add_argument("--datasets_root", type=Path, help=\
18         "Path to the directory containing your LibriSpeech/TTS
19             datasets.")
20     parser.add_argument("-o", "--out_dir", type=Path, default=
21         argparse.SUPPRESS, help=\
22             "Path to the output directory that will contain the mel
23             spectrograms, the audios and the "
24                 "embeds. Defaults to <datasets_root>/SV2TTS/synthesizer/")
25     parser.add_argument("-n", "--n_processes", type=int, default=
26         4, help=\
27             "Number of processes in parallel.")
28     parser.add_argument("-s", "--skip_existing", action="\
29         store_true", help=\
30             "Whether to overwrite existing files with the same name.
31             Useful if the preprocessing was "
32                 "interrupted.")
33     parser.add_argument("--hparams", type=str, default="", help=\
34             "Hyperparameter overrides as a comma-separated list of
35             name-value pairs")
36     parser.add_argument("--no_alignments", action="store_true",
37         help=\
38             "Use this option when dataset does not include alignments
39                 \
40                     (these are used to split long audio files into sub-
41                         utterances.)")
42     parser.add_argument("--datasets_name", type=str, default="\
43             synthesizer_dataset", help=\
44                 "Name of the dataset directory to process.")
45     parser.add_argument("--subfolders", type=str, default="cv-
46             corpus-7.0-2021-07-21", help=\
47                 "Comma-separated list of subfolders to process inside
48             your dataset directory")
49     args = parser.parse_args()
50
51     # Process the arguments
52     if not hasattr(args, "out_dir"):
53         args.out_dir = args.datasets_root.joinpath("SV2TTS", "synthesizer")
54
55     # Create directories
56     assert args.datasets_root.exists()
57     args.out_dir.mkdir(exist_ok=True, parents=True)
58
```

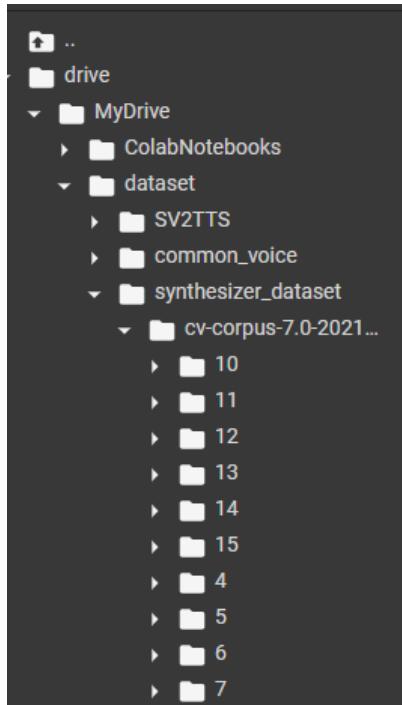
```

44 # Preprocess the dataset
45 print_args(args, parser)
46 args.hparams = hparams.parse(args.hparams)
47 preprocess_dataset(**vars(args))

```

Listing 10.10 Training Speaker Encoder Model

Berikut struktur folder dataset yang saya gunakan:

**Gambar 10.19** Struktur Folder Dataset Synthesizer

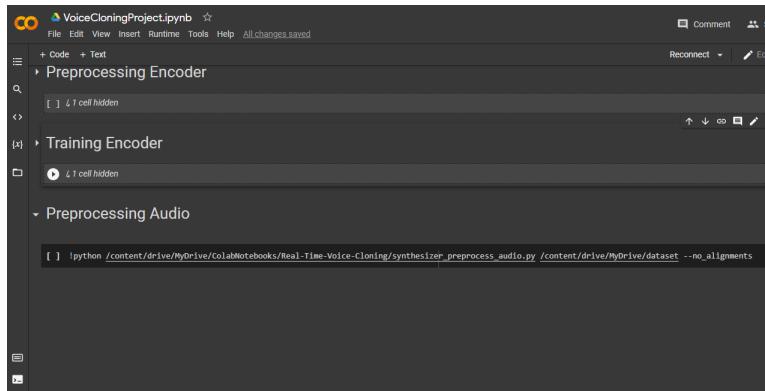
16. Selanjutnya, klik tambahkan text dan ketikkan `#Preprocessing Audio` lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan preprocessing audio. Tambahkan argumen `--no_alignments` jika teman-teman gunakan tidak memiliki alignments.

```

! python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer_preprocess_audio.py /content/drive/MyDrive/dataset --no_alignments

```

Listing 10.11 Preprocessing Audio



Gambar 10.20 Preprocessing Audio

17. Buka file synthesizer_preprocess_embed.py lalu edit pada bagian argument –encoder_model_fpath dan isikan lokasi penyimpanan model speaker encoder yang telah di training sebelumnya pada bagian defaults.

```

1 from synthesizer.preprocess import create_embeddings
2 from utils.argutils import print_args
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Creates embeddings for the synthesizer from
10        the LibriSpeech utterances.",
11        formatter_class=argparse.ArgumentDefaultsHelpFormatter
12    )
13    parser.add_argument("synthesizer_root", type=Path, help=\
14        "Path to the synthesizer training data that contains the
15        audios and the train.txt file."
16        "If you let everything as default, it should be <
17        datasets_root>/SV2TTS/synthesizer/")
18    parser.add_argument("-e", "--encoder_model_fpath", type=Path,
19        default="/content/drive/MyDrive/
20        ColabNotebooks/Real-Time-Voice-Cloning/encoder/saved_models/
21        pretrained.pt", help=\
22        "Path your trained encoder model")
23    parser.add_argument("-n", "--n_processes", type=int, default
24        =4, help= \
25        "Number of parallel processes. An encoder is created for
26        each, so you may need to lower "
27        "this value on GPUs with low memory. Set it to 1 if CUDA
28        is unhappy.")
29    args = parser.parse_args()
30
31    # Preprocess the dataset
32    print_args(args, parser)

```

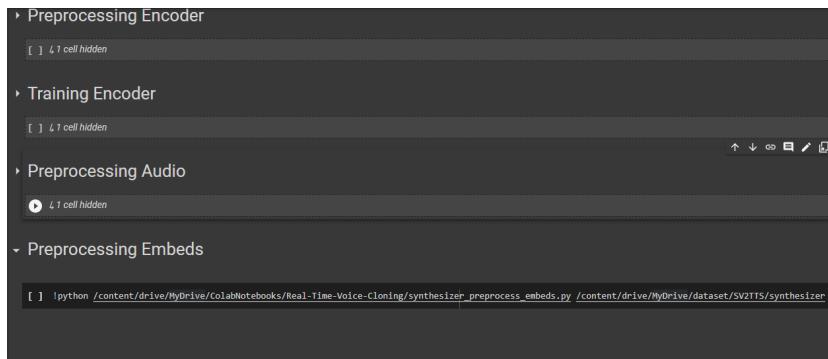
```
25 create_embeddings(**vars(args))
```

Listing 10.12 Training Speaker Encoder Model

18. Selanjutnya, klik tambahkan text dan ketikkan #Preprocessing Embeds lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan preprocessing audio embeddings.

```
1 !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer_preprocess_embeds.py /content/drive/MyDrive/dataset/SV2TTS/synthesizer
```

Listing 10.13 Preprocessing Embeds



Gambar 10.21 Preprocessing Audio Embeddings

19. Buka file synthesizer_train.py lalu edit argument –models_dir, pada bagian default isikan lokasi penyimpanan hasil training model synthesizer.

```
1 from pathlib import Path
2
3 from synthesizer.hparams import hparams
4 from synthesizer.train import train
5 from utils.argutils import print_args
6 import argparse
7
8
9 if __name__ == "__main__":
10     parser = argparse.ArgumentParser()
11     parser.add_argument("run_id", type=str, help= \
12         "Name for this model. By default, training outputs will \
13         be stored to saved_models/<run_id>/ . If a model state \
14         from the same run ID was previously saved, the training \
15         will restart from there. Pass -f to overwrite saved \
16         states and restart from scratch.")
17     parser.add_argument("syn_dir", type=Path, help= \
18         "Path to the synthesizer directory that contains the \
19         ground truth mel spectrograms, \
20         the wavs and the embeds.")
```

```

18 parser.add_argument("--m", "--models_dir", type=Path, default=\
19     "/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-\
20     Cloning/synthesizer/saved.models", help=\
21         "Path to the output directory that will contain the saved\
22         model weights and the logs.")
23 parser.add_argument("--s", "--save_every", type=int, default=\
24     1000, help=\
25         "Number of steps between updates of the model on the disk\
26         . Set to 0 to never save the"\
27             "model.")
28 parser.add_argument("--b", "--backup_every", type=int, default=\
29     25000, help=\
30         "Number of steps between backups of the model. Set to 0\
31         to never make backups of the"\
32             "model.")
33 parser.add_argument("-f", "--force_restart", action="\
34     store_true", help=\
35         "Do not load any saved model and restart from scratch.")
36 parser.add_argument("--hparams", default="", help=\
37         "Hyperparameter overrides as a comma-separated list of\
38         name=value pairs")
39 args = parser.parse_args()
40 print_args(args, parser)
41
42 args.hparams = hparams.parse(args.hparams)
43
44 # Run the training
45 train(**vars(args))

```

Listing 10.14 Training Speaker Encoder Model

20. Selanjutnya, klik tambahkan text dan ketikkan #Training Synthesizer Model lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan proses training synthesizer model.

```

1 !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-\
2     Cloning/synthesizer_train.py pretrained /content/drive/\
3         MyDrive/dataset/SV2TTS/synthesizer

```

Listing 10.15 Training Synthesizer Model

The screenshot shows a Google Colab notebook with the following structure:

- Training Encoder**: Contains a cell titled "L1 cell hidden".
- Preprocessing Audio**: Contains a cell titled "L1 cell hidden".
- Preprocessing Embeds**: Contains a cell titled "L1 cell hidden".
- Training Synthesizer Model**: Contains a cell with the command: `[] !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer_train.py pretrained /content/drive/MyDrive/dataset/SV2TTS/synthesizer`.

Gambar 10.22 Training Synthesizer Model

21. Buka file vocoder_preprocess.py lalu edit argument--syn_model_fpath, pada bagian default isikan lokasi penyimpanan hasil training model synthesizer.

```

1 import argparse
2 import os
3 from pathlib import Path
4
5 from synthesizer.hparams import hparams
6 from synthesizer.synthesize import run_synthesis
7 from utils.argutils import print_args
8
9
10 if __name__ == "__main__":
11     class MyFormatter(argparse.ArgumentDefaultsHelpFormatter,
12                         argparse.RawDescriptionHelpFormatter):
13         pass
14
15     parser = argparse.ArgumentParser(
16         description="Creates ground-truth aligned (GTA)
17         spectrograms from the vocoder.",
18         formatter_class=MyFormatter
19     )
20     parser.add_argument("datasets_root", type=Path, help=\
21         "Path to the directory containing your SV2TTS directory.
22         If you specify both --in_dir and "
23         "--out_dir, this argument won't be used.")
24     parser.add_argument("-s", "--syn_model_fpath", type=Path,
25                         default="/content/drive/MyDrive/
26                         ColabNotebooks/Real-Time-Voice-Cloning/synthesizer/
27                         saved_models/pretrained/synthesizer.pt",
28                         help="Path to a saved synthesizer")
29     parser.add_argument("-i", "--in_dir", type=Path, default=
30                         argparse.SUPPRESS, help= \
31         "Path to the synthesizer directory that contains the mel
32         spectrograms, the wavs and the "
33         "embeds. Defaults to <datasets_root>/SV2TTS/synthesizer
34         /.")
35     parser.add_argument("-o", "--out_dir", type=Path, default=
36                         argparse.SUPPRESS, help= \

```

```

28     "Path to the output vocoder directory that will contain
29     the ground truth aligned mel "
30     "spectrograms. Defaults to <datasets_root>/SV2TTS/vocoder
31     /.")
32     parser.add_argument("--hparams", default="", help=\
33         "Hyperparameter overrides as a comma-separated list of
34         name=value pairs")
35     parser.add_argument("--cpu", action="store_true", help=\
36         "If True, processing is done on CPU, even when a GPU is
37         available.")
38     args = parser.parse_args()
39     print_args(args, parser)
40     modified_hp = hparams.parse(args.hparams)

41     if not hasattr(args, "in_dir"):
42         args.in_dir = args.datasets_root / "SV2TTS" / "
43             synthesizer"
44     if not hasattr(args, "out_dir"):
45         args.out_dir = args.datasets_root / "SV2TTS" / "vocoder"
46
47     if args.cpu:
48         # Hide GPUs from Pytorch to force CPU processing
49         os.environ["CUDA_VISIBLE_DEVICES"] = "-1"

50     run_synthesis(args.in_dir, args.out_dir, args.syn_model_fpath
51     , modified_hp)

```

Listing 10.16 Vocoder Preprocess

22. Selanjutnya, klik tambahkan text dan ketikkan #Preprocessing Vocoder lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan proses preprocessing vocoder model.

```
! python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-
Cloning/vocoder_preprocess.py /content/drive/MyDrive/dataset
```

Listing 10.17 Preprocessing Vococder Model

+ Code + Text

- ▶ Preprocessing Embeds
- [] 41 cell hidden
- ▶ Training Synthesizer Model
- [] 41 cell hidden
- ▼ Preprocessing Vocoder

```
[ ] !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/vocoder_preprocess.py /content/drive/MyDrive/dataset
```

Gambar 10.23 Preprocessing Vocoder Model

23. Buka file `vocoder_train.py` lalu edit argument `--models_dir`, pada bagian default isikan lokasi untuk menyimpan hasil training vocoder model.

```

1 from utils.argutils import print_args
2 from vocoder.train import train
3 from pathlib import Path
4 import argparse
5
6
7 if __name__ == "__main__":
8     parser = argparse.ArgumentParser(
9         description="Trains the vocoder from the synthesizer
10        audios and the GTA synthesized mels, "
11        "or ground truth mels.",
12        formatter_class=argparse.ArgumentDefaultsHelpFormatter
13    )
14
15    parser.add_argument("--run_id", type=str, help= \
16        "Name for this model instance. If a model state from the
17        same run ID was previously"
18        "saved, the training will restart from there. Pass -f to
19        overwrite saved states and"
20        "restart from scratch.")
21    parser.add_argument("--datasets_root", type=str, help= \
22        "Path to the directory containing your SV2TTS directory.
23        Specifying --syn_dir or --voc_dir"
24        "will take priority over this argument.")
25    parser.add_argument("--syn_dir", type=str, default=argparse.
26        SUPPRESS, help= \
27        "Path to the synthesizer directory that contains the
28        ground truth mel spectrograms,"
29        "the wavs and the embeds. Defaults to <datasets_root>/
30        SV2TTS/synthesizer/.")
```

```

24     parser.add_argument("--voc_dir", type=str, default=argparse.SUPPRESS, help= \
25         "Path to the vocoder directory that contains the GTA synthesized mel spectrograms."
26         "Defaults to <datasets_root>/SV2TTS/vocoder/. Unused if --ground_truth is passed.")
27     parser.add_argument("-m", "--models_dir", type=str, default="/content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/vocoder/saved_models/", help= \
28         "Path to the directory that will contain the saved model weights, as well as backups"
29         "of those weights and wavs generated during training.")
30     parser.add_argument("-g", "--ground_truth", action="store_true", help= \
31         "Train on ground truth spectrograms (<datasets_root>/SV2TTS/synthesizer/mels).")
32     parser.add_argument("-s", "--save_every", type=int, default=1000, help= \
33         "Number of steps between updates of the model on the disk"
34         ". Set to 0 to never save the"
35         "model.")
36     parser.add_argument("-b", "--backup_every", type=int, default=25000, help= \
37         "Number of steps between backups of the model. Set to 0"
38         "to never make backups of the"
39         "model.")
40     parser.add_argument("-f", "--force_restart", action="store_true", help= \
41         "Do not load any saved model and restart from scratch.")
42     args = parser.parse_args()
43
44     # Process the arguments
45     if not hasattr(args, "syn_dir"):
46         args.syn_dir = Path(args.datasets_root, "SV2TTS", "synthesizer")
47     args.syn_dir = Path(args.syn_dir)
48     if not hasattr(args, "voc_dir"):
49         args.voc_dir = Path(args.datasets_root, "SV2TTS", "vocoder")
50     args.voc_dir = Path(args.voc_dir)
51     del args.datasets_root
52     args.models_dir = Path(args.models_dir)
53     args.models_dir.mkdir(exist_ok=True)
54
55     # Run the training
56     print_args(args, parser)
57     train(**vars(args))

```

Listing 10.18 Vocoder Train

24. Selanjutnya, klik tambahkan text dan ketikkan #Training Vocoder lalu klik tambahkan kode dan ketikkan kode program untuk menjalankan proses training vocoder model. Tambahkan argument –ground-truth agar proses training dilakukan menggunakan mel spektrogram yang telah di preprocessing sebelumnya pada model synthesizer.

```
! python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/vocoder_train.py pretrained /content/drive/MyDrive/dataset --ground-truth
```

Listing 10.19 Training Vococder Model

The screenshot shows a Google Colab notebook interface. The title bar says 'VoiceCloningProject.ipynb'. The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and 'All changes saved'. Below the menu is a toolbar with 'Comment' and 'Share' buttons. A code editor window displays a list of sections and a command cell:

- Preprocessing Embeds
- Training Synthesizer Model
- Preprocessing Vocoder
- Training Vocoder

In the 'Training Vocoder' section, there is a code cell containing the command: `[] ! python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/vocoder_train.py pretrained /content/drive/MyDrive/dataset --ground-truth`.

Gambar 10.24 *Training Vococder Model*

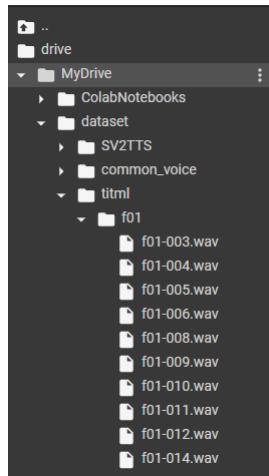
BAB 11

RESULT, ERROR, AND SOLUTION

11.1 Pembahasan Hasil dari Proses

Pada bab ini saya akan memaparkan dan menjelaskan hasil dari masing-masing proses yang telah kita jalankan baik menggunakan komputer pribadi maupun menggunakan google colab.

1. Proses pertama yaitu encoder preprocessing, pada proses ini input yang dibutuhkan yaitu file audio dataset dengan struktur folder seperti gambar 11.1. Pertama yaitu membuat folder dengan nama dataset, lalu buat folder dengan nama dataset yang digunakan, contoh pada gambar menggunakan dataset titml. Apabila folder dengan nama dataset telah dibuat, maka buat folder dengan nama speaker misalnya f01 (female01), f02 (female02) dan seterusnya untuk speaker dengan jenis kelamin perempuan. Berikan nama folder m01 (male), m02 (male02) dan seterusnya untuk speaker dengan jenis kelamin laki-laki. Isikan file audio sesuai dengan masing-masing folder. Masukkan file suara female01 ke folder f01 dan seterusnya. Hal ini dilakukan agar model dapat mengenali suara setiap speaker pada dataset dan memudahkan model dalam mempelajari suara masing-masing speaker, memetakan dan menampilkan hasilnya menggunakan library umap-learn.



Gambar 11.1 Struktur Folder Dataset untuk Preprocessing Speaker Encoder

Berikut hasil dari proses encoder preprocessing:

```
[ ] !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder_preprocess.py

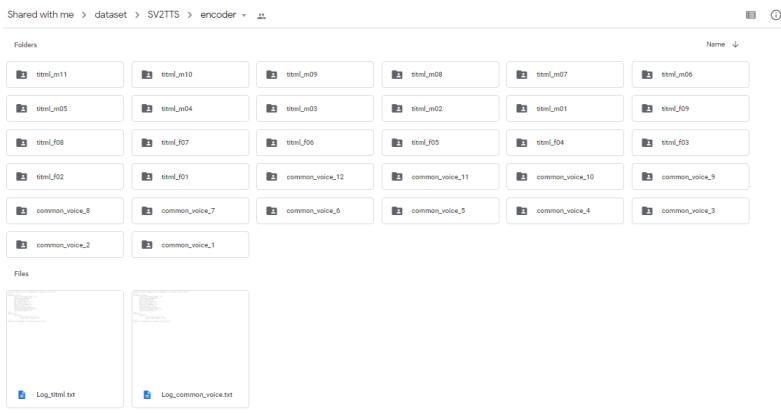
Arguments:
    datasets_root:  /content/drive/MyDrive/dataset
    out_dir:        /content/drive/MyDrive/dataset/SV2TTS/encoder
    datasets:       ['titml', 'common_voice']
    skip_existing: False

Preprocessing titml
titml: Preprocessing data for 20 speakers.
titml: 100% 20/20 [08:34<00:00, 25.74s/speakers]
Done preprocessing titml.

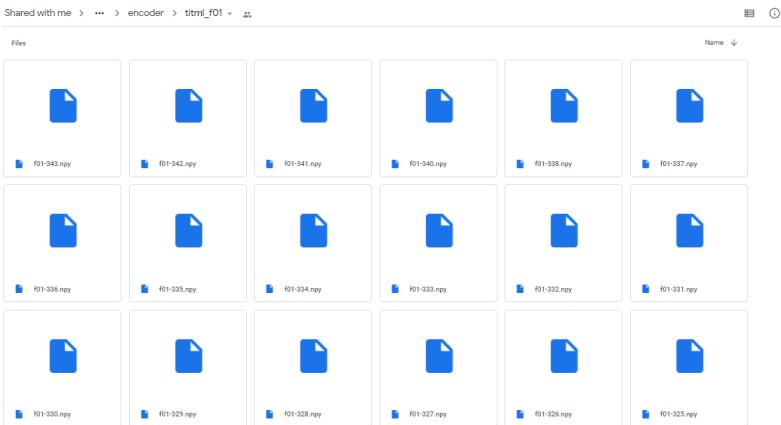
Preprocessing common_voice
common_voice: Preprocessing data for 12 speakers.
common_voice: 100% 12/12 [07:46<00:00, 38.85s/speakers]
Done preprocessing common_voice.
```

Gambar 11.2 Hasil Preprocessing Speaker Encoder

Periksa folder dataset lalu buka folder dengan nama SV2TTS, didalam folder tersebut teman-teman akan menemukan folder dengan nama encoder yang didalamnya terdapat hasil preprocessing encoder. Perhatikan gambar 11.3 terdapat 32 folder yang mewakiliakan masing-masing speaker, didalam folder tersebut terdapat numpy array hasil pre-process dari dataset titml dan common_voice yang saya gunakan.



Gambar 11.3 Folder Hasil Preprocessing Speaker Encoder



Gambar 11.4 File Numpy Array Hasil Preprocessing Speaker Encoder

Berikut log hasil preprocessing encoder menggunakan dataset titml dan common_voice:

```

Creating dataset titml on Wednesday 12 January 2022 at 03:37
-----
Parameter values:
    audio_norm_target_dBFS: -30
    inference_n_frames: 80
    mel_n_channels: 40
    mel_window_length: 25
    mel_window_step: 10
    partials_n_frames: 160
    sampling_rate: 16000
    vad_max_silence_length: 6
    vad_moving_average_width: 8
    vad_window_length: 30
-----
Statistics:
    duration:
        min 1.590, max 20.790
        mean 6.764, median 6.570
-----
Finished on Wednesday 12 January 2022 at 03:46

```

Gambar 11.5 Log TITML-IDN

```

Creating dataset common_voice on Wednesday 12 January 2022 at 03:46
-----
Parameter values:
    audio_norm_target_dBFS: -30
    inference_n_frames: 80
    mel_n_channels: 40
    mel_window_length: 25
    mel_window_step: 10
    partials_n_frames: 160
    sampling_rate: 16000
    vad_max_silence_length: 6
    vad_moving_average_width: 8
    vad_window_length: 30
-----
Statistics:
    duration:
        min 1.590, max 22.740
        mean 3.931, median 3.150
-----
Finished on Wednesday 12 January 2022 at 03:53

```

Gambar 11.6 Log Common Voice Indonesia

2. Proses training speaker encoder model membutuhkan hasil preprocessing encoder yaitu file numpy array dari audio dataset yang telah di preprocess. Perhatikan gambar 11.7 saya melakukan proses training selama dua minggu menggunakan google colab, saya memperoleh hasil model dengan loss sekitar 0,7 dan error rate sekitar 0.015. Proses training speaker encoder model bisa kita lihat menggunakan visdom server atau teman-teman bisa melihat hasilnya dalam bentuk gambar yang tersimpan di folder pretrained_backups yang ada di folder Real-Time-Voice-Cloning, masuk ke folder encoder, lalu ke folder saved_models seperti pada gambar 11.9

Training Encoder

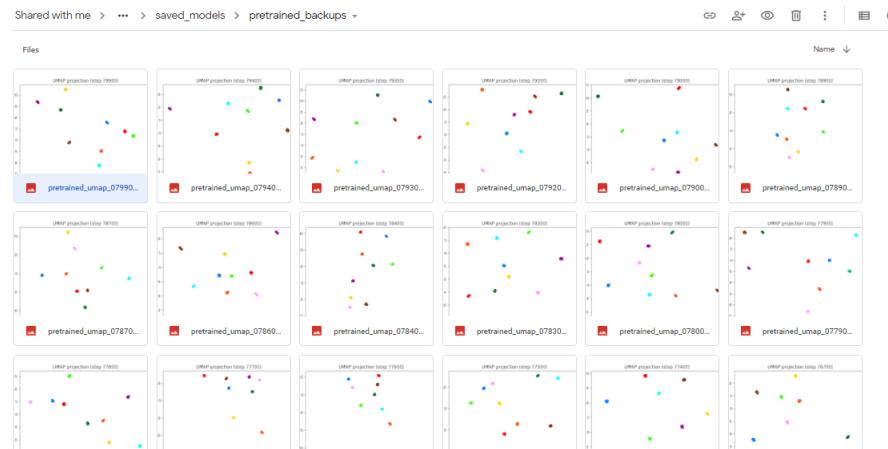
```
!python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/encoder_train.py pretrained /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/pretrained

Streaming output truncated to the last 5000 lines.
Average execution time over 10 steps:
  Blocking, waiting for batch (threaded) (10/10): mean: 556ms std: 1610ms
  Data to cuda (10/10): mean: 4ms std: 1ms
  Forward pass (10/10): mean: 158ms std: 6ms
  Loss (10/10): mean: 98ms std: 35ms
  Backward pass (10/10): mean: 198ms std: 28ms
  Parameter update (10/10): mean: 291ms std: 33ms
  Extras (visualizations, saving) (10/10): mean: 0ms std: 0ms

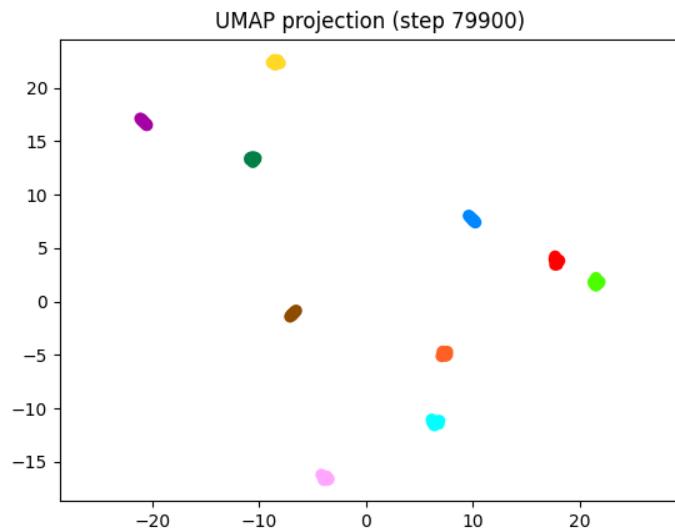
.....
Step 75860  Loss: 0.7039  EER: 0.0157  Step time: mean: 1269ms std: 1447ms

Average execution time over 10 steps:
  Blocking, waiting for batch (threaded) (10/10): mean: 504ms std: 1452ms
  Data to cuda (10/10): mean: 5ms std: 2ms
  Forward pass (10/10): mean: 156ms std: 3ms
  Loss (10/10): mean: 108ms std: 39ms
  Backward pass (10/10): mean: 194ms std: 40ms
  Parameter update (10/10): mean: 303ms std: 14ms
  Extras (visualizations, saving) (10/10): mean: 0ms std: 1ms
```

Gambar 11.7 Hasil Training Speaker Encoder Model

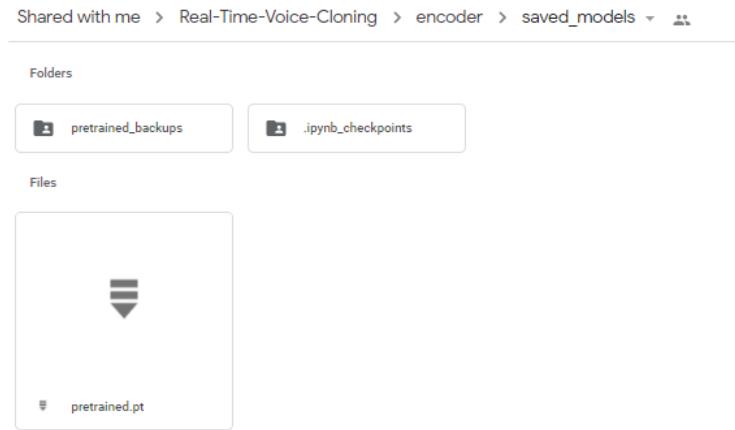


Gambar 11.8 File Hasil Training Speaker Encoder Model



Gambar 11.9 UMAP Projection in 79900 steps

Berikut hasil dari model yang telah di training dan di save dengan nama file pretrained.pt, file ini akan teman-teman temukan pada folder saved_models.



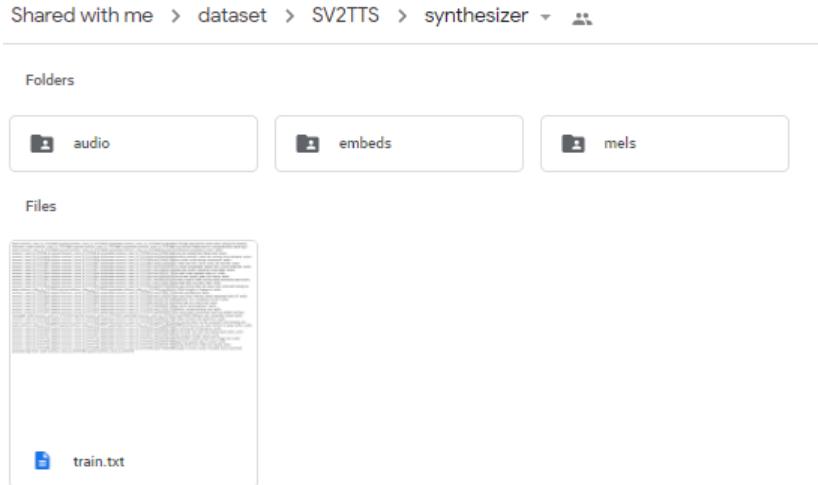
Gambar 11.10 File Speaker Encoder Model

3. Proses selanjutnya yaitu preprocessing audio, pada tahap ini kita membutuhkan file audio dan file text pada dataset. Dataset yang saya gunakan pada proses ini yaitu common voice indonesia, berikut struktur folder dataset untuk menjalankan preprocessing audio.

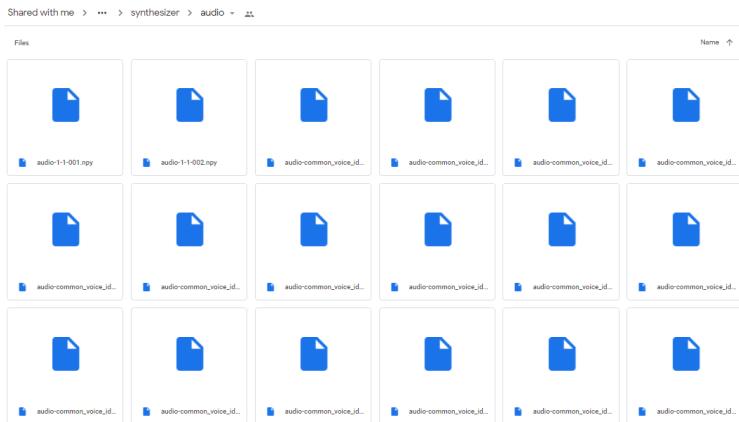
```
datasets_root
    * LibriTTS
        * train-clean-100
            * speaker-001
                * book-001
                    * utterance-001.wav
                    * utterance-001.txt
                    * utterance-002.wav
                    * utterance-002.txt
                    * utterance-003.wav
                    * utterance-003.txt
```

Gambar 11.11 Struktur Dataset untuk Preprocessing Audio dan Embeds

Proses ini menghasilkan file dengan format numpy array pada folder audio dan mels seperti pada gambar 11.12. Selain itu proses ini juga menghasilkan file train.txt yang didalamnya berisi data file name numpy array audio, mels, embeds, serta text dari file audio dataset.



Gambar 11.12 Hasil Preprocessing Audio



Gambar 11.13 File Numpy Array dari Salah Satu Folder

```

audio-common_voice_id_27373820.npy|mel-common_voice_id_27373820.npy|63360|317|Tidak ada hierarki sosial dalam penggunaan bahasa Indonesia.
audio-common_voice_id_27373845.npy|mel-common_voice_id_27373845.npy|embed-common_voice_id_27373845.npy|35040|176|Maukah kau mengatakannya sekali lagi?
audio-common_voice_id_27373849.npy|mel-common_voice_id_27373849.npy|embed-common_voice_id_27373849.npy|32160|161|Semua tergantung cuaca.
audio-common_voice_id_27373816.npy|mel-common_voice_id_27373816.npy|embed-common_voice_id_27373816.npy|37920|190|Kucing itu sedang tidur diatas sofa.
audio-common_voice_id_27373829.npy|mel-common_voice_id_27373829.npy|embed-common_voice_id_27373829.npy|59040|296|Namanya sahabat, susah dan senang harus bersama.
audio-common_voice_id_27373828.npy|mel-common_voice_id_27373828.npy|embed-common_voice_id_27373828.npy|31600|159|Kami sudah mudah dengan keluhanmu.
audio-common_voice_id_27373844.npy|mel-common_voice_id_27373844.npy|embed-common_voice_id_27373844.npy|48600|219|Kamu punya banyak teman di luar sana.
audio-common_voice_id_27373847.npy|mel-common_voice_id_27373847.npy|embed-common_voice_id_27373847.npy|37920|248|Tom dapat mengatakan bahwa Mary cinta sang kerakasian.
audio-common_voice_id_27373827.npy|mel-common_voice_id_27373827.npy|embed-common_voice_id_27373827.npy|48600|219|Salah satu bahan membuat bunga palsu.
audio-common_voice_id_27373841.npy|mel-common_voice_id_27373841.npy|embed-common_voice_id_27373841.npy|37920|248|Kamu punya banyak teman di luar sana.
audio-common_voice_id_27373844.npy|mel-common_voice_id_27373844.npy|embed-common_voice_id_27373844.npy|48600|205|Dua bersahabat sejati gelar ahli madu.
audio-common_voice_id_27373831.npy|mel-common_voice_id_27373831.npy|embed-common_voice_id_27373831.npy|37920|248|Kamu punya banyak teman di luar sana.
audio-common_voice_id_27373832.npy|mel-common_voice_id_27373832.npy|embed-common_voice_id_27373832.npy|31600|159|Apa tidak ada yang lebih baik?
audio-common_voice_id_27373833.npy|mel-common_voice_id_27373833.npy|embed-common_voice_id_27373833.npy|37920|248|Apa tidak ada yang lebih baik?
audio-common_voice_id_27373845.npy|mel-common_voice_id_27373845.npy|embed-common_voice_id_27373845.npy|48600|219|Apa yang selang kalau dia pergi untuk melunasi hutangnya?
audio-common_voice_id_27373846.npy|mel-common_voice_id_27373846.npy|embed-common_voice_id_27373846.npy|37920|248|Bin tinggal di Singapura.
audio-common_voice_id_27373847.npy|mel-common_voice_id_27373847.npy|embed-common_voice_id_27373847.npy|41280|207|Apa yang kamu lakukan dalam beberapa bulan ini?
audio-common_voice_id_27373853.npy|mel-common_voice_id_27373853.npy|embed-common_voice_id_27373853.npy|37920|154|Bayangan Tom mengintai hal itu?
audio-common_voice_id_27373854.npy|mel-common_voice_id_27373854.npy|embed-common_voice_id_27373854.npy|37920|154|Kamu mendengarnya?
audio-common_voice_id_27373855.npy|mel-common_voice_id_27373855.npy|embed-common_voice_id_27373855.npy|37920|154|Kamu mendengarnya?
audio-common_voice_id_27373856.npy|mel-common_voice_id_27373856.npy|embed-common_voice_id_27373856.npy|37920|154|Kamu mendengarnya?
audio-common_voice_id_27373857.npy|mel-common_voice_id_27373857.npy|embed-common_voice_id_27373857.npy|37920|154|Kamu mendengarnya?
audio-common_voice_id_27374024.npy|mel-common_voice_id_27374024.npy|embed-common_voice_id_27374024.npy|38600|231|Dia merutuskan perencanaan ayahnya setelah ayahnya meninggal.
audio-common_voice_id_27374034.npy|mel-common_voice_id_27374034.npy|embed-common_voice_id_27374034.npy|37920|159|Apa suka mengopleksi segitu
audio-common_voice_id_27374028.npy|mel-common_voice_id_27374028.npy|embed-common_voice_id_27374028.npy|38600|176|Deket rumahku ada apartemen.
audio-common_voice_id_27374035.npy|mel-common_voice_id_27374035.npy|embed-common_voice_id_27374035.npy|4740|274|Ande bekerja dalam rumahku yang lebih baik daripada eks.
audio-common_voice_id_27374027.npy|mel-common_voice_id_27374027.npy|embed-common_voice_id_27374027.npy|38600|159|Semua itu akan akan berlalu ke pulau seribu.
audio-common_voice_id_27374032.npy|mel-common_voice_id_27374032.npy|embed-common_voice_id_27374032.npy|38600|159|Semua itu akan akan berlalu ke pulau seribu.
audio-common_voice_id_27374033.npy|mel-common_voice_id_27374033.npy|embed-common_voice_id_27374033.npy|49400|248|Varuna badai, dia tidak bisa datang dengan waktu.
audio-common_voice_id_27374140.npy|mel-common_voice_id_27374140.npy|embed-common_voice_id_27374140.npy|49400|248|Abdullah, bayaku sedih banget lauu.
audio-common_voice_id_27374141.npy|mel-common_voice_id_27374141.npy|embed-common_voice_id_27374141.npy|49400|248|Abdullah, bayaku sedih banget lauu.
audio-common_voice_id_27374142.npy|mel-common_voice_id_27374142.npy|embed-common_voice_id_27374142.npy|49400|248|Bayak Bodu apakah wkt di rumah.
audio-common_voice_id_27374056.npy|mel-common_voice_id_27374056.npy|embed-common_voice_id_27374056.npy|1800|158|Rumah ini terlalu sempit. Mustahil untuk menanam perkebunan.
audio-common_voice_id_27374166.npy|mel-common_voice_id_27374166.npy|embed-common_voice_id_27374166.npy|48000|241|Aku tidak menginginkan keseklasikan itu terdiri.
audio-common_voice_id_27374131.npy|mel-common_voice_id_27374131.npy|embed-common_voice_id_27374131.npy|37920|159|Tom memerlukan air dingin ke dia sendiri.
audio-common_voice_id_27374123.npy|mel-common_voice_id_27374123.npy|embed-common_voice_id_27374123.npy|37920|159|Tom terlalu lelah untuk berjalan.
audio-common_voice_id_27374103.npy|mel-common_voice_id_27374103.npy|embed-common_voice_id_27374103.npy|49400|248|Selokan ada di dekat gunung itu.
audio-common_voice_id_27374117.npy|mel-common_voice_id_27374117.npy|embed-common_voice_id_27374117.npy|38600|159|Selokan ada di dekat gunung itu.
audio-common_voice_id_27374117.npy|mel-common_voice_id_27374117.npy|embed-common_voice_id_27374117.npy|38600|158|Itu karena kamu tidak ingin sendiri.
audio-common_voice_id_27374151.npy|mel-common_voice_id_27374151.npy|embed-common_voice_id_27374151.npy|37920|159|Dulu teman ramai, sekarang sendiri.
audio-common_voice_id_27374090.npy|mel-common_voice_id_27374090.npy|embed-common_voice_id_27374090.npy|37920|159|Universitas ini tua dan terkenal.

```

Gambar 11.14 Data Pada File train.txt

1. `audio-common_voice_id_27373820.npy | mel-common_voice_id_27373820.npy | embed-common_voice_id_27373820.npy | 63360 | 317 | Tidak ada hierarki sosial dalam penggunaan bahasa Indonesia.`
2. `audio-common_voice_id_27373866.npy | mel-common_voice_id_27373866.npy | 35040 | 176 | Maukah kau mengatakannya sekali lagi?`
3. `audio-common_voice_id_27373849.npy | mel-common_voice_id_27373849.npy | embed-common_voice_id_27373849.npy | 32160 | 161 | Semua tergantung cuaca.`
4. `audio-common_voice_id_27373816.npy | mel-common_voice_id_27373816.npy | embed-common_voice_id_27373816.npy | 37920 | 190 | Kucing itu sedang tidur diatas sofa.`
5. `audio-common_voice_id_27373829.npy | mel-common_voice_id_27373829.npy | embed-common_voice_id_27373829.npy | 59040 | 296 | Namanya sahabat, susah dan senang harus bersama .`

```

6 audio-common_voice_id_27373828.npy|mel-common_voice_id_27373828.npy|embed-common_voice_id_27373828.npy|31680|159|Kami sudah  
muak dengan keluhanmu.  

7 audio-common_voice_id_27373867.npy|mel-common_voice_id_27373867.npy|embed-common_voice_id_27373867.npy|61440|308|Di kulkas  
ada telur, sayur, buah, dan lain-lain.  

8 audio-common_voice_id_27373847.npy|mel-common_voice_id_27373847.npy|embed-common_voice_id_27373847.npy|52800|265|Tom dapat  
mengatakan bahwa Mary cukup ketakutan.  

9 audio-common_voice_id_27373827.npy|mel-common_voice_id_27373827.npy|embed-common_voice_id_27373827.npy|43680|219|Salah satu  
hobiku membuat bunga palsu.  

10 audio-common_voice_id_27373848.npy|mel-common_voice_id_27373848.npy|embed-common_voice_id_27373848.npy|35520|178|Dia tidak  
bicara sepatah kata pun.  

11 audio-common_voice_id_27373844.npy|mel-common_voice_id_27373844.npy|embed-common_voice_id_27373844.npy|40800|205|Dia berhasil  
meraih gelar ahli madya.  

12 audio-common_voice_id_27373851.npy|mel-common_voice_id_27373851.npy|embed-common_voice_id_27373851.npy|56640|284|Sehabis  
maghrib akan dimulai rapat koordinasi acara.  

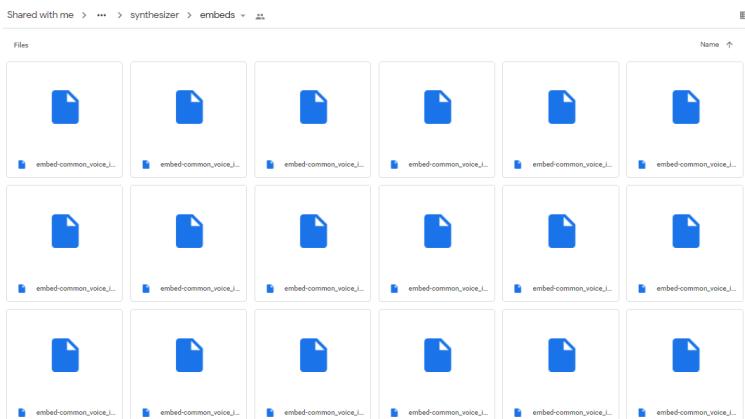
13 audio-common_voice_id_27373817.npy|mel-common_voice_id_27373817.npy|embed-common_voice_id_27373817.npy|31680|159|Apa tidak  
ada yang lebih baik?  

14 audio-common_voice_id_27373825.npy|mel-common_voice_id_27373825.npy|embed-common_voice_id_27373825.npy|53760|269|Ada yang  
bilang kalau dia pergi untuk melunasi hutangnya.

```

Listing 11.1 Beberapa Data Pada File train.txt

4. Preprocessing Embeds menghasilkan numpy array pada folder embeds yang digenerate berdasarkan pada file train.txt. Berikut hasil preprocessing embeds:



Gambar 11.15 File Numpy Array pada Folder embeds

5. Proses Training Synthesizer membutuhkan hasil dari preprocessing audio dan preprocessing embeds. Proses training synthesizer menghasilkan prediksi mel-spektrogram terdiri dari 4 folder seperti pada gambar 11.17.

```

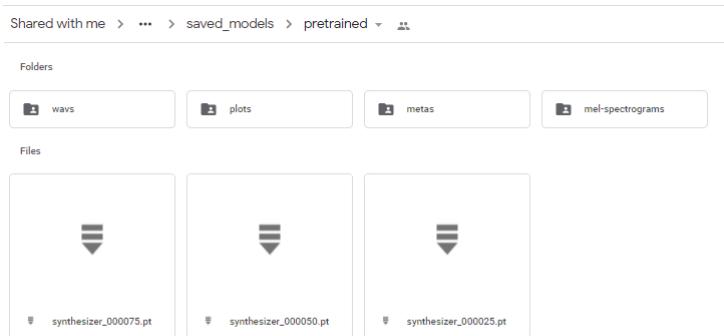
Checkpoint path: /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer/saved_models/pretrained/synthesizer
Loading training data from: /content/drive/MyDrive/dataset/SV2TTS/synthesizer/train.txt
Using model: Tacotron
Using device: cuda

Initialising Tacotron Model...
Trainable Parameters: 30.870M

Loading weights at /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/synthesizer/saved_models/pretrained/synthesizer
Tacotron weights loaded from step 79000
Using Inputs from:
    /content/drive/MyDrive/dataset/SV2TTS/synthesizer/train.txt
    /content/drive/MyDrive/dataset/SV2TTS/synthesizer/mels
    /content/drive/MyDrive/dataset/SV2TTS/synthesizer/embeds
Found 2039 samples
+-----+-----+-----+-----+
| Steps with r=2 | Batch Size | Learning Rate | Outputs/Step (r) |
+-----+-----+-----+-----+
| 1k Steps | 12 | 0.0002 | 2 |
+-----+-----+-----+-----+
{| Epoch: 1/6 (170/170) | Loss: 0.9844 | 0.25 steps/s | Step: 79k | }
{| Epoch: 2/6 (170/170) | Loss: 0.9833 | 0.25 steps/s | Step: 79k | }
{| Epoch: 3/6 (160/170) | Loss: 0.9818 | 0.25 steps/s | Step: 79k | }Input at step 79500: tom terjatuh ke dalam kolam.~.
{| Epoch: 3/6 (170/170) | Loss: 0.9824 | 0.25 steps/s | Step: 79k | }
{| Epoch: 4/6 (170/170) | Loss: 0.9827 | 0.25 steps/s | Step: 79k | }
{| Epoch: 5/6 (170/170) | Loss: 0.9828 | 0.25 steps/s | Step: 79k | }
{| Epoch: 6/6 (150/170) | Loss: 0.9844 | 0.27 steps/s | Step: 80k | }Input at step 80000: saya mengembalikan buku ke perpus.

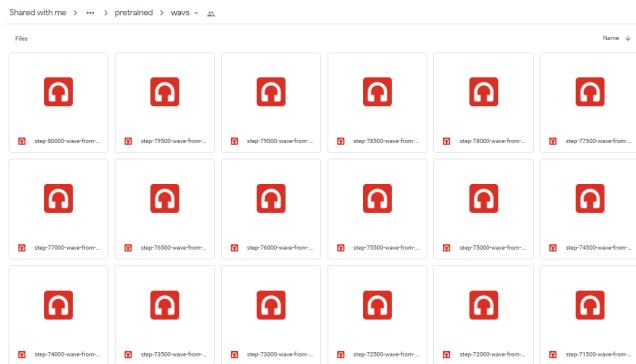
```

Gambar 11.16 Hasil Proses Training Synthesizer



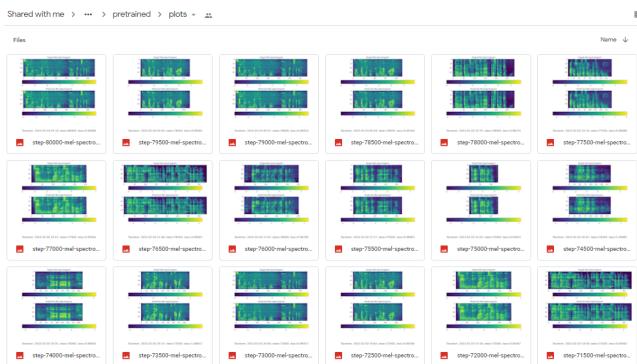
Gambar 11.17 Folder Hasil Proses Training Synthesizer

Folder wav berisikan file audio prediksi berdasarkan mel-spektrogram seperti pada gambar 11.18.



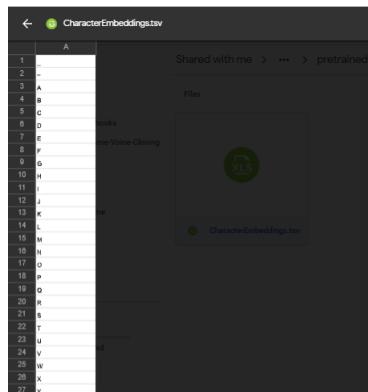
Gambar 11.18 *Folder Wav*

Folder plots berisikan visualisasi dari mel-spektrogram seperti pada gambar 11.19



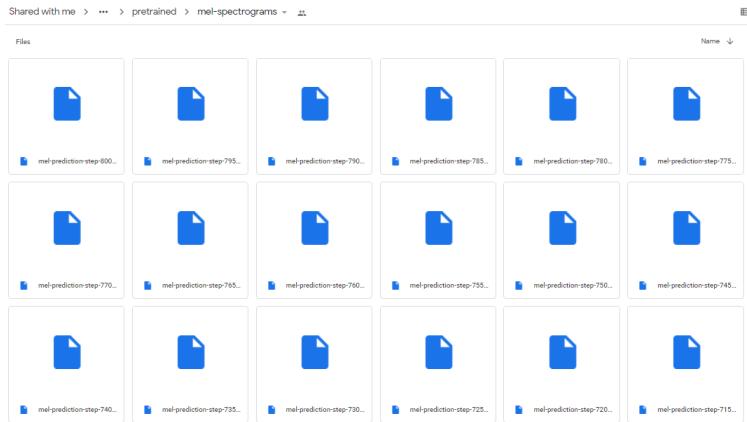
Gambar 11.19 *Folder Plots*

Folder metas berisikan file CharacterEmbeddings.tsv mulai dari a-z hingga angka 0-9 seperti pada gambar 11.20.



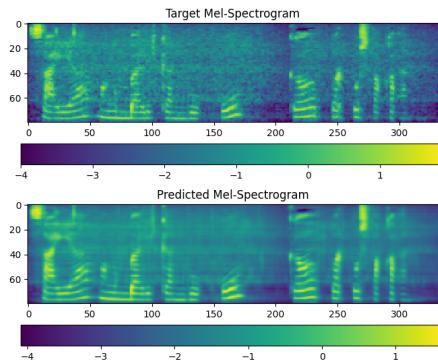
Gambar 11.20 File *CharacterEmbeddings.tsv*

Folder mel-spektrogram berisikan file numpy array dari mel-spektrogram seperti pada gambar 11.21.



Gambar 11.21 Folder *Mel-Spectrogram*

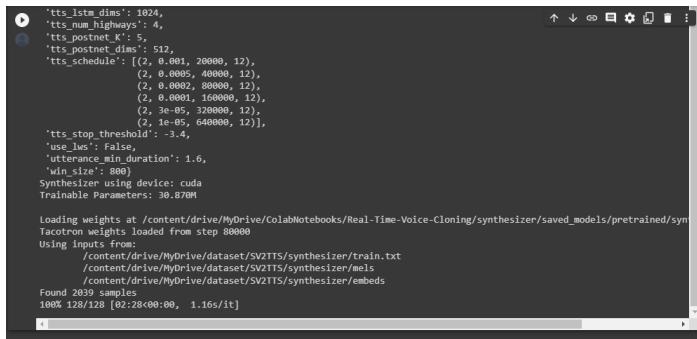
Berikut visualisasi target mel-spectrogram dan predicted mel-spectrogram hasil training synthesizer dengan nilai loss 0.9568.



Tacotron, 2022-02-03 03:19, step=80000, loss=0.95688

Gambar 11.22 *Mel-Spectrogram*

6. Preprocessing vocoder membutuhkan hasil dari proses training synthesizer model, file train.txt, serta membutuhkan file numpy array mels dan embeds.



```
tts_lstm_dim': 1024,
'tts_num_highways': 4,
'tts_postnet_k': 5,
'tts_postnet_dims': [512, 512, 512, 512, 512],
'tts_schedule': [(0, 0.001, 20000, 12),
(0, 0.0005, 40000, 12),
(0, 0.0002, 80000, 12),
(0, 0.0001, 160000, 12),
(2, 3e-05, 320000, 12),
(2, 1e-05, 640000, 12)],
'tts_stop_threshold': -3.4,
'use_lws': False,
'duration_stretch': 1.6,
'win_size': 800
Synthesizer using device: cuda
Trainable Parameters: 30.870M

Loading weights at /content/drive/MyDrive/colabNotebooks/Real-Time-Voice-Cloning/synthesizer/saved_models/pretrained/syn
Tacotron weights loaded from step 80000
Using Input: /content/drive/MyDrive/dataset/SV2TTS/synthesizer/train.txt
/content/drive/MyDrive/dataset/SV2TTS/synthesizer/mels
/content/drive/MyDrive/dataset/SV2TTS/synthesizer/embeds
Found 2039 samples
100% 128/128 [02:28<00:00, 1.16s/it]
```

Gambar 11.23 *Preprocessing Vocoder*

Proses ini menghasilkan file synthesized.txt dan folder mels-gta yang berisikan file target mel-spectrogram dengan format file binary numpy array yang akan digunakan dalam proses training synthesizer model.

Dibagikan kepada saya > dataset > SV2TTS > vocoder >

Folder



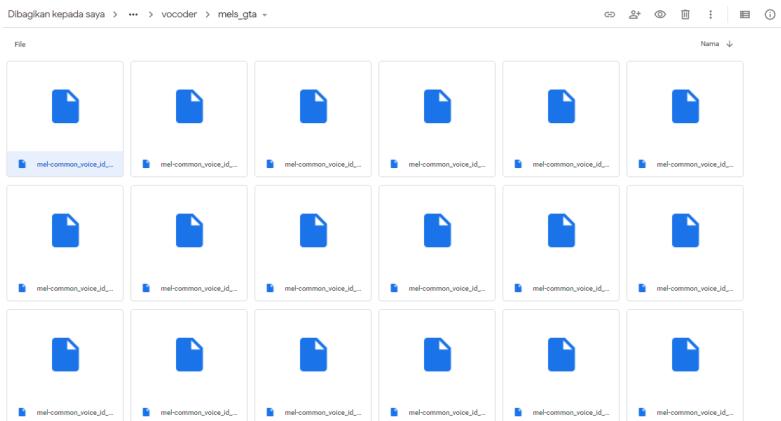
mels_gta

File



synthesized.txt

Gambar 11.24 Hasil Preprocessing Vocoder



Gambar 11.25 Folder mels-gta

File synthesized.txt berisikan nama file audio, nama file mel-spectrogram, nama file em-beds, dan teks.

```

audio-common_voice_id_27373820.npy|mel-common_voice_id_27373820.npy|embed-common_voice_id_27373820.npy|63360|317|Tidak ada
hierarki sosial dalam penggunaan bahasa Indonesia.
audio-common_voice_id_27373866.npy|mel-common_voice_id_27373866.npy|embed-common_voice_id_27373866.npy|35040|176|Meukah
kau mengatakannya sekali lagi?
audio-common_voice_id_27373849.npy|mel-common_voice_id_27373849.npy|embed-common_voice_id_27373849.npy|32160|161|Semua
tergenting cuaca.
audio-common_voice_id_27373816.npy|mel-common_voice_id_27373816.npy|embed-common_voice_id_27373816.npy|37920|190|Kucung
itu sedang tidak diajak sofa.
audio-common_voice_id_27373829.npy|mel-common_voice_id_27373829.npy|embed-common_voice_id_27373829.npy|59040|1296|Namanya
sakitnya dia takut dan takut bersama dengan teman.
audio-common_voice_id_27373828.npy|mel-common_voice_id_27373828.npy|embed-common_voice_id_27373828.npy|31680|159|Kami
sudah muak dengan Keluhanmu.
audio-common_voice_id_27373867.npy|mel-common_voice_id_27373867.npy|embed-common_voice_id_27373867.npy|61440|308|Dia kulkas
ada telur, sayur, buah dan lain-lain.
audio-common_voice_id_27373847.npy|mel-common_voice_id_27373847.npy|embed-common_voice_id_27373847.npy|52800|1265|Tom dapat
mengatakan bahwa Mary cukup ketakutan.
audio-common_voice_id_27373827.npy|mel-common_voice_id_27373827.npy|embed-common_voice_id_27373827.npy|43680|1219|salah
satu hobi mereka membuat bunga palsu.
audio-common_voice_id_27373848.npy|mel-common_voice_id_27373848.npy|embed-common_voice_id_27373848.npy|35520|178|Dia tidak
berikan kabar kepadaku.
audio-common_voice_id_27373844.npy|mel-common_voice_id_27373844.npy|embed-common_voice_id_27373844.npy|40800|205|Dia
berhasil meraih gelar ahli madya.
audio-common_voice_id_27373851.npy|mel-common_voice_id_27373851.npy|embed-common_voice_id_27373851.npy|56640|1284|Sehabis
maghrab akan dimulai, Rajah Koodinai selalu
audio-common_voice_id_27373821.npy|mel-common_voice_id_27373821.npy|embed-common_voice_id_27373821.npy|31680|159|Apa tidak
ada yang lebih baik?
audio-common_voice_id_27373825.npy|mel-common_voice_id_27373825.npy|embed-common_voice_id_27373825.npy|53760|1269|Ada yang
bilang kalau dia pergi untuk melunasi hutangnya.
audio-common_voice_id_27373845.npy|mel-common_voice_id_27373845.npy|embed-common_voice_id_27373845.npy|26400|1333|Buu
tinggi di Singapura.
audio-common_voice_id_27373832.npy|mel-common_voice_id_27373832.npy|embed-common_voice_id_27373832.npy|27360|137|Nikmati
pekerjaannya!
audio-common_voice_id_27373846.npy|mel-common_voice_id_27373846.npy|embed-common_voice_id_27373846.npy|41280|1207|Apa yang
kamu lakukan dalam masa liburan ini?
audio-common_voice_id_27373853.npy|mel-common_voice_id_27373853.npy|embed-common_voice_id_27373853.npy|30720|154|Begini
sama menantikannya ha?

```

Gambar 11.26 File Synthesized

7. Proses training vocoder membutuhkan prediksi mel, embeds, target mel, dan file train.txt. Proses ini akan menghasilkan file prediksi audio dari prediksi mel-spectrogram dan menghasilkan target audio dari target mel-spectrogram. Proses generate audio bisa dilihat pada gambar 11.27.

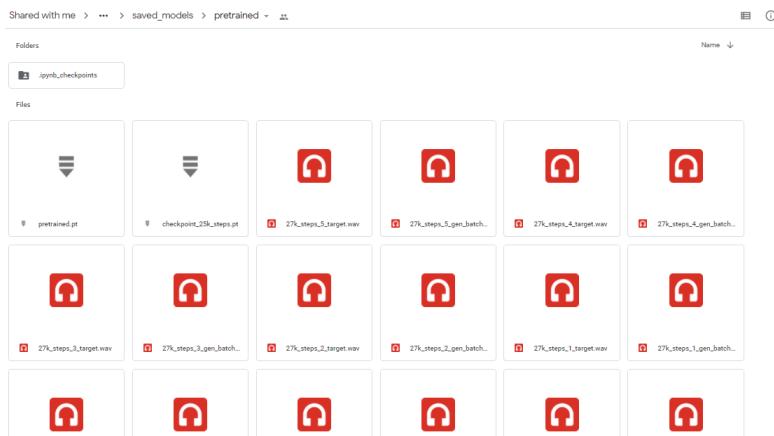
```

In [1]: !python /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/vocoder_train.py pretrained /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/weights/pretrained_vocoders.hdf5 -f /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/foldernamae_folds.json -t /content/drive/MyDrive/ColabNotebooks/Real-Time-Voice-Cloning/foldernamae_train.txt --n_gpus 1 --batch_size 8 --n_workers 8 --display_step 1000 --n_epochs 50000
    
```

{	43500/44000	Batch Size: 5	Gen Rate: 3.4kHz
Epoch: 199 (21/21)	Loss: 3.7537	0.4 steps/s	Step: 27K
Generating: 1/5			
{	52200/52800	Batch Size: 6	Gen Rate: 4.1kHz
Generating: 2/5			
{	52200/52800	Batch Size: 6	Gen Rate: 4.1kHz
Generating: 3/5			
{	43500/44000	Batch Size: 5	Gen Rate: 3.4kHz
Generating: 4/5			
{	121800/123200	Batch Size: 14	Gen Rate: 9.8kHz
Generating: 5/5			
{	60900/61600	Batch Size: 7	Gen Rate: 5.0kHz
Epoch: 200 (21/21)	Loss: 3.7330	0.4 steps/s	Step: 27K
Generating: 1/5			
{	34800/35200	Batch Size: 4	Gen Rate: 2.8kHz
Generating: 2/5			
{	87000/88000	Batch Size: 10	Gen Rate: 7.0kHz
Generating: 3/5			
{	78300/79200	Batch Size: 9	Gen Rate: 6.3kHz
Generating: 4/5			
{	60900/61600	Batch Size: 7	Gen Rate: 4.9kHz
Generating: 5/5			
{	52200/52800	Batch Size: 6	Gen Rate: 4.2kHz
Epoch: 201 (21/21)	Loss: 3.7601	0.4 steps/s	Step: 27K
Generating: 1/5			
{	43500/44000	Batch Size: 5	Gen Rate: 3.5kHz
Generating: 2/5			
{	113100/114400	Batch Size: 12	Gen Rate: 9.2kHz

Gambar 11.27 Training Vocoder

Hasil generate prediksi audio dan target audio bisa teman-teman lihat pada folder saved_models yang terdapat didalam folder vocoder. Prediksi audio dan target audio bisa teman-teman dengarkan dan menilai hasilnya apakah prediksi audio sudah mendekati target audionya atau belum serta menilai seberapa alami dan kemiripan antara prediksi dan targetnya.

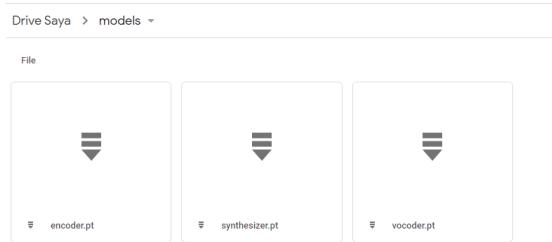


Gambar 11.28 Hasil Generate Audio

11.2 Demo Aplikasi Voice Cloning

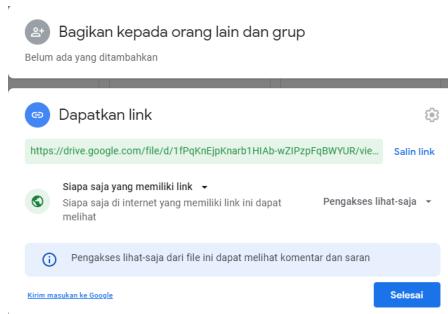
Sekarang kita memiliki tiga model yang telah di training yaitu speaker encoder model, synthesizer model, dan vocoder model. Berikut cara menjalankan aplikasi voice cloning dengan menggunakan ketiga model tersebut.

- Pertama buat folder pada google drive dengan nama models, lalu upload ketiga model kedalam folder tersebut seperti pada gambar 11.29



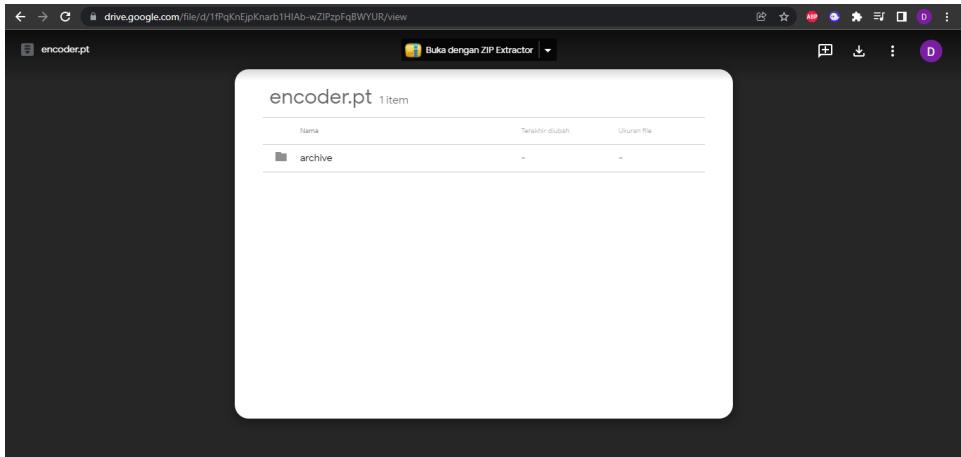
Gambar 11.29 Folder Models

- Klik kanan pada file model lalu pilih dapatkan link, ubah akses ke siapa saja yang memiliki link, lalu salin link.



Gambar 11.30 Link Akses

3. Akses link tersebut dan klik tombol download pada kanan atas, maka link akan berubah menjadi link download seperti berikut <https://drive.google.com/u/0/uc?id=1fPqKnEjpKnarb1HIAb-wZIPzpFqBWYUR>. Copy link dan simpan, link ini akan berguna untuk mendownload model teman-teman nantinya. Lakukan hal yang sama pada kedua model lainnya.



Gambar 11.31 Link Akses

4. Buat new notebook lalu rename menjadi demo.ipynb. Kemudian tambahkan kode program untuk clone repo, install requirements, dan download model. Paste link download model yang telah teman-teman simpan sebelumnya pada bagian kode program download model.

```
#@title Setup Real-Time-Voice-Cloning
#@markdown * clone the project
#@markdown * download pretrained models
#@markdown * initialize the voice cloning models

%tensorflow_version 1.x
import os
from os.path import exists, join, basename, splitext

git_repo_url = 'https://github.com/dindamajesty13/Real-Time-Voice-Cloning.git'
project_name = splitext(basename(git_repo_url))[0]
if not exists(project_name):
    # clone and install
    !git clone -q --recursive {git_repo_url}
    # install dependencies
    !cd {project_name} && pip install -q -r requirements.txt
    !pip install -q gdown
    !apt-get install -qq libportaudio2
    !pip install -q https://github.com/tugstugi/dl-colab-notebooks/archive/cola

# download pretrained model
!cd {project_name} && mkdir -p saved_models/default/
!cd {project_name}/saved_models/default/ && gdown https://drive.google.com/
!cd {project_name}/saved_models/default/ && gdown https://drive.google.com/
!cd {project_name}/saved_models/default/ && gdown https://drive.google.com/
```

Setup Real-Time-Voice-Cloning

- clone the project
- download pretrained models
- initialize the voice cloning models

Gambar 11.32 Demo Notebook

```
1 # @title Setup CorentinJ/Real-Time-Voice-Cloning
2
3 #@markdown * clone the project
4 #@markdown * download pretrained models
5 #@markdown * initialize the voice cloning models
6
7 %tensorflow_version 1.x
8 import os
9 from os.path import exists, join, basename, splitext
10
11 #sesuaikan repo dengan repo masing-masing.
12 git_repo_url = 'https://github.com/dindamajesty13/Real-Time-Voice-Cloning.git'
13 project_name = splitext(basename(git_repo_url))[0]
14 if not exists(project_name):
15     # clone and install
16     !git clone -q --recursive {git_repo_url}
17     # install dependencies
18     !cd {project_name} && pip install -q -r requirements.txt
19     !pip install -q gdown
20     !apt-get install -qq libportaudio2
21     !pip install -q https://github.com/tugstugi/dl-colab-notebooks/archive/colab_utils.zip
22
23 # download pretrained model
24 !cd {project_name} && mkdir -p saved_models/default/
25 #paste kan link download model pada kode program di bawah ini.
26 !cd {project_name}/saved_models/default/ && gdown https://drive.google.com/uc?id=1fPqKnEjpKnarb1HIAb-wZlPzpFqBWYUR
27 !cd {project_name}/saved_models/default/ && gdown https://drive.google.com/uc?id=12oq_PFdcbKsx4BVdSEiSdvg5x5Cyrjza
28 !cd {project_name}/saved_models/default/ && gdown https://drive.google.com/uc?id=1_OEHrkS7u1NzQ0WQjeWVdrUYhgOYxQng
```

```

30 import sys
31 sys.path.append(project_name)
32
33 from IPython.display import display, Audio, clear_output
34 from IPython.utils import io
35 import ipywidgets as widgets
36 import numpy as np
37 from dl_colab_notebooks.audio import record_audio, upload_audio
38
39 from synthesizer.inference import Synthesizer
40 from encoder import inference as encoder
41 from vocoder import inference as vocoder
42 from pathlib import Path
43
44 !ls
45 encoder.load_model(project_name / Path("saved_models/default/
    encoder.pt"))
46 synthesizer = Synthesizer(project_name / Path("saved_models/
    default/synthesizer.pt"))
47 vocoder.load_model(project_name / Path("saved_models/default/
    vocoder.pt"))

```

Listing 11.2 Setup Project

5. Klik tambahkan code lalu ketikkan kode program berikut untuk record atau upload suara sampel.

```

+ Code & Text
  #title Record or Upload
  #@markdown * Either record audio from microphone or upload audio from file (.mp3 or .wav)

  SAMPLE_RATE = 22050
  record_or_upload = "Record" #@param ["Record", "Upload (.mp3 or .wav)"]
  record_seconds = 10#@param {type:"number", min:1, max:10, step:1}

  embedding = None
  def _compute_embedding(audio):
    display(Audio(audio, rate=SAMPLE_RATE, autoplay=True))
    global embedding
    embedding = None
    embedding = encoder.embed_utterance(encoder.preprocess_wav(audio, SAMPLE_RATE))
  def record_file(b):
    clear_output()
    audio = record_audio(record_seconds, sample_rate=SAMPLE_RATE)
    _compute_embedding(audio)
  def upload_audio(b):
    clear_output()
    audio = upload_audio(sample_rate=SAMPLE_RATE)
    _compute_embedding(audio)

  if record_or_upload == "Record":
    button = widgets.Button(description="Record Your Voice")
    button.on_click(record_file)
    display(button)
  else:
    #button = widgets.Button(description="Upload Voice File")
    #button.on_click(upload_audio)

```

Gambar 11.33 Record or Upload Audio Sample

```

1 # @title Record or Upload
2 #@markdown * Either record audio from microphone or upload audio
   from file (.mp3 or .wav)
3
4 SAMPLE_RATE = 22050
5 record_or_upload = "Record" #@param ["Record", "Upload (.mp3 or .
   wav)"]

```

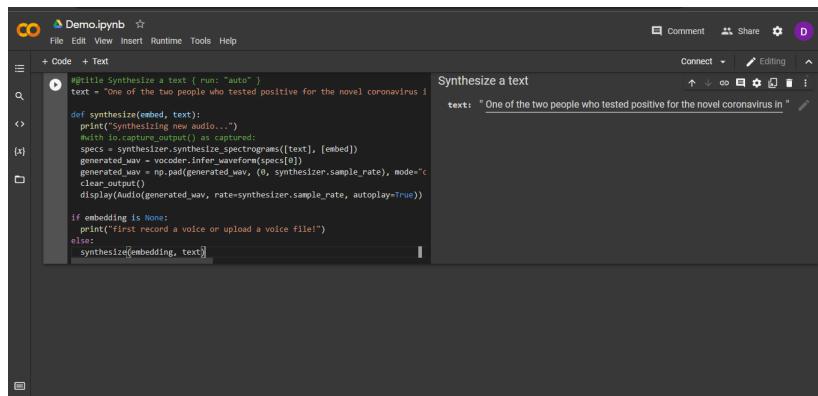
```

6 record_seconds = 10 #@param {type:"number", min:1, max:10, step
7 :1}
8
9 embedding = None
10 def _compute_embedding(audio):
11     display(Audio(audio, rate=SAMPLE_RATE, autoplay=True))
12     global embedding
13     embedding = None
14     embedding = encoder.embed_utterance(encoder.preprocess_wav(
15         audio, SAMPLE_RATE))
16 def _record_audio(b):
17     clear_output()
18     audio = record_audio(record_seconds, sample_rate=SAMPLE_RATE)
19     _compute_embedding(audio)
20 def _upload_audio(b):
21     clear_output()
22     audio = upload_audio(sample_rate=SAMPLE_RATE)
23     _compute_embedding(audio)
24
25 if record_or_upload == "Record":
26     button = widgets.Button(description="Record Your Voice")
27     button.on_click(_record_audio)
28     display(button)
29 else:
30     #button = widgets.Button(description="Upload Voice File")
31     #button.on_click(_upload_audio)
32     _upload_audio("")

```

Listing 11.3 Record or Upload Audio Sample

6. Tambahkan kode program untuk menginputkan teks, synthesize teks dan audio sampel, generate audio output berdasarkan teks input dan audio sampel.



Gambar 11.34 Synthesize and Generate Audio

```

1 #@title Synthesize a text { run: "auto" }
2 text = "One of the two people who tested positive for the novel
3     coronavirus in the United Kingdom is a student at the

```

```

1     University of York in northern England.” #@param {type:"string"}
2
3
4 def synthesize(embed, text):
5     print("Synthesizing new audio...")
6     #with io.capture_output() as captured:
7     specs = synthesizer.synthesize_spectrograms([text], [embed])
8     generated_wav = vocoder.infer_waveform(specs[0])
9     generated_wav = np.pad(generated_wav, (0, synthesizer.
10                            sample_rate), mode="constant")
11    clear_output()
12    display(Audio(generated_wav, rate=synthesizer.sample_rate,
13                  autoplay=True))
14
15 if embedding is None:
16     print("first record a voice or upload a voice file!")
17 else:
18     synthesize(embedding, text)

```

Listing 11.4 Synthesize dan Generate Audio Output

11.3 Pembahasan Error dan Solusi

Berikut beberapa error dan solusinya yang saya temui pada saat pembuatan project ini:

1. librosa.util.exceptions.ParameterError: Audio buffer is not finite everywhere
Error ini dapat terjadi karena adanya nan value pada audio dataset teman-teman. Jadi pastikan file audio pada dataset teman-teman sebelum melakukan preprocessing dan training model agar terhindar dari error ini
2. RuntimeError: cuDNN error: CUDNN_STATUS_EXECUTION_FAILED
Error ini biasanya terjadi karena versi CUDA, cuDNN, dan Pytorch yang tidak sesuai. Saya menyarankan teman-teman untuk menginstal ulang pytorch dengan versi yang sesuai.
3. RuntimeError: CUDA out of memory.
Error ini terjadi akibat kapasitas penyimpanan GPU mencapai maksimal dan tidak ada lagi ruang kosong tersisa. Pastikan teman-teman menggunakan GPU yang lebih baik dengan kapasitas yang lebih besar, upgrade GPU, atau jika memiliki keterbatasan biaya teman-teman bisa menggunakan Google Colab atau menggunakan CPU. Penggunaan CPU tidak disarankan karena lambat dan menyita banyak waktu.
4. RuntimeError: cuDNN error: CUDNN_STATUS_MAPPING_ERROR
Error ini terjadi ketika versi PyTorch tidak cocok dengan versi CUDA atau GPU sudah tidak didukung karena terlalu tua. Saya sarankan untuk mendapatkan GPU yang lebih baru, menurut saya teman-teman setidaknya menggunakan GTX 1080Ti.
5. audioread.NoBackendError
Error ini terjadi ketika file audio memiliki format selain .wav. Jika teman-teman menggunakan file audio dengan format .mp3 atau .flac maka pastikan teman-teman menginstall ffmpeg pada laptop atau komputer teman-teman.

DAFTAR PUSTAKA

- [1] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, “Transfer learning from speaker verification to multispeaker text-to-speech synthesis,” *CoRR*, vol. abs/1806.04558, 2018. [Online]. Available: <http://arxiv.org/abs/1806.04558>
- [2] S. Ö. Arik, G. F. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, “Deep voice 2: Multi-speaker neural text-to-speech,” *CoRR*, vol. abs/1705.08947, 2017. [Online]. Available: <http://arxiv.org/abs/1705.08947>
- [3] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, “Deep speaker feature learning for text-independent speaker verification,” *arXiv preprint arXiv:1705.03670*, 2017.
- [4] S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, “Multi-speaker sequence-to-sequence speech synthesis for data augmentation in acoustic-to-word speech recognition,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6161–6165.
- [5] M. Mimura, S. Ueno, H. Inaguma, S. Sakai, and T. Kawahara, “Leveraging sequence-to-sequence speech synthesis for enhancing acoustic-to-word speech recognition,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 477–484.
- [6] A. J. Hunt and A. W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database,” *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, pp. 373–376 vol. 1, 1996.

- [7] H. Ze, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7962–7966.
- [8] H. Zen, K. Tokuda, and A. W. Black, “Statistical parametric speech synthesis,” *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167639309000648>
- [9] P. Partila, J. Tovarek, G. H. Ilk, J. Rozhon, and M. Voznak, “Deep learning serves voice cloning: How vulnerable are automatic speaker verification systems to spoofing trials?” *IEEE Communications Magazine*, vol. 58, no. 2, pp. 100–105, 2020.
- [10] L. Zhao and F. Chen, “Research on voice cloning with a few samples,” in *2020 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, 2020, pp. 323–328.
- [11] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomirgiannakis, and Y. Wu, “Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions,” *CoRR*, vol. abs/1712.05884, 2017. [Online]. Available: <http://arxiv.org/abs/1712.05884>
- [12] Q. Xie, X. Tian, G. Liu, K. Song, L. Xie, Z. Wu, H. Li, S. Shi, H. Li, F. Hong, H. Bu, and X. Xu, “The multi-speaker multi-style voice cloning challenge 2021,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 8613–8617.
- [13] Y. Win and T. Masada, “Myanmar text-to-speech system based on tacotron-2,” in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 578–583.
- [14] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *CoRR*, vol. abs/1609.03499, 2016. [Online]. Available: <http://arxiv.org/abs/1609.03499>
- [15] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, “End-to-end text-dependent speaker verification,” *CoRR*, vol. abs/1509.08062, 2015. [Online]. Available: <http://arxiv.org/abs/1509.08062>