

## **BUKU TUTORIAL OBJECT DETECTION**

### **“PENGOLAHAN DATA GAMBAR UNTUK DETEKSI PLAT NOMOR KENDARAAN MENGGUNAKAN METODE CONVOLUTION NEURAL NETWORK (CNN)”**

Buku ini dibuat untuk memenuhi persyaratan kelulusan  
matakuliah Program Internship I



**Dibuat Oleh,**

**1.16.4.049      Nur Arkhamia Batubara**

**PROGRAM DIPLOMA IV TEKNIK INFORMATIKA  
POLITEKNIK POS INDONESIA  
BANDUNG  
2019**

## KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah memberikan rahmat, hidayah serta kekuatan sehingga Buku tutorial ini dengan judul “Pengolahan Data Gambar Untuk Mendeteksi Plat Nomor Kendaraan Menggunakan Metode Convolution Neural Network (CNN)” dapat terselesaikan.

Banyak kendala yang dihadapi dalam penyusunan Buku ini dan penulis menyadari bahwa penyusunan laporan ini masih belum sempurna. Ini mengingat keterbatasan pengetahuan, pengalaman serta kemampuan penulis. Penulis megharapkan kritik dan saran yang sifatnya membangun dari pembaca. Untuk itu penulis mengucapkan terima kasih kepada:

1. Allah SWT, karena dengan Rahmat dan Ridho-Nya penulis dapat menyelesaikan laporan intership I.
2. Kedua orang tua dan keluarga penulis yang telah mendorong dan memberi semangat kepada penulis.
3. Hedi Krishna selaku pembimbing external di PT.Telkomunikasi Indonesia Tbk.
4. Rolly Maulana Awangga, S.T., M.T. selaku dosen pembimbing internship I.
5. Nisa Hanum Harani, S.Kom., M.T.selaku Koordinator Internship I.
6. M. Yusril Helmi Setyawan, S.Kom., M.Kom. selaku Ketua Prodi DIV Teknik Informatika.

Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga buku ini membawa manfaat bagi pengembangan ilmu.

Bandung, 17 Januari 2020

Penulis

## DAFTAR ISI

---

BAB I .....	1
PENGENALAN AI, ML, NN DAN DL.....	1
1.1 PENGANTAR ARTIFICIAL INTELLIGENCE.....	1
1.2 Pengantar Machine Learning .....	10
1.3 Pengantar Neural Network.....	25
1.4 Pengantar Deep Learning .....	35
BAB II .....	42
PENGENALAN CNN, R-CNN, FAST R-CNN, FASTER R-CNN DAN YOLO .....	42
2.1 Pengantar Convolution Neural Network (CNN) .....	42
2.2 Pengantar Region - Convolution Neural Network (R-CNN) .....	71
2.3 Pengantar Fast Region - Convolution Neural Network (Fater R- CNN) .....	75
2.4 Faster Region - Convolution Neural Network (Faster R-CNN)....	76
2.5 Pengantar YOLO - You Only Look Once .....	82
BAB III .....	85
PENJELASAN TOOLS DAN BAHASA PEMOGRAMAN YANG DIGUNAKAN .....	85
3.1 Tools Yang Digunakan.....	85
3.2 Bahasa Pemograman .....	94
BAB IV.....	95
INSTALASI TOOLS YANG DIGUNAKAN .....	95
4.1 Tools Yang Digunakan.....	95
4.2 Bahasa Pemograman Yang Digunakan .....	107
BAB V .....	112
PENJELASAN, INSTALASI LIBRARY & FRAMEWORK YANG DIGUNAKAN .....	112

5.1	Membuat Environment Pada Anacoda .....	112
5.1.1	Framework Tensorflow .....	115
5.1.2	Library Keras .....	118
5.1.3	Library Open-CV .....	120
5.1.4	Library Pandas .....	122
5.1.5	Library cudnn.....	124
5.1.6	Library numpy .....	130
5.1.7	Library Tensorflow-GPU.....	135
5.1.8	Library matplotlib .....	136
5.1.9	Library pillow .....	138
5.1.10	Library absl-py.....	140
5.1.11	Library astor.....	141
5.1.12	Library gast.....	142
5.1.13	Library html5lib .....	143
5.1.14	Library Markdown .....	143
5.1.15	Library protobuf.....	144
5.1.16	Library termcolor .....	145
5.1.17	Library Werkzeug .....	146
5.1.18	Library contextlib2 .....	147
5.1.19	Library Cython.....	148
5.1.20	Library lxml .....	150
5.1.21	Library json-scema .....	151
5.1.22	Library Scikit-Learn .....	153
5.1.23	Library Bleach .....	154
5.1.24	Library Tensorboard.....	155
5.1.25	Library six.....	157
BAB VI.....		159
TUTORIAL OBJECT DETECTION PLATE NUMBER.....		159

6.1	Mempersiapkan Tools .....	160
6.2	Mengatur Direktori TensorFlow dan Lingkungan Virtual Anaconda 161	
6.3	Mempersiapkan Dataset .....	166
6.4	Proses Pengujian Model .....	214
BAB VII .....		232
HASIL DAN KESIMPULAN.....		232
7.1	Kesimpulan Dan Saran .....	<b>Error! Bookmark not defined.</b>

## DAFTAR GAMBAR

---

Gambar 1. 1 Diagram Artificial Intelligence .....	4
Gambar 1. 2 Perception .....	27
Gambar 1. 3 Struktur Neuron Pada Otak Manusia .....	28
Gambar 1. 4 Struktur NN .....	30
Gambar 1. 5 Linier Function .....	32
Gambar 1. 6 Sigmoid and Tanh Function .....	32
Gambar 1. 7 Arsitektur Neural Network .....	33
Gambar 1. 8 Persamaan Forward Pass .....	35
Gambar 1. 9 Perbedaan Arsitektur Neural Network dan Deep Learning	41
Gambar 2. 1 Arsitektur CNN .....	44
Gambar 2. 2 Ilustrasi Input RGB .....	45
Gambar 2. 3 Gambaran Citra RGB .....	46
Gambar 2. 4 Proses Convolusi Perkalian Dot .....	48
Gambar 2. 5 Zero Padding Pada Matriks 4x4 Menjadi 6x6 .....	50
Gambar 2. 6 Maximum Pooling Layer .....	54
Gambar 2. 7 Average Pooling Layer .....	55
Gambar 2. 8 Activation Functions .....	56
Gambar 2. 9 Grafik Activation .....	59
Gambar 2. 10 Grafik Activation .....	60
Gambar 2. 11 Ilustrasi Fully Connected Layer .....	61
Gambar 2. 12 Piramida Abstraksi Saraf .....	66
Gambar 2. 13 Layer MLP .....	68
Gambar 2. 14 Neuron Input x Neuron Output x Tinggi x Lebar .....	69
Gambar 2. 15 Contoh dan Arsitektur Dari CNN .....	70
Gambar 2. 16 Perbedaan Arsitektur NN Dengan CNN .....	71

Gambar 2. 17 Arsitektur R-CNN.....	73
Gambar 2. 18 Proses R-CNN .....	74
Gambar 2. 19 Fast R-NN.....	75
Gambar 2. 20 Perbandingan Algoritma Objek Deteksi.....	76
Gambar 2. 21 Arsitektur Faster R-CNN.....	76
Gambar 2. 22 Perbandingan Kecepatan Uji Waktu Dari Algoritma Objek Deteksi.....	77
Gambar 2. 23 Faster R-CNN .....	78
Gambar 2. 24 Modul Pertama Inception V2.....	80
Gambar 2. 25 Modul kedua Inception V2 .....	80
Gambar 2. 26 Modul Ketiga Inception V2 .....	81
Gambar 2. 27 Region Proposal Network (RPN).....	81
Gambar 2. 28 Cara Kerja Yolo .....	82
Gambar 2. 29 Arsitektur Yolo .....	84
Gambar 3. 1 Tampilan Sublime .....	86
Gambar 3. 2 Tampilan LabelImg .....	88
Gambar 3. 3 Aliran Pemrosesan CUDA .....	91
Gambar 3. 4 Tampilan Anaconda Desktop .....	93
Gambar 3. 5 Tampilan Anaconda Desktop .....	93
Gambar 4. 1 Downlod file .exe .....	96
Gambar 4. 2 Double Klik File .exe .....	96
Gambar 4. 3 Proses Instalasi Sublime.....	97
Gambar 4. 4 Proses Instalasi Sublime.....	98
Gambar 4. 5 Proses Instalasi Sublime.....	98
Gambar 4. 6 Proses Instalasi Sublime.....	98
Gambar 4. 7 Proses Instalasi Sublime.....	99
Gambar 4. 8 Instalasi LabelImg .....	99

Gambar 4. 9 Downlod File .exe CUDA .....	101
Gambar 4. 10 Melakukan Double Klik .....	102
Gambar 4. 11 Proses Instalasi CUDA.....	102
Gambar 4. 12 Downlod File Anaconda.....	103
Gambar 4. 13 Proses Instalasi Anaconda .....	104
Gambar 4. 14 Proses Instalasi.....	104
Gambar 4. 15 Install VS Code .....	105
Gambar 4. 16 Instalasi Selesai.....	106
Gambar 4. 17 Buka Aplikasi Anaconda.....	106
Gambar 4. 18 Downlod File Python .....	107
Gambar 4. 19 Eksekusi File Python.....	107
Gambar 4. 20 Install Python .....	108
Gambar 4. 21 Tentukan Lokasi Intallasi .....	108
Gambar 4. 22 Aktifkan “Add python.exe to path” .....	109
Gambar 4. 23 Hasil Aktivasi .....	109
Gambar 4. 24 Instalasi Selesai.....	110
Gambar 4. 25 Melakukan Uji Coba .....	110
Gambar 4. 26 Uji Coba Python CMD .....	111
Gambar 5. 1 Membuka Anaconda Commannd Prompt .....	112
Gambar 5. 2 Membuat environment .....	113
Gambar 5. 3 Proses Membuat Environment.....	113
Gambar 5. 4 Proses Membuat Environment.....	114
Gambar 5. 5 Instalasi Selesai .....	114
Gambar 5. 6 Aktivasi Environment .....	115
Gambar 5. 7 Tensorflow.....	116
Gambar 5. 8 Instalasi Tensorflow .....	118
Gambar 5. 9 Instalasi Keras.....	119

Gambar 5. 10 Instalasi OpenVC .....	122
Gambar 5. 11 Instalasi Pandas.....	123
Gambar 5. 12 Downlod Cudnn .....	124
Gambar 5. 13 Daftar Membersip .....	125
Gambar 5. 14 Login Membership .....	125
Gambar 5. 15 Selesai Login .....	126
Gambar 5. 16 Memilih versi cudnn .....	126
Gambar 5. 17 Pilih Penyimpanan File .....	127
Gambar 5. 18 EXtract File .zip .....	127
Gambar 5. 19 Fili cudnn.....	128
Gambar 5. 20 Membuka Environment .....	128
Gambar 5. 21 Membuka Path CUDA .....	129
Gambar 5. 22 Copy Link .....	129
Gambar 5. 23 Replace File .....	130
Gambar 5. 24 Instalasi Numpy .....	135
Gambar 5. 25 Instalasi Tensorflow-gpu.....	136
Gambar 5. 26 Instalasi Matplotlib .....	138
Gambar 5. 27 Instalasi Pillow.....	139
Gambar 5. 28 Instalasi absl-py .....	140
Gambar 5. 29 Instalasi astor .....	142
Gambar 5. 30 Instalasi gast .....	142
Gambar 5. 31 Instalasi Markdown.....	144
Gambar 5. 32 Instalasi Protobuf .....	145
Gambar 5. 33 Instalasi termcolor.....	145
Gambar 5. 34 Instalasi werlzeug.....	147
Gambar 5. 35 Instalasi contexlib2 .....	148
Gambar 5. 36 Instalasi cython .....	149

Gambar 5. 37 Instalasi lxml.....	151
Gambar 5. 38 Instalasi Scikit-learn.....	154
Gambar 5. 39 Instalasi bleach.....	155
Gambar 5. 40 Instalasi Tensorboard .....	156
Gambar 5. 41 Visualisasi Tensorboard .....	157
Gambar 5. 42 Instalasi six .....	158
Gambar 5. 43 Data Train.....	169
Gambar 5. 44 Data Test.....	170
Gambar 5. 45 Data Pengujian.....	170
Gambar 6. 1 Folder Object Detection .....	162
Gambar 6. 2 Environment Variabel .....	165
Gambar 6. 3 Python Path.....	165
Gambar 6. 4 Python Path Anaconda .....	166
Gambar 6. 5 Dataset Kaggle .....	167
Gambar 6. 6 Buka aplikasi labelImg.....	171
Gambar 6. 7 Klik tombol “OpenDir” .....	172
Gambar 6. 8 Pilih direktori dataset .....	172
Gambar 6. 9 Klik Open Save Dir .....	173
Gambar 6. 10 Pilih direktori penyimpanan anotasi .....	173
Gambar 6. 11 Klik tombol “Create RectBox” .....	174
Gambar 6. 12 Arahkan kursos .....	174
Gambar 6. 13 Barikan Label.....	175
Gambar 6. 14 Code Pada File xml .....	176
Gambar 6. 15 Perintah xml to csv.....	180
Gambar 6. 16 Proses xml ke csv .....	181
Gambar 6. 17 File CSV Data Test .....	182
Gambar 6. 18 File Csv Data Train .....	182

Gambar 6. 19 Proses Generate TFRecord .....	189
Gambar 6. 20 Hasil Generate TFRecord .....	190
Gambar 6. 21 Susunan Direktori .....	198
Gambar 6. 22 Perintah Training Data .....	206
Gambar 6. 23 Proses Sebelum Training Dimulai .....	206
Gambar 6. 24 Proses Sebelum Training .....	207
Gambar 6. 25 Proses Training Data .....	207
Gambar 6. 26 Proses Training Selesai .....	208
Gambar 6. 27 Checkpoint Training.....	209
Gambar 6. 28 Perintah Tensorboard .....	209
Gambar 6. 29 Grafik Loss .....	210
Gambar 6. 30 Grafik Distributions .....	210
Gambar 6. 31 Grafik Histogram .....	211
Gambar 6. 32 Grafik Histogram .....	211
Gambar 6. 33 Generate Model.....	212
Gambar 6. 34 Proses Generate Model.....	212
Gambar 6. 35 Proses Generate Model NN .....	213
Gambar 6. 36 File Frozen Inference Graph.....	213
Gambar 6. 37 Proses Pengujian Image.....	218
Gambar 6. 38 Menampilkan Objek Deteksi Image .....	219
Gambar 6. 39 Proses Pengujian Vidio .....	223
Gambar 6. 40 Menampilkan Objek Deteksi Vidio .....	225
Gambar 6. 41 Proses Pengujian Webcam .....	230
Gambar 6. 42 Menampilkan Objek Deteksi Webcam .....	231

## **DAFTAR TABEL**

---

Tabel 5. 1 Hierarki toolkit TensorFlow .....	116
Tabel 6. 1 Pengumpulan Data.....	168
Tabel 6. 2 Kode Program XML ke CSV .....	177
Tabel 6. 3 Kode Program Generate TFRecord .....	183
Tabel 6. 4 Kode Program Konfigurasi LabelMap .....	190
Tabel 6. 5 Kode Konfigurasi Pipeline .....	192
Tabel 6. 6 Kode Program Train.py .....	199
Tabel 6. 7 Kode Program Pengujian Image.....	214
Tabel 6. 8 Kode Program Pengujian Vidio .....	220
Tabel 6. 9 Kode Program Pengujian Webcam.....	226
Tabel 7. 1 Hasil Pengujian.....	232

# **BAB I**

---

## **PENGENALAN AI, ML, NN DAN DL**

Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning adalah istilah dalam dunia teknologi informasi yang sangat populer saat ini. Pada buku ini penulis akan memaparkan materi tentang teknologi tersebut dengan penerapan teknologi deep learning yaitu object detection.

### **1.1 PENGANTAR ARTIFICIAL INTELLIGENCE**

#### **1.1.1 Definisi Artificial Intelligence**

Artificial Intelligence atau biasa disebut Kecerdasan Buatan adalah kecerdasan yang ditambahkan kepada suatu sistem yang bisa diatur dalam konteks ilmiah didefinisikan sebagai kecerdasan entitas ilmiah. Kecerdasan buatan diartikan sebagai “kemampuan sistem untuk menafsirkan data eksternal dengan benar, untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut untuk mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel”. Kecerdasan dibuat dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Penerapan Artificial Intelligence antara lain sistem pakar, permainan komputer (games), logika fuzzy, jaringan saraf tiruan dan robotika.

Beberapa hal yang kelihatannya sulit untuk kecerdasan manusia, tetapi untuk Informatika relatif tidak bermasalah. Contohnya mentransformasikan persamaan, menyelesaikan persamaan integral, membuat permainan catur atau Backgammon. Di lain sisi hal yang bagi manusia kelihatannya menuntut sedikit kecerdasan, sampai sekarang masih sulit untuk direalisasikan dalam Informatika. Seperti contoh:

Pengenalan Objek/Muka, bermain sepak bola. Teknologi AI menciptakan cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku, pembelajaran dan adaptasi yang cerdas dalam sebuah mesin.

Teknologi penelitian pada AI menyangkut pembuatan mesin dan program komputer untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Seperti halnya pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. AI diterapkan pada beberapa bidang yaitu ekonomi, sains, obat-obatan, teknik dan militer, seperti yang telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game.

AI terbagi ke dalam dua paham pemikiran yaitu AI Konvensional dan Kecerdasan Komputasional CI (Computational Intelligence). Yang pertama AI konvensional bekerja dengan melibatkan metode-metode yang sekarang diklasifikasikan sebagai pembelajaran mesin, yang ditandai dengan formalisme dan analisis statistik. Disebut juga sebagai AI simbolis, AI logis, AI murni dan AI cara lama GOFAI (Good OldFashioned Artificial Intelligence). Metode-metodenya meliputi:

- Sistem pakar: menerapkan kapabilitas pertimbangan untuk mencapai kesimpulan. Sistem pakar dapat memproses sejumlah besar informasi yang diketahui dan menyediakan kesimpulan-kesimpulan berdasarkan pada informasi-informasi tersebut.
- Petimbangan berdasar kasus
- Jaringan Bayesian

- AI berdasarkan tingkah laku metode modular pada pembentukan sistem AI secara manual

Pada kecerdasan komputasional melibatkan pengembangan atau pembelajaran iteratif (misalnya penalaan parameter seperti dalam sistem koneksiis. Dalam pembelajaran berdasarkan pada data empiris dan diasosiasikan dengan AI non-simbolis, AI yang tak teratur dan perhitungan lunak. Metode-metode pokoknya meliputi:

1. Fuzzy Logic (FL)

Teknik ini digunakan oleh mesin untuk mengadaptasi bagaimana makhluk hidup menyesuaikan kondisi dengan memberikan keputusan yang tidak kaku 0 atau 1. Sehingga dimunculkan sistem logika fuzzy yang tidak kaku. Penerapan logika fuzzy ini salah satunya adalah untuk sistem penggereman kereta api di Jepang.

2. Evolutionary Computing (EC)

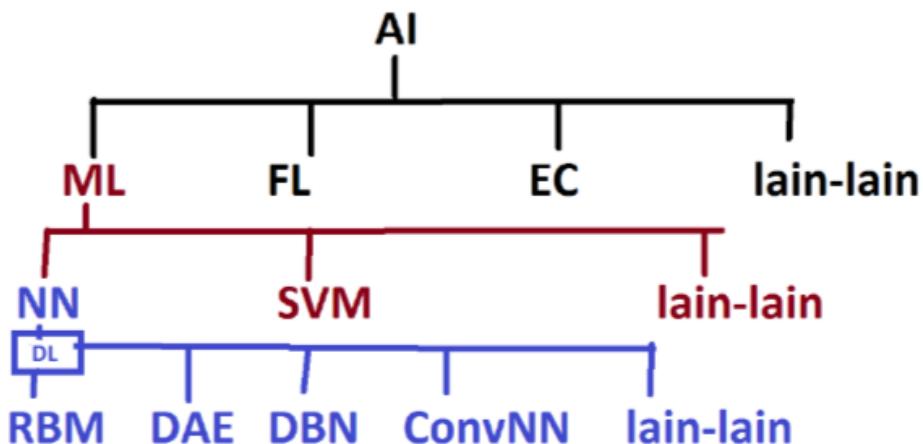
Pendekatan ini menggunakan skema evolusi yang menggunakan jumlah individu yang banyak dan memberikan sebuah ujian untuk menyeleksi individu terbaik untuk membangkitkan generasi selanjutnya. Seleksi tersebut digunakan untuk mencari solusi dari suatu permasalahan. Contoh dari pendekatan ini adalah Algoritme Genetika yang menggunakan ide mutasi dan kawin silang, Particle Swarm Optimization (PSO) yang meniru kumpulan binatang seperti burung dan ikan dalam mencari mangsa, Simulated Annealing yang menirukan bagaimana logam ditempa, dan masih banyak lagi.

3. Machine Learning (ML)

Pembelajaran mesin merupakan teknik yang paling populer karena banyak digunakan untuk menggantikan atau menirukan perilaku manusia untuk menyelesaikan masalah. Sesuai namanya mencoba menirukan

bagaimana proses manusia atau makhluk cerdas belajar dan mengeneralisasi

Skema utama dari AI bisa dilihat pada gambar berikut :



Gambar 1. 1 Diagram Artificial Intelligence

### 1.1.2 Sejarah Artificial Intelligence

Pada abad 17, Rene Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Kemudian Blaise Pascal menciptakan mesin penghitung digital mekanis pertama pada 1642. Pada 19, Charles Babbage dan Ada Lovelace bekerja pada mesin penghitung mekanis yang dapat diprogram. Bertrand Russell dan Alfred North Whitehead menerbitkan Principia Mathematica, yang merombak logika formal. Warren McCulloch dan Walter Pitts menerbitkan "Kalkulus Logis Gagasan yang tetap ada dalam Aktivitas" pada 1943 yang meletakkan fondasi untuk jaringan saraf. Tahun 1950-an adalah periode usaha aktif dalam AI.

AI merupakan inovasi terbaru di bidang ilmu pengetahuan. Mulai terbentuk sejak adanya komputer modern dan kira-kira terjadi sekitaran tahun 1950. Tidak bisa dipungkiri bahwa di tahun tersebut memang sedang

gencar-gencarnya pembuatan cikal bakal, konsep, hingga teknologi berbasis AI. Ilmu pengetahuan komputer ini khusus ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Pada awal 50-an, studi tentang “mesin berpikir” memiliki berbagai nama seperti cybernetics, teori automata, dan pemrosesan informasi. Pada tahun 1956, para ilmuan jenius seperti Alan Turing, Norbert Wiener, Claude Shannon dan Warren McCullough telah bekerja secara independen dibidang cybernetics, matematika, algoritma dan teori jaringan. Namun, seprang ilmuan komputer dan kognitif John McCarthy adalah orang yang dating dengan ide untuk bergabung dengan upaya penelitian terpisah ini kedalam satu bidang yang akan mempelajari topic baru untuk imajinasi manusia yaitu kecerdasan buatan. Dia adalah orang yang menciptakan istilah tersebut dan kemudian mendirikan laboratorium Kecerdasan Buatan di MIT dan Stan ford.

Pada tahun 1956, McCarthy yang sama mendirikan Konferensi Dartmouth di Hanover, New Hampshire. Peneliti terkemuka dalam teori kompleksitas, simulasi bahasa, hubungan antara keacakan dan pemikiran kreatif, jaringan saraf diundang. Tujuan dari bidang penelitian yang baru dibuat adalah untuk mengembangkan mesin yang dapat mensimulasikan setiap aspek kecerdasan. Itulah sebabnya Konferensi Dartmouth 1956 dianggap sebagai kelahiran Kecerdasan Buatan. Sejak saat itu, Kecerdasan Buatan telah hidup melalui decade kemuliaan dan cemoohan, yang dikenal luas sebagai musim panas dan musim dingin AI. Musim panasnya ditandai dengan optimism dan dana besar, sedangkan musim dinginnya dihadapkan dengan pemotongan dana, ketidakpercayaan dan pesimisme.

Tahun 1960 dan 1970, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program

Macsyma, program berbasis pengetahuan yang sukses pertama kali yaitu dalam bidang matematika. Marvin Minsky dan Seymour Papert kemudian menerbitkan Perceptrons, yang mendemonstrasikan batas jaringan saraf sederhana dan Alain Colmerauer mengembangkan bahasa komputer Prolog. Ted Shortliffe mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang kadangkala disebut sebagai sistem pakar pertama. Hans Moravec mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan berintang yang kusut secara mandiri.

1980, jaringan saraf digunakan secara meluas dengan algoritme perambatan balik, yang pertama kali diterangkan oleh Paul John Werbos pada 1974. 1982, para ahli fisika seperti Hopfield menggunakan teknik-teknik statistika untuk menganalisis sifat-sifat penyimpanan dan optimasi pada jaringan saraf. Ahli psikologi yaitu David Rumelhart dan Geoff Hinton, melanjutkan penelitian mengenai model jaringan saraf pada memori. Tahun 1985 sedikitnya empat kelompok riset menemukan kembali algoritme pembelajaran propagansi balik atau Back-Propagation learning. Dengan algoritma ini berhasil diimplementasikan ke dalam ilmu komputer dan psikologi.

Pada tahun 1990 ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi berbagai macam aplikasi. Yaitu Deep Blue, sebuah komputer permainan catur, mengalahkan Garry Kasparov dalam sebuah pertandingan 6 game yang terkenal pada tahun 1997. DARPA mengungkapkan bahwa biaya yang disimpan melalui penerapan metode AI untuk unit penjadwalan dalam Perang Teluk pertama telah mengganti seluruh investasi dalam penelitian AI sejak tahun 1950 pada pemerintah AS. DARPA memiliki sebuah tantangan hebat, yang dimulai pada 2004

dan berlanjut hingga hari ini, adalah sebuah pacuan untuk hadiah 2 juta dolar dimana kendaraan dikemudikan sendiri tanpa komunikasi dengan manusia, menggunakan GPS dan komputer dengan susunan sensor yang canggih, melintasi beberapa ratus mil daerah gurun yang menantang.

Konsep dan teknologi kecerdasan buatan disempurnakan oleh seorang ahli yang namanya masih diingat sampai sekarang sebagai seorang pakar kecerdasan buatan, yaitu Alan Turin. Pada saat itu, Alan Turin meneliti dan menguji coba algoritma AI yang diberi nama dengan “Turing Test”. Hingga seiring berkembangnya waktu, konsep teknologi AI banyak digunakan di berbagai teknologi baik itu multimedia, search engine, dan masih banyak lainnya. Rasanya itulah sekilas mengenai sejarah AI yang diramalkan akan membuat kemajuan teknologi dengan sangat luar biasa.

### **1.1.3 Perkembangan Kecerdasan Buatan**

Teknologi Artificial Intelligence semakin ramai dibahas dalam berbagai diskusi teknologi di seluruh dunia. Menurut kebanyakan orang, pekerjaan seperti kasir, operator telepon, pengendara truk, dan lainnya sangat berpeluang besar untuk tergantikan oleh Artificial Intelligence. Mengapa terjadi hal demikian? dikarenakan memang bahwa AI lebih unggul dalam hal kinerja, fitur dan lain sebagainya. Namun, dalam beberapa aspek memang pekerja manusia masih unggul dibandingkan AI itu sendiri.

Para generasi muda yang ada di dunia terutama di daerah Asia terlihat sudah memahami fungsi dan efek dari AI dalam kehidupan kita sehari-hari. Berdasarkan survei yang dilakukan oleh Microsoft, terdapat 39 persen responden yang mempertimbangkan untuk menggunakan mobil tanpa pengemudi dan 36 persen lainnya setuju bahwa robot masa depan dengan software untuk beroperasi mampu meningkatkan produktivitas. Dari survey tersebut kita sebagai pengguna AI harus lebih bijaksana dalam

pengembangan dan penggunaan dari AI sehingga tanpa memberikan efek samping terhadap etos kerja dan keseharian kita sebagai pengguna dalam kehidupan sehari-hari.

AI Summer 1 (1956-1973) KOnferensi Dartmounth diikuti oleh 17 tahun kemajuan luar biasa. Proyek penelitian yang dilakukan di MIT, universitas di Edinburgh, Stanford dan Carnegie Mellon menerima dana besar-besaran, yang akhirnya membawa hasil. Selama tahun-tahun itu lah komputer pemrograman mulai melakukan masalah aljabar, membuktikan teorema geometris, memahami dan menggunakan sintaks dan tata bahasa Inggris. Terlepas dari ditinggalkannya koneksionisme dan terjemahan mesin yang gagal, yang menunda penelitian Natural Language Processing (NLP) selama bertahun-tahun, banyak prestasi dari masa lalu yang membuat sejarah.

Berikut ini beberapa diantaranya, Pelopor pembelajaran mesin, Ray Solomonoff meletakkan dasar-dasar teori metematika AI, memperkenalkan metode Bayesian universal untuk inferensi dan preddiksi induktif Thomas Evans menciptakan program ANALOGI heuristik, yang memungkinkan komputer memecahkan masalah geometri-analogi Unimation, perusahaan robotika pertama didunia, menciptakan robot industri

Unimate, yang bekerja pada jalur perakitan modil Genenral Motors. Joseph Weizenbaum membangun ELIZA-program interaktif yang dapat membawa percakapan dalam bahasan Inggris tentang topik apapun. Ross Quillian menunjukkan jaring semantik, sedangkan Jaime Carbonell (Sr.) mengembangkan Cendikia-program interaktif untuk instruksi yang dibantu komputer berdasarkan jaring semantik. Edward Feigenbaum dan

Julian Feldman menerbitkan Computeks and Thought, kumpulan artikel pertama tentang AI.

#### **1.1.4 Fungsi Artificial Intelligence**

Teknologi kecerdasan buatan berfungsi mengembangkan metode dan sistem untuk menyelesaikan suatu masalah, yang mana masalah tersebut juga dapat diselesaikan oleh manusia. Misalnya pencarian tempat, bidang bisnis, rumah tangga dan dapat meningkatkan kinerja sistem informasi yang berbasis komputer. Seperti contohnya robot cerdas. Robot ini dapat membantu pekerjaan manusia menggunakan sistem navigasi. Kedigunakan dibidang kesehatan yakni untuk merawat orang jompo, konsultasi, dan pendekripsi tumor, membantu melakukan operasi medis dll. Diperkirakan ada 90 startup yang mengembangkan AI di insustri kesehatan.

Dalam industri ekonomi dan keuangan, AI membantu di berbagai hal seperti investasi keuangan, pencataan keuangan, jual beli saham, penipuan dan kriminalisasi di bank, dan lain-lain. Di bidang penerbangan teknologi AI digunakan untuk mensimulasikan penerbangan sehingga pilot bisa diasisten dengan memberikan informasi pergerakan yang terbaik, informasi keadaan udara dan tekanan, dan lain-lain. Pada bidang pendidikan, diciptakan tutor robot di kelas untuk mengajar anak-anak mulai dari pelajaran biologi sampai dengan ilmu komputer, meskipun hal ini belum banyak dilakukan.

Di dalam dunia industri, teknologi robot pun sering dijumpai untuk menggantikan manusia, terutama dalam pekerjaan yang repetitif (berulang-ulang), seperti adanya mesin-mesin otomatis di pabrik-pabrik yang membantu proses produksi sebuah produk. Robot juga dapat mengambil alih beberapa pekerjaan paling berbahaya bagi manusia,

termasuk pembinasan bom. Menurut BBC robot ini bukan sekedar robot. Digunakan sebagai mitra fisik untuk meredakan bom, namun membutuhkan manusia untuk mengendalikannya daripada menggunakan AI, karena mereka telah menyelamatkan ribuan nyawa dengan mengambil alih salah satu pekerjaan paling berbahaya di dunia. Seiring kemajuan teknologi,

Pekerjaan lain juga dipertimbangkan kembali untuk integrasi robot. Yaitu welding yang terkenal untuk memproduksi zat beracun, panas yang hebat, dan kebisingan yang memikat, kini bisa di-outsource ke robot dalam banyak kasus. Robot worx memaparkan bahwa sel las robot sudah digunakan dan memiliki fitur keselamatan untuk mencegah pekerja manusia dari asap dan kerusakan fisik lainnya. Selain robot dapat digunakan oleh manusia untuk membantu sebuah pekerjaan dalam kehidupan sehari-hari dan pekerjaan yang berbahaya, robot juga dapat digunakan sebagai teman yang mana dapat membaca emosi manusia, mengembangkan emosinya sendiri serta membantu teman-teman manusianya tetap bahagia.

AI juga menimbulkan dampak negatif dikehidupan manusia. Adapun dampak yang terasa saat ini adalah hilangnya lapangan kerja dan pemindahan tenaga kerja secara ekonomi semua menggunakan komputer maupun robot. Umumnya AI juga menggunakan metode machine learning, dimana AI akan terus belajar dan belajar. AI Dapat diterapkan dalam proyek besar, seperti konstruksi bangunan, medis, otomotif, dan keamanan.

## **1.2 Pengantar Machine Learning**

### **1.2.1 Definisi Machine Learning**

Machine learning (ML) merupakan ilmu komputer yang bisa bekerja tanpa diprogram secara eksplisit. Banyak peneliti berpikir bagaimana cara

untuk membuat kemajuan menuju AI terhadap tingkat manusia. ML merupakan kecerdasan buatan yang mempelajari bagaimana membuat data. ML dibutuhkan untuk menerapkan teknik yang cepat dan kuat dalam menemukan masalah baru. Akhirnya, pemakaian teknik ini berkaitan dengan pembelajaran mesin dan AI. Mesin ini membuktikan kepada algoritma atau program yang berjalan di komputer. Maka dari itu, jika kita ingin belajar machine learning, pastikan anda terus berinteraksi dengan data. Semua pengetahuan machine learning pasti akan melibatkan data. Dari pada penasaran, langsung aja ikutin ulasan berikut.

Machine learning melakukan pembelajaran mesin dengan cara pendekatan kecerdasan buatan Artificial Intelligence yang berfokus pada pembuatan mesin yang dapat belajar tanpa diprogram secara eksplisit. Belajar adalah bagian yang sangat penting dari apa yang membuat kita menjadi manusia. Jika kita akan membangun AI yang dapat melakukan tugas dengan kecerdasan seperti manusia, maka kita perlu membuat mesin yang bisa belajar sendiri, berdasarkan pengalaman masa lalu mereka.

Machine learning telah banyak mempengaruhi dunia industri, sebagian besar dunia industri yang bekerja dengan sejumlah besar data telah mengakui nilai teknologi penggunaan machine learning. Tujuannya adalah untuk mendapatkan wawasan dari data yang mereka miliki, dan dengan adanya teknologi ini dapat membuat pekerjaan menjadi lebih efisien atau lebih cepat dengan adanya data baru yang cenderung sama.

Tahun 2018 Machine Learning adalah teknik untuk melakukan inferensi terhadap data dengan pendekatan matematis. Tujuan utama machine learning adalah untuk membuat model (matematis) yang merefleksikan pola-pola data. Dalam machine learning, inferensi yang dimaksud lebih menitikberatkan ranah hubungan variabel. Ked mudian

machine learning berada pada daerah representasi data/ilmu/pengetahuan dalam bentuk matematis karena keilmuan machine learning di turunkan dari matematika dan statistika. Machine learning ibarat sebuah “alat”, sama seperti rumus matematika. Bagaimana cara menggunakannya tergantung pada domain permasalahan. Tujuan utama dari machine learning ada dua yaitu untuk memprediksi masa depan (unobserved event) dan memperoleh ilmu pengetahuan (knowledge discovery atau discovering unknown structure).

Machine Learning adalah sub bidang dari Artificial Intelligence (AI). Perbedaan utama antara mesin dan manusia adalah kecerdasan, manusia mampu untuk belajar dari pengalaman sebelumnya dengan menganalisis data dan membuat keputusan dari pengetahuan masa lalu. Namun, kecerdasan buatan (AI) membawa mesin lebih dekat dengan manusia. Mesin dapat di program untuk mengingat dan mengambil keputusan seperti manusia. Machine learning bertujuan untuk lebih cepat dan lebih akurat dari pada manusia, dan dapat mempelajari data melalui kecerdasan buatan (AI). Data dimanfaatkan machine learning sebagai kode untuk komputasi tradisional. Cara lain untuk memperoleh kecerdasan dalam mesin bisa melalui pemrograman logis, penalaran induktif berdasarkan aturan dasar dan sebagainya. Dengan demikian machine learning dapat dianggap sebagai salah satu pendekatan menuju kecerdasan buatan. Berdasarkan sifat dari berbagai masalah yang ada dan kelimpahan data untuk masalah itu, wajar saja bahwa machine learning merupakan pendekatan untuk mencapai Artificial Intelligence (AI).

Machine Learning diibaratkan seperti aplikasi artificial intelligence yang menyediakan sistem kinerja secara otomatis serta belajar memperbaiki diri dari pengalaman tanpa diprogram secara eksplisit.

Pembelajaran mesin berfokus pada pengembangan program komputer yang bisa mengakses data dan menggunakan untuk belajar sendiri. Proses pembelajaran dari machine learning dimulai dengan observasi data, contohnya pengalaman langsung, atau intruksi untuk mencari pola data dan membuat keputusan yang lebih baik dimasa depan berdasarkan contoh tersebut. Tujuan utamanya adalah membiarkan komputer belajar secara otomatis tanpa intervensi atau bantuan manusia dan menyesuaikan aktivitas yang sesuai. Dengan adanya teknologi komputasi, machine learning saat ini tidak seperti machine learning di masa lalu.

Sebenarnya metode machine learning sudah ada sejak lama, kemampuannya secara otomatis menggunakan perhitungan matematis yang kompleks ke data besar dan yang lebih cepat merupakan perkembangan terakhir. Masih banyak lagi situs web dan perangkat modern yang mungkin besar dan berisi beberapa model Machine Learning yang mungkin tidak kita sadari. Model dari Machine Learning tersebut digunakan untuk melakukan klasifikasi atau prediksi terhadap data baru yang memungkinkan kita untuk membuat atau mendukung pengambilan keputusan. Berikut adalah beberapa contoh aplikasi pembelajaran mesin yang dipublikasikan secara luas:

Berikut adalah beberapa contoh aplikasi pembelajaran mesin yang dipublikasikan secara luas:

- Mobil Google yang sangat hyped dan self-driving. Inti pembelajaran mesin.
- Penawaran rekomendasi online seperti Amazon dan Netflix. Aplikasi belajar mesin untuk kehidupan sehari-hari.

- Mengetahui apa yang pelanggan katakan tentang Anda di Twitter. Pembelajaran mesin dikombinasikan dengan pembuatan aturan linguistik.
- Deteksi penipuan Salah satu kegunaan yang lebih jelas dan penting di dunia kita saat ini.

Teknologi Machine Learning meningkat dikarenakan faktor data mining dan analisis Bayesian lebih populer dari pada sebelumnya. Pengolahan komputasi yang lebih murah dan lebih bertenaga termasuk penyimpanan data yang terjangkau merupakan peningkatan. Semua hal ini secara cepat dan otomatis menghasilkan model yang dapat menganalisis data yang lebih besar dan lebih kompleks memberikan hasil yang lebih cepat dan akurat dalam skala yang sangat besar. Prediksi nilai tinggi bisa mengarah pada keputusan dan tindakan cerdas secara real-time tanpa campur tangan manusia. Salah satu kunci untuk menghasilkan gerakan cerdas secara real-time merupakan pembuatan model otomatis.

### **1.2.2 Sejarah Machine Learning**

Machine Learning bermula di awal abad 20, seorang penemu Spanyol, Torres y Quevedo, membuat sebuah mesin learning setelah ditemukannya komputer digital. Machine Learning pada dasarnya merupakan proses komputer untuk belajar dari data (Learn from data). Sejak pertama kali komputer diciptakan manusia sudah memikirkan bagaimana caranya agar komputer dapat belajar dari pengalamannya. Gagasan ini terbukti yaitu pada tahun 1952, Arthur Samuel menciptakan sebuah program, game of checkers, pada sebuah komputer IBM.

Program tersebut dapat mempelajari gerakan untuk memenangkan permainan checkers dan menyimpan gerakan tersebut kedalam memorinya. Istilah teknologi machine learning pada dasarnya adalah

proses komputer untuk belajar dari data (learn from data). Jika tidak ada data, maka komputer tidak akan bisa belajar apa-apa. Maka dari itu ketika kita ingin belajar machine learning, pasti akan terus berinteraksi dengan data. Semua pengetahuan machine learning pasti akan melibatkan data. Data bisa saja sama, akan tetapi algoritma dan pendekatannya berbeda-beda untuk mendapatkan hasil yang optimal.

### **1.2.3 Perkembangan Machine Learning**

Dengan berkembangnya teknologi kecerdasan buatan, muncul salah satu cabang kecerdasan buatan yang memperoleh banyak perhatian dari para peneliti yang disebut machine learning. Machine Learning berkerja dengan mempelajari teori agar komputer mampu “belajar” dari data, machine learning melibatkan berbagai disiplin ilmu seperti statistika, ilmu komputer, matematika dan bahkan neurologi. Algoritma machine learning yang menarik pada akhir-akhir ini adalah jaringan saraf tiruan, seperti namanya jaringan saraf tiruan terinspirasi dari cara kerja otak manusia.

Teknologi Machine Learning meningkat dikarenakan faktor data mining dan analisis Bayesian lebih populer dari pada sebelumnya. Pengolahan komputasi yang lebih murah dan lebih bertenaga termasuk penyimpanan data yang terjangkau merupakan peningkatan. Semua hal ini secara cepat dan otomatis menghasilkan model yang dapat menganalisis data yang lebih besar dan lebih kompleks memberikan hasil yang lebih cepat dan akurat dalam skala yang sangat besar. Prediksi nilai tinggi bisa mengarah pada keputusan dan tindakan cerdas secara real-time tanpa campur tangan manusia. Salah satu kunci untuk menghasilkan gerakan cerdas secara real-time merupakan pembuatan model otomatis.

Secara intuisi mencari inspirasi untuk membuat mesin mampu “berfikir” dari cara kerja otak adalah langkah yang bagus sama halnya

seperti ingin membuat alat yang mampu terbang dengan melihat cara kerja burung terbang. Pada model jaringan saraf tiruan yang disebut MLP atau multi-layer perceptron dikenal istilah layer, beberapa neuron tiruan dikelompokan menjadi satu layer kemudian layer satu menjadi input bagi layer yang lain. MLP yang sebenarnya yaitu sebuah model atau matematika yang terdiri dari komposisi-komposisi fungsi dari vektor ke vektor.

Model yang sudah dipilih dan digunakan algortima optimisasi berbasis gradien seperti gradient descent, berbagai masalah muncul ketika model jaringan saraf tiruan memiliki banyak layer, salah satu masalah yang terkenal disebut the vanishing gradient. Problem ini muncul karena jaringan saraf tiruan dengan banyak layer sebenarnya adalah fungsi yang terdiri dari banyak komposisi fungsi sehingga ketika menghitung gradien terhadap parameter dari fungsi tersebut, kita harus menerapkan aturan rantai yang menyebabkan gradien parameternya bernilai kecil sehingga algoritma gradient descent berjalan lambat.

#### **1.2.4 Bagaimana Machine Learning Bekerja**

Terdapat dua jenis Teknik pada Machine learning yang pertama Supervised Learning, yaitu yang melatih model pada data input dan output yang diketahui sehingga dapat memprediksi keluaran masa depan. Kemudian yang kedua adalah Unsupervised Learning, yang menemukan pola tersembunyi atau struktur intrinsik pada data masukan. Penggunaan metode Machine Learning dalam beberapa tahun terakhir telah berkembang di mana-mana dalam kehidupan sehari-hari. Machine Learning bukan hal baru dalam lanskap ilmu komputer. Machine Learning lebih mengaitkan proses struktural dimana setiap bagian untuk

menciptakan versi mesin yang lebih baik. Berikut adalah penjelasan dari Supervised Learning dan Unsupervised Learning :

### 1. Supervised Learning

Pembelajaran mesin yang diawasi menciptakan model yang melancarkan prediksi berdasarkan bukti adanya ketidakpastian. Pembelajaran pada algoritma supervised memerlukan seperangkat data masukan dan tanggapan yang diketahui terhadap data (output) dan melatih model untuk menghasilkan prediksi yang masuk akal untuk respon terhadap data baru. Algoritma pembelajaran ini digunakan jika Anda ingin mengetahui data output yang ingin Anda prediksi. Pembelajaran ini diawasi menggunakan teknik klasifikasi dan regresi untuk mengembangkan model prediktif.

Teknik dalam klasifikasi berfungsi untuk memprediksi respons diskrit misalnya pada email, apakah email yang masuk itu asli atau spam, atau apakah tumor itu kanker atau tidak. Model klasifikasi mengklasifikasikan data masukan ke dalam kategori tersebut. Aplikasi yang umum termasuk pencitraan medis. Misalnya aplikasi untuk pengenalan tulisan, maka anda harus menggunakan klasifikasi untuk mengenali huruf dan angka.

Jika Anda 17ect melakukannya, Anda memiliki landasan yang dapat Anda gunakan pada satu dataset ke dataset yang akan dicoba lagi selanjutnya. Anda 17ect mengisi waktu seperti mempersiapkan data lebih lanjut dan memperbaiki hasilnya nanti, begitu Anda lebih percaya diri. Dalam pengolahan citra dan penglihatan 17ector17r, 17ector pengenalan pola tanpa pemeriksaan digunakan untuk deteksi objek dan segmentasi. Algoritma yang umum mengadakan klasifikasi yang meliputi dukungan mesin 17ector (SVM).

## 2. Unsupervised Learning

Unsupervised Learning belajar dengan menemukan pola tersembunyi atau struktur intrinsik dalam data. Ini digunakan untuk menarik kesimpulan dari kumpulan data yang terdiri dari data masukan tanpa respon berlabel. Clustering adalah teknik belajar tanpa pengamatan yang umum. Ini digunakan untuk analisis data eksplorasi dalam menemukan pola atau pengelompokan tertutup dalam data. Aplikasi untuk analisis cluster meliputi analisis urutan gen, riset pasar dan pengenalan objek.

Misalnya, jika sebuah perusahaan telepon seluler ingin mengoptimalkan lokasi di mana mereka membangun menara telepon seluler, mereka dapat menggunakan pembelajaran mesin untuk memperkirakan jumlah kelompok orang yang bergantung pada menara mereka. Telepon hanya bisa berbicara dengan satu menara sekaligus, sehingga tim menggunakan algoritma pengelompokan untuk merancang peletakan menara seluler terbaik dalam mengoptimalkan penerimaan sinyal bagi kelompok dan dari pelanggan mereka. Beberapa algoritma Unsupervised clustering meliputi k-means dan k-medoids, hirarki clustering, model campuran Gaussian, model Markov tersembunyi, peta pengorganisasian sendiri, clustering fuzzy c-means dan clustering subtraktif.

### 1.2.5 Peran Data Dalam Machine Learning

Machine Learning tanpa adanya data maka dia bukan apa-apa. Yang berartikan semua aplikasi Machine Learning membutuhkan Data sebagai bahan training dan untuk di analisa sehingga mampu mengeluarkan Output. Machine learning agar dapat bekerja maka ia harus membutuhkan Data untuk "latihan" atau proses training, hasil dari proses training nantinya akan diuji atau ditesting dengan data yang sama atau berbeda.

Misalnya Kita membuat machine learning untuk mengenali object Nasi Padang (Nasi Pada recognition system), maka untuk melakukan training kita sediakan koleksi ratusan, ribuan bahkan jutaan data gambar nasi padang. setelah setelai, baru kita test hasil latihan atau model itu dengan menginputkan nasi padang (sejenis) dan kita juga test dengan memasukan foto object lain seperti Mobil, Gunung, komputer dll. Dari hasil testing nantinya dapat diketahui apakah model hasil training berhasil mengklasifikasikan Nasi Padang. Pada umumnya Output dari aplikasi machine learning yaitu berupa Prekdiksi beserta label tingkat kepercayaanya. Misalnya jika input gambar Nasi padang pada sistem maka sistem nantinya akan memberikan Output seperti ini:

- Nasi Padang 0.99
- Nasi pecel 0.78
- Karedok 0.58
- Nasi Kucing 0.40
- Nasi kamvret 0.10

Hasil atau output dari No 1 adalah output yang kita harapkan yaitu Nasi Padang dan juga tingkat kepercayaannya mencapai 0.99 yang merupakan paling tinggi. Skala yang digunakan dalam machine learning adalah 0 - 1.

Data pada machine Learning dapat berupa apa saja. Yaitu berupa table seperti data pada Ms Excel, atau berupa gambar seperti yang pada Tungle atau pada Google Image atau apa dalam bentuk apa saja yang memang disiapkan khusus untuk Input machine learning.

Secara matematis sederhana, rumus umum Machine leaning adalah sebagai berikut :

$$Y = f(X)$$

Dimana:

$Y$  = Output,

$X$  = Input,

$f$  = fungsi (function)

Dimana Machine Learning itu memenuhi kaidah secara singkatnya:

$Output = f(Input)$

Output ( $Y$ ) dari machine learning adalah hasil dari pengolahan fungsi ( $f$ ) terhadap Input ( $X$ )

Contohnya pada aplikasi Tungle, Jika saya kasih input berupa gambar/foto Nasi Padang, maka fungsi-fungsi/Algorithm dari tungle diharapkan akan memberitahu kalo foto itu adalah Nasi Padang. Kurang lebih seperti itulah cara kerja data pada machine learning.

### **1.2.6 Konsep Dasar Machine Learning**

Konsep tersebut meliputi kemampuan suatu individu dalam meningkatkan kecerdasan tersebut untuk belajar tanpa terkecuali pada sebuah mesin. Mesin yang mampu belajar, akan meningkatkan produktivitas manusia. Maka ia juga akan memiliki kekuatan yang mungkin tidak dimiliki mesin lainnya.

a. Manfaat pembelajaran mesin dalam memprediksi,

Apabila anda hanya mengenal wajah teman anda dalam gambar, berarti anda tidak memakai model pembelajaran mesin. Inti pembelajaran mesin adalah meramalkan hal-hal berdasarkan pola dan faktor lain yang telah dilatih. Apa yang menjadikan sesuatu itu jadi lebih mudah untuk di kuasai. Pengenalan dilakukan dengan cara yang simple tanpa menghabiskan banyak waktu.

b. Pembelajaran mesin membutuhkan pelatihan,

Untuk membuat model maka harus memberi tahu model pembelajaran mesin seperti apa yang akan diprediksikannya. Pikirkan bagaimana anak manusia belajar. Ini adalah penyederhanaan yang berlebihan sedikit karena saya meninggalkan bagian dimana Anda juga harus mengatakan bahwa itu bukan pisang dan tunjukkan berbagai jenis pisang, warna yang berbeda, gambar dari perspektif dan sudut yang berbeda, dll.

c. Ketepatan 80% dianggap sukses

Teknologi ini tidak mengetahui dimana platform pembelajaran mesin akan mencapai akurasi 100% dengan mengidentifikasi pisang dalam gambar. Tapi tidak apa-apa, ternyata manusia juga tidak 100% akurat. Aturan yang dikatakan dalam industri ini adalah bahwa model dengan akurasi 80% adalah sebuah kesuksesan. Misalnya jika kita memikirkan betapa bergunanya untuk mengidentifikasi 800.000 gambar dengan benar di koleksi kita, sementara MUNGKIN TIDAK mendapatkan 200.000 yang benar, maka kita masih menyimpan 80% dari waktu kita. Itu merupakan perspektif nilai yang sangat besar. Jika saya bisa melambaikan tongkat sihir dan meningkatkan produktivitas kita sebanyak itu, kita akan mendapatkan banyak uang. Nah, ternyata kita bisa melakukannya dengan mesin pembelajaran.

d. Pembelajaran Machine (mesin) berbeda dengan AI

Kebanyakan orang mengatakan hal ini sama dan sangat sederhana. Namun, kenyataan yang di dapat dari para ahli, ini memiliki perbedaan. Perbedaannya sebagai berikut:

- AI (Artificial Intelligence), Yang berarti komputer lebih baik dari manusia untuk melakukan tugas tertentu. Seperti robot yang bisa membuat keputusan berdasarkan banyaknya masukan, tidak seperti

Terminator atau C3PO. Sebenarnya istilah yang sangat luas itu tidak terlalu berguna.

- ML (Machine Learning), Adalah metode untuk mencapai AI. Yang membuat prediksi tentang sesuatu berdasarkan pelatihan dari kumpulan data parsing. Ada berbagai macam cara yang berbeda di platform ML yang dapat menerapkan perangkat pelatihan untuk memprediksi sesuatu.
- NN (Neural Network), Neural Network atau Jaringan syaraf tiruan adalah salah satu cara model pembelajaran mesin untuk memprediksi sesuatu. Jaringan saraf bekerja sedikit seperti otak Anda, dengan menyesuaikan diri dan banyak berlatih untuk memahaminya. Anda akan menciptakan lapisan simpul yang sangat dalam.
- Memberikan struktur yang jelas terhadap AI, Kebanyakan model Machine learning bergantung pada manusia untuk melakukan apa yang akan dikerjakan mesin pembelajaran. Inilah yang membuat anda selalu bergantung dengan teknologi tersebut, karena sesuatu yang ingin anda kerjakan. Dan bahkan saat Anda memberikan instruksi yang jelas, biasanya itu masih saja salah. Anda harus begitu eksplisit dengan sistem ini sehingga kesempatan itu tiba-tiba menjadi lebih mudah. Bahkan halaman web sederhana yang menunjukkan sebuah kotak dengan sebuah kata di dalamnya mengharuskan Anda untuk memberi tahu persis di mana kotak itu muncul, seperti apa bentuknya, warna apa itu, bagaimana cara bekerja pada peramban yang berbeda, bagaimana ditampilkan dengan benar pada perangkat yang berbeda. dll. Ada banyak cara menghalangi jaringan syaraf yang sangat dalam untuk mengambil alih dunia dan mengubah kita agar terlihat lebih

kuat, terutama karena semua yang akan kita lakukan tidak segampang dan semudah yang kita pikirkand.

### **1.2.7 Aplikasi Machine learning**

Data bisa saja sama, namun untuk pendekatan terhadap algoritmanya berbeda-beda dalam hal mendapatkan hasil yang optimal. Berikut merupakan contoh aplikasi pembelajaran mesin:

- Pada Penelusuran web: Terdapat laman peringkat berdasarkan apa yang anda klik
- Pada Biologi komputasional: Sebagai obat desain rasional di komputer berdasarkan eksperimen masa lalu.
- Pada Keuangan: Untuk menetapkan siapa yang akan mengirim kartu kredit yang ditawarkan. Evaluasi risiko pada penawaran kredit dan bagaimana cara memutuskan dimana menginvestasikan uangnya.
- Pada E-commerce: Untuk memprediksi customer churn. Apakah transaksi itu salah atau tidak.
- Pada Eksplorasi ruang angkasa: Untuk menyelidiki ruang angkasa dan astronomi radio.
- Pada Robotika: Bagaimana menangani ketidakpastian di lingkungan baru, seperti otonom dan Mobil self-driving.
- Pada Pengambilan informasi: Mengajukan pertanyaan melalui database di seluruh web.
- Pada Jaringan sosial: Data tentang hubungan dan preferensi. Mesin belajar mengekstrak nilai dari data.
- Pada Debugging: Digunakan dalam masalah ilmu komputer seperti debugging.

Dari model yang telah didapatkan, maka kita dapat melakukan prediksi yang dibedakan menjadi dua macam, tergantung tipe keluarannya.

Apabila hasil prediksi bersifat diskrit, maka ini dinamakan proses klasifikasi. Contoh teknik untuk pengaplikasian machine learning adalah supervised learning. Seperti yang sudah dibahas dan dijelaskan pada materi sebelumnya, machine learning tanpa data ini tidak akan bisa bekerja. Maka dari itu hal yang pertama kali disiapkan adalah data. Data pada umumnya akan dibagi menjadi 2 kelompok, yaitu data training dan data testing.

### **1.2.8 Dampak Machine Learning**

Penerapan machine learning pada teknologi saat ini, kebanyakan orang mungkin telah merasakan dampaknya sekarang. Pada perkembangan teknologi machine learning memiliki dampak yang saling bertolak belakang yaitu dampak negatif dan dampak positif. Ini yang akan memberikan masukan yang berdampak buruk dan baiknya, tergantung terhadap orang yang menilainya. Akan tetapi semua ini tidak selalu berjalan dengan mulus.

- **Dampak Positif**

Dampak positif pada teknologi machine learning adalah mendapat kesempatan bagi para wirausaha dan praktisi teknologi untuk terus berkreasi dalam mengembangkan machine learning. Terutama untuk membantu aktivitas manusia sebagai sesuatu yang menguntungkan. Itulah salah satu dampak positif dari machine learning. Salah satu contohnya dapat digunakan untuk pengecekan ejaan untuk tiap bahasa yang ada dalam microsoft Word. Pengecekan manual akan menghabiskan waktu untuk beberapa hari, juga memerlukan banyak tenaga untuk mendapatkan penulis yang sempurna. Namun, dengan bantuan fitur pengecekan tersebut, maka

secara real-time kesalahan yang terjadi saat pengetikan kita bisa langsung melihatnya.

- Dampak Negatif

Dampak negative yang harus diwaspada dan di khawatirkan yaitu adanya pengurangan tenaga kerja. Kenapa demikian? Karena pekerjaan yang seharusnya di kerjakan oleh banyak orang, sekarang telah digantikan oleh alat teknologi yang disebut sebagai machine learning. Hal tersebut merupakan suatu permasalahan yang akan kita hadapi. Ditambah dengan ketergantungan terhadap teknologi yang semakin banyak dan berkembang di kehidupan kita. Kadang manusia lebih nyaman dengan perkembangan teknologi sekarang ini seperti gadget.

### 1.3 Pengantar Neural Network

Cabang ilmu kecerdasan buatan cukup luas, dan erat kaitannya dengan disiplin ilmu yang lainnya. Hal ini bisa dilihat dari berbagai aplikasi yang merupakan hasil kombinasi dari berbagai ilmu. Seperti halnya yang ada pada peralatan medis yang berbentuk aplikasi. Sudah berkembang bahwa aplikasi yang dibuat merupakan hasil perpaduan dari ilmu kecerdasan buatan dan juga ilmu kedokteran atau lebih khusus lagi yaitu ilmu biologi.

Jaringan Syaraf Tiruan atau Artificial Neural Network (NN) adalah teknik dalam ML yang menirukan syaraf manusia yang merupakan bagian fundamental dari otak. NN terdiri atas lapis masukan (input layer) dan lapis keluaran (output layer). Setiap lapis terdiri atas satu atau beberapa unit neuron yang mempunyai sebuah fungsi aktivasi yang menentukan keluaran dari unit tersebut. Kita bisa menambahkan lapis tersembunyi (hidden layer) untuk menambah kemampuan dari NN tersebut.

NN bisa dilatih dengan menggunakan data training. Semakin banyak data training maka akan semakin bagus unjuk kerja dari NN tersebut. Namun, kemampuan NN juga terbatas pada jumlah lapisan, semakin banyak jumlah lapisan semakin tinggi kapasitas NN tersebut. Semakin banyak lapisan juga membawa kekurangan yaitu semakin banyaknya jumlah iterasi atau training yang dibutuhkan. Untuk mengatasi hal ini, dikembangkanlah teknik Deep Learning. Beberapa aplikasi NN antara lain untuk Principal Component Analysis, regresi, klasifikasi citra, dan lain-lain.

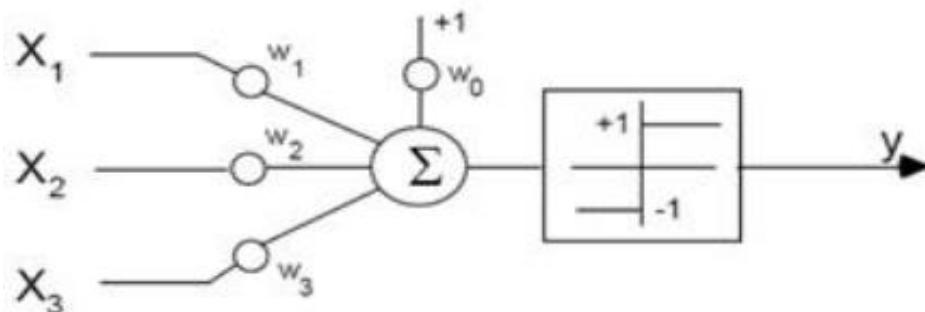
Neural Network merupakan kategori ilmu Soft Computing. Neural Network bekerja dengan mengadopsi prinsip dari kemampuan otak manusia yang mampu memberikan stimulasi/rangsangan, melakukan proses, dan memberikan output. Output diperoleh dari variasi stimulasi dan proses yang terjadi di dalam otak manusia. Kemampuan manusia dalam memproses informasi merupakan hasil kompleksitas proses di dalam otak. Contohnya yang terjadi pada anak-anak, mereka mampu belajar untuk melakukan pengenalan meskipun mereka tidak mengetahui algoritma apa yang digunakan. Kekuatan komputasi yang luar biasa dari otak manusia ini merupakan sebuah keunggulan di dalam kajian ilmu pengetahuan.

Fungsi dari Neural Network diantaranya adalah:

- Pengklasifikasian pola
- Mengatur pola yang didapat dari input ke dalam pola baru pada output
- Penyimpan pola yang akan dipanggil kembali
- Memetakan pola-pola yang sejenis
- Pengoptimasi permasalahan
- Prediksi

### 1.3.1 Sejarah Neural Network

Teknologi pengembangan Neural Network sudah ada sejak tahun 1943 ketika Warren McCulloch dan Walter Pitts memperkenalkan perhitungan model neural network yang pertama kalinya. Mereka melakukan kombinasi beberapa processing unit sederhana bersama-sama yang mampu memberikan peningkatan secara keseluruhan pada kekuatan komputasi. Kemudian penelitian dilanjutkan dan dikerjakan oleh Rosenblatt pada tahun 1950, dimana dia berhasil menemukan sebuah two-layer network, yang disebut sebagai perceptron. Perceptron memungkinkan untuk pekerjaan klasifikasi pembelajaran tertentu dengan penambahan bobot pada setiap koneksi antar-network.



Gambar 1. 2 Perception

Pencapaian keberhasilan dari perceptron dalam pengklasifikasian pola tertentu ini tidak sepenuhnya sempurna, masih ditemukan juga beberapa keterbatasan didalamnya. Perceptron tidak mampu untuk menyelesaikan permasalahan XOR (exclusive-OR). Penilaian pada keterbatasan neural network ini membuat penelitian di bidang ini sempat mati selama kurang lebih 15 tahun. Meski demikian, perceptron berhasil menjadi sebuah dasar untuk penelitian-penelitian selanjutnya di bidang neural network.

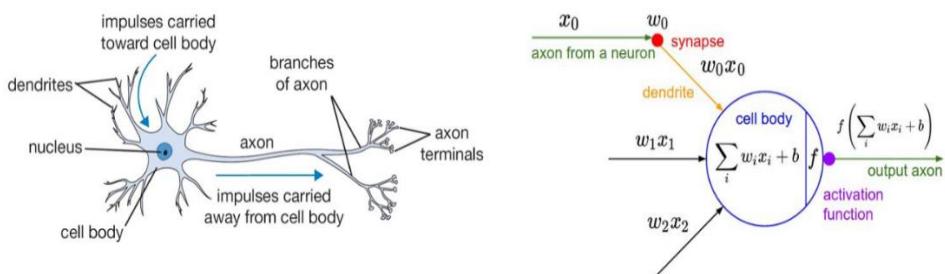
Pengujian dan pengkajian pada teknologi neural network mulai berkembang lagi selanjutnya di awal tahun 1980-an. Banyak peneliti yang

berhasil menemukan bidang interest baru pada domain ilmu neural network. Penelitian yang terakhir dilakukan adalah mesin Boltzmann, jaringan Hopfield, model pembelajaran kompetitif, multilayer network dan teori model resonansi adaptif. Pada saat ini, Neural Network sudah dapat diterapkan pada beberapa task, diantaranya classification, recognition, approximation, prediction, clusterization, memory simulation dan banyak task-task berbeda yang lainnya, dimana jumlahnya semakin bertambah seiring berjalannya waktu.

### 1.3.2 Konsep Neural Network

Dasar pembelajaran Neural Network dimulai dari otak manusia, dimana otak memuat sekitar 10 neuron. Neuron ini berfungsi memproses setiap informasi yang masuk. Pada satu buah neuron memiliki satu akson dan minimal satu dendrit. Setiap sel syaraf terhubung dengan syaraf lain jumlahnya mencapai sekitar 10 sinapsis. Setiap sel dan masing-masing sel tersebut saling berinteraksi satu sama lain yang menghasilkan kemampuan tertentu pada kerja otak manusia.

Neural network merupakan model yang terinspirasi oleh bagaimana neuron dalam otak manusia bekerja. Tiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut. Ilustrasi neuron dengan model matematisnya dapat dilihat pada gambar berikut pada gambar 1.3 :



Gambar 1. 3 Struktur Neuron Pada Otak Manusia

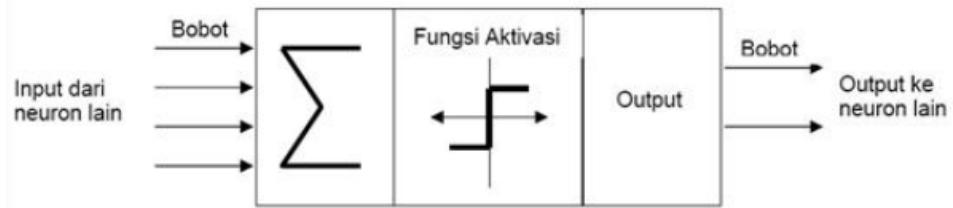
Dari gambar 1.3, bisa dilihat ada beberapa bagian dari otak manusia, yaitu:

- Dendrit (Dendrites) berfungsi untuk mengirimkan impuls yang diterima ke badan sel syaraf.
- Akson (Axon) berfungsi untuk mengirimkan impuls dari badan sel ke jaringan lain.
- Sinapsis berfungsi sebagai unit fungsional di antara dua sel syaraf.

Proses yang terjadi pada otak manusia yaitu neuron menerima impuls dari neuron lain melalui dendrit dan mengirimkan sinyal yang dihasilkan oleh badan sel melalui akson. Kemudian akson dari sel syaraf ini akan bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lain dengan cara mengirimkan impuls melalui sinapsis. Sinapsis merupakan sebuah unit fungsional antara 2 buah sel syaraf, misal A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi adalah dendrit dari neuron B. Kekuatan dari sinapsis bisa saja menurun atau meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya. Impuls-impuls sinyal (informasi) ini akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering disebut dengan nilai ambang (threshold).

### 1.3.3 Struktur Neural Network

Pada otak manusia struktur neuron akan melakukankerja yang dijelaskan di atas, maka konsep dasar pembangunan neural network buatan (Artificial Neural Network) terbentuk. Cara kerja 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 atau error, dan juga parallel processing.



Gambar 1. 4 Struktur NN

Keunikan dari NN dapat dilihat dari pola hubungan antar neuron, metode penentuan bobot dari tiap koneksi, dan fungsi aktivasinya. Pada gambar di atas menjelaskan struktur NN 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 dibuat dengan menggunakan banyak node atau unit yang dihubungkan oleh link secara langsung. Caranya link dari node yang satu dihubungkan dengan node yang lainnya yang berfungsi untuk melakukan propagasi aktivasi dari unit pertama ke unit selanjutnya. Setiap link memiliki bobot numerik. Bobot ini menentukan kekuatan serta penanda dari sebuah konektivitas. Proses pembelajaran pada Neural Network dimulai dari input atau masukan yang diterima oleh neuron beserta dengan nilai bobot dari tiap-tiap input yang ada.

Selanjutnya setelah masuk ke dalam neuron, nilai input yang ada akan dijumlahkan oleh suatu fungsi perambatan (summing function), yang bisa dilihat seperti pada di gambar dengan lambang sigma ( $\sum$ ). Hasil dari penjumlahan selanjutnya 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

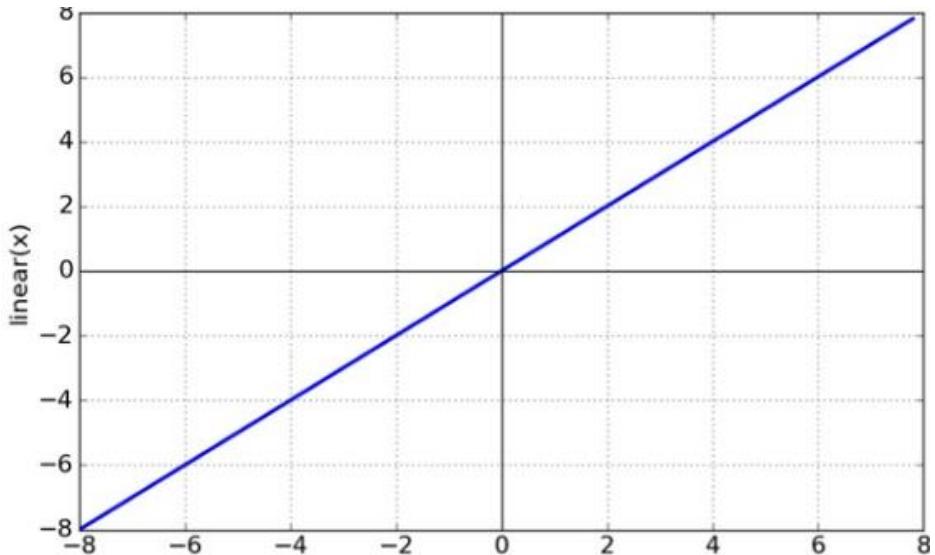
diaktifkan. Setelah aktif, neuron akan mengirimkan nilai output melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya. Proses ini akan terus berulang pada input-input selanjutnya.

NN terdiri dari banyak neuron di dalamnya. Neuron-neuron ini akan dikelompokkan ke dalam beberapa layer. Pada setiap layer NN terdapat neuron yang dihubungkan pada neuron di layer lainnya. Namun proses ini tentunya tidak berlaku pada layer input dan output, tapi hanya layer yang berada di antaranya. Informasi yang diterima dari proses di layer input dilanjutkan ke layer-layer dalam NN secara satu persatu hingga mencapai layer terakhir/layer output. Terdapat hidden layer yaitu layer yang terletak di antara input dan output. Akan tetapi tidak semua NN memiliki hidden layer, ada juga yang hanya terdapat layer input dan output saja.

#### **1.3.4 Activation Function**

Apakah yang dimaksud activation function pada Teknik neural network? Activation function sesuai dengan Namanya aktivasi fungsi befungsi untuk menentukan apakah neuron tersebut harus “aktif” atau tidak berdasarkan dari weighted sum dari inputan. Pada umumnya umum terdapat 2 jenis activation function pada neural network yaitu, Linear dan Non-Linear Activation function.

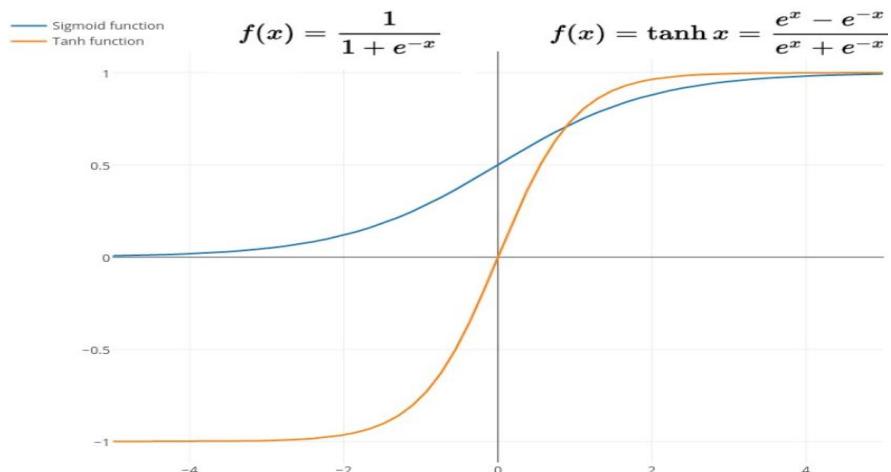
a. Linier Function



Gambar 1. 5 Linier Function

Activation function dapat dikatakan secara “default” dari sebuah neuron adalah Linear. Apabila sebuah neuron menggunakan linear function, maka keluaran dari neuron tersebut merupakan weighted sum dari input + bias.

b. Sigmoid and Tanh Function (Non-Linear)



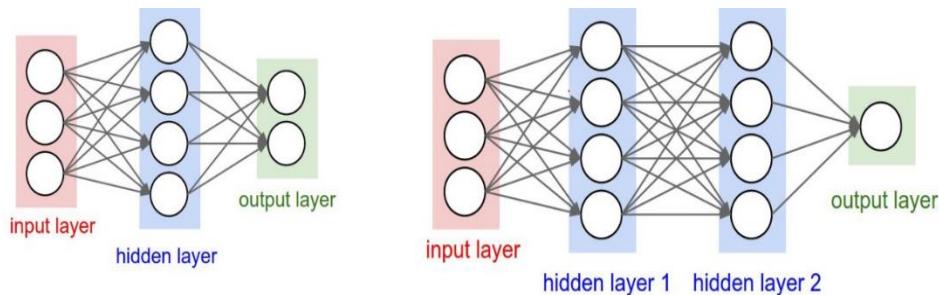
Gambar 1. 6 Sigmoid and Tanh Function

Pada Neural Network aktivasi Sigmoid function mempunyai rentang antara 0 hingga 1 sedangkan rentang dari Tanh adalah -1 hingga 1. Kedua fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data. Namun terdapat kelemahan dari kedua fungsi ini,

c. ReLU (Non-Linear

Aktivasi ReLU pada dasarnya melakukan “threshold” dari 0 hingga infinity. ReLU juga berfungsi untuk menutupi kelemahan yang dimiliki oleh Sigmoid dan Tanh. Pada dasarnya masih banyak lagi activation function yang lain pada neurak network, namun beberapa fungsi yang saya sebutkan diatas merupakan fungsi yang sering digunakan.

### 1.3.5 Neural Network Architectures



Gambar 1. 7 Arsitektur Neural Network

Arsitektur pada gambar 1.7 biasanya disebut sebagai Multi Layer Perceptron (MLP) atau Fully-Connected Layer. Yaitu arsitektur pertama mempunyai 3 buah neuron pada Input Layer dan 2 buah node Output Layer. Diantara Input dan Output, terdapat 1 Hidden Layer dengan 4 buah neuron. Sedangkan spesifikasi dari Weight dan Activation function adalah sebagai berikut:

a. Weight and Bias

Seluruh neuron pada MLP saling terhubung dan ditandai dengan tanda panah pada gambar diatas. Kemudian seluruh koneksi memiliki weight

yang nantinya nilai dari tiap weight akan berbeda-beda. Pada lapisan hidden layer dan output layer akan memiliki tambahan “input” yang biasa disebut dengan bias (Tidak disebutkan pada gambar diatas).

Maka pada arsitektur pertama terdapat  $3 \times 4$  weight ditambah 4 bias dan  $4 \times 2$  weight dengan 2 bias. Totalnya terdapat 26 parameter yang pada proses training akan mengalami perubahan untuk mendapatkan hasil yang terbaik. Sedangkan pada arsitektur yang kedua terdapat 41 parameter.

#### b. Activation Function

Neuron yang terdapat pada input layer tidak memiliki activation function, akan tetapi neuron yang terdapat pada hidden layer dan output layer memiliki activation function yang kadang berbeda tergantung daripada data atau problem yang kita miliki. Training a Neural Network

#### c. Training A Neural Network

Pada Teknik Supervised Learning yang menggunakan Neural Network, pada umumnya Learning terdiri dari 2 tahap, yaitu training dan evaluation. Namun kadang terdapat tahap tambahan yaitu testing, namun sifatnya tidak wajib. Pada proses training setiap weight dan bias pada tiap neuron akan diupdate terus menerus hingga output yang dihasilkan sesuai dengan harapan. Setiap iterasi akan dilakukan proses evaluation yang biasanya digunakan untuk menentukan kapan proses training harus dihentikan (stopping point). Secara garis besar proses training pada NN terbagi menjadi 2 tahap yaitu forward pass dan backward pass.

#### d. Forward Pass

Teknik Forward pass yang biasa juga disebut forward propagation adalah proses dimana kita membawa data pada input melewati tiap neuron pada hidden layer sampai kepada output layer yang nanti akan dihitung errornya.

$$dot_j = \sum_i^3 w_{ji}x_i + b_j$$

$$h_j = \sigma(dot_j) = \max(0, dot_j)$$

Gambar 1. 8 Persamaan Forward Pass

Rumus pada gambar 1.8 adalah contoh teknik forward pass pada arsitektur pertama (lihat gambar arsitektur diatas) yang menggunakan ReLU sebagai activation function. Dimana i adalah node pada input layer (3 node input), j adalah node pada hidden layer sedangkan h adalah output dari node pada hidden layer.

#### e. Backward Pass

Error yang terjadi dan terdapat pada forward pass akan digunakan untuk mengupdate setiap weight dan bias dengan learning rate tertentu. Semua proses diatas akan dilakukan berulang-ulang sampai didapatkan nilai weight dan bias yang dapat memberikan nilai error sekecil mungkin pada output layer (pada saat forward pass) sehingga dapat menghasilkan model yang bagus.

### 1.4 Pengantar Deep Learning

#### 1.4.1 Penjelasan Deep Learning

Apa itu Deep Learning? Oke kita kenalan dulu ya! Karena ada pepatah yang mengatakan seperti ini, “Tak kenal maka sayang”. Deep Learning dapat diartikan Pembelajaran Dalam atau sering dikenal dengan istilah Pembelajaran Struktural Mendalam (Deep Structured Learning) atau Pembelajaran Hierarki (Hierarchical learning) adalah salah satu cabang dari ilmu pembelajaran mesin (Machine Learning) yang di dalamnya

terdiri algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi non-linear yang ditata berlapis-lapis dan mendalam.

Algoritma dalam Deep Learning dapat digunakan baik untuk kebutuhan pembelajaran terarah (supervised learning), pembelajaran tak terarah (unsupervised learning) dan semi-terarah (semi-supervised learning) dalam berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya. Teknik Deep Learning dikatakan sebagai Deep atau dalam karena struktur dan jumlah jaringan saraf dalam algoritmanya sangat banyak yang bisa mencapai hingga ratusan lapisan.

Teknik Deep Learning merupakan salah satu jenis algoritma jaringan saraf tiruan yang menggunakan metadata sebagai proses input dan mengolahnya menggunakan sejumlah lapisan tersembunyi (hidden layer) transformasi non linier dari data masukan untuk menghitung nilai output. Algoritma yang ada pada Deep Learning memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis.

Hal ini berarti algoritma yang dimilikinya secara otomatis dapat menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Kemudian algoritma pada deep learning juga dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (supervised), tanpa pengawasan (unsupervised), dan semi terawasi (semi supervised).

Dalam jaringan saraf tiruan tipe Deep Learning setiap lapisan tersembunyi bertanggung jawab untuk melatih serangkaian fitur unik berdasarkan output dari jaringan sebelumnya. Deep Learning akan menjadi

semakin kompleks dan bersifat abstrak ketika jumlah lapisan tersembunyi (hidden layer) semakin bertambah banyak. Jaringan saraf yang dimiliki oleh Deep Learning terbentuk dari hierarki sederhana dengan beberapa lapisan hingga tingkat tinggi atau banyak lapisan (multilayer). Berdasarkan hal itulah Deep Learning dapat digunakan untuk memecahkan masalah kompleks yang lebih rumit dan terdiri dari sejumlah besar lapisan transformasi non linier.

Deep Learning diproses menggunakan teknik tertentu seperti Restricted Boltzmann Machine (RBM) untuk mempercepat proses pembelajaran dalam Neural Network yang menggunakan lapis yang banyak. Dengan terciptanya Deep Learning, waktu yang dibutuhkan untuk training akan semakin sedikit karena masalah hilangnya gradien pada propagasi balik akan semakin rendah. Beberapa jenis teknik Deep Learning antara lain yaitu Deep Auto Encoder, Deep Belief Nets, Convolutional NN, dan lain lain.

Dalam istilah praktis, deep learning merupakan bagian dari machine learning. Sebuah model machine learning perlu 'diberitahu' untuk bagaimana ia menciptakan prediksi akurat, dengan terus diberikan data. Sementara model deep learning dapat mempelajari metode komputasinya sendiri, dengan 'otaknya' sendiri, apabila diibaratkan. Sebuah model deep learning dirancang untuk terus menganalisis data dengan struktur logika yang mirip dengan bagaimana manusia mengambil keputusan. Untuk dapat mencapai kemampuan itu, deep learning menggunakan struktur algoritma berlapis yang disebut artificial neural network (ANN).

Dikutip dari Zendesk, desain ANN terinspirasi dari jaringan neural biologis dari otak manusia. Hal ini membuat mesin kecerdasannya menjadi jauh lebih tangguh dibandingkan model machine learning standar. Rumit

memang untuk memastikan model deep learning yang diciptakan tidak memberikan kesimpulan yang tidak tepat. Tapi ketika ia telah bekerja dengan benar, maka fungsi deep learning akan menjadi terobosan yang berpotensi menjadi tulang belakang sebuah kecerdasan buatan sebenarnya.

Data yang digunakan pada teknologi deep learning sangatlah penting, karena semakin banyak datanya, maka semakin banyak yang bisa dipahami model deep learning tersebut. Contoh dari penerapan model deep learning bisa dilihat dari AlphaGo-nya Google. Google menciptakan program komputer yang belajar bermain sebuah game sejenis catur dari China bernama Go. Tentunya, game ini membutuhkan pemikiran dan intuisi yang tajam untuk menang.

Dengan bermain melawan pemain Go profesional, deep learning AlphaGo mempelajari bagaimana ia bermain di tingkat yang belum terjamah sebelumnya dalam kecerdasan buatan. Hebatnya, apa yang dilakukannya tanpa instruksi apapun ketika melancarkan gerakan-gerakan spesifik. Saat pemain game AlphaGo berhasil mengalahkan sejumlah pemain Go 'nyata' dunia, dunia melihat bagaimana cerdasnya sebuah mesin yang bahkan bisa mengungguli manusia.

#### **1.4.2 Sejarah Deep Learning**

Pada tahun 2006, Geoffrey Hinton memperkenalkan salah satu varian jaringan saraf tiruan yang disebut deep belief nets, ide untuk men-train model jaringan saraf tiruan ini adalah dengan men-train dua layer kemudian tambahkan satu layer diatasnya, kemudian train hanya layer teratas dan begitu seterusnya. Dengan strategi ini kita dapat men-train model jaringan saraf tiruan dengan layer lebih banyak dari model-model sebelumnya. Paper ini merupakan awal populernya istilah deep learning untuk membedakan arsitektur jaringan saraf tiruan dengan banyak layer.

Setelah istilah deep learning populer, deep learning belum menjadi daya tarik yang besar bagi para peneliti karena jaringan saraf tiruan dengan banyak layer memiliki kompleksitas algoritma yang besar, sehingga membutuhkan komputer dengan spesifikasi tinggi, dan tidak efisien secara komputasi saat itu.

Hingga pada tahun 2009 Andrew Ng dkk memperkenalkan penggunaan GPU untuk deep learning melalui paper yang berjudul Large-scale Deep Unsupervised Learning using Graphics Processors. Dengan menggunakan GPU jaringan saraf tiruan dapat berjalan lebih cepat dibanding dengan menggunakan CPU. Dengan tersedianya hardware yang memadai perkembangan deep learning mulai pesat, dan menghasilkan produk-produk yang dapat kita nikmati saat ini seperti pengenal wajah, self-driving car, pengenal suara, dan lain lain.

### **1.4.3 Jenis Deep Learning**

Berikut adalah jenis-jenis pada deep learning berdasarkan pembelajarannya:

1. Deep Learning untuk Pembelajaran Tanpa Pengawasan (Unsupervised Learning): Deep Learning tipe ini digunakan pada saat label dari variabel target tidak tersedia dan korelasi nilai yang lebih tinggi harus dihitung dari unit yang diamati untuk menganalisis polanya.
2. Hybrid Deep Networks (Deep Learning gabungan): Pendekatan tipe ini bertujuan agar dapat dicapai hasil yang baik dengan menggunakan pembelajaran yang diawasi untuk melakukan analisis pola atau dapat juga dengan menggunakan pembelajaran tanpa pengawasan.

#### **1.4.4 Seberapa banyak jumlah lapisan tersembunyi (Hidden Layer) yang harus dipakai pada Deep Learning?**

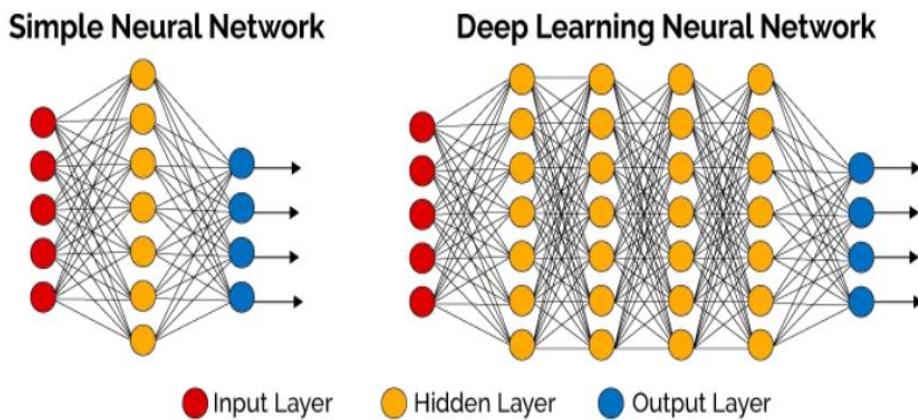
Deep Learning bekerja berdasarkan pada arsitektur jaringan dan prosedural optimal yang digunakan pada arsitektur. Setiap output dari lapisan per lapisan yang tersembunyi dapat dipantau dengan menggunakan grafik khusus yang dirancang untuk setiap output neuron. Kombinasi dan rekombinasi dari setiap neuron yang saling terhubung dari semua unit lapisan tersembunyi dilakukan menggunakan gabungan dari fungsi aktivasi. Prosedur-prosedur tersebut dikenal sebagai Transformasi Non Linier yang digunakan untuk prosedur optimal untuk menghasilkan bobot optimal pada setiap unit lapisan guna mendapatkan nilai target yang dibutuhkan.

Ketika dalam proses perancangan, apabila jumlah saraf yang ditambahkan sangat banyak, hal tersebut tidak akan pernah cocok untuk menyelesaikan setiap masalah. Persoalan dalam Deep Learning yang terpenting adalah jaringan sarafnya dilatih dengan cara penurunan gradien secara sederhana. Pada saat kita menambahkan lapisan jaringan yang semakin banyak, maka sebaliknya penurunan dari gradien semakin berkurang sehingga dapat mempengaruhi nilai outpunya.

#### **1.4.5 Perbedaan yang dimiliki oleh Neural Network dengan Deep Learning**

Jaringan Saraf Tiruan atau NN adalah jaringan saraf yang biasanya menggunakan jaringan seperti umpan maju (feed forward) atau recurrent network yang hanya memiliki 1 atau 2 lapisan tersembunyi. Namun, jika lapisan jaringan sarafnya lebih dari 2 layer ke atas atau bahkan mencapai ratusan lapisan itulah yang disebut sebagai Deep Learning.

Pada Jaringan Syaraf Tiruan arsitektur jaringan yang dimilikinya kurang kompleks dan membutuhkan lebih banyak informasi tentang data input sehingga dapat menentukan algoritma mana yang dapat digunakan. Dalam Jaringan Saraf Tiruan terdiri dari beberapa algoritma yaitu Model Hebb, Perceptron, Adaline, Propagasi Maju, dll. Sedangkan pada algoritma jaringan saraf Deep Learning tidak memerlukan informasi apapun terhadap data yang akan dipelajarinya, dan algoritmanya dapat secara mandiri melakuan tuning (penyetelan) dan pemilihan model yang paling optimal.



Gambar 1. 9 Perbedaan Arsitektur Neural Network dan Deep Learning

## **BAB II**

---

### **PENGENALAN CNN, R-CNN, FAST R-CNN, FASTER R-CNN DAN YOLO**

#### **2.1 Pengantar Convolution Neural Network (CNN)**

Saat belajar data science atau artificial intelligence, terutama ketika kamu mulai mendalami tentang Deep Learning, mungkin kamu pernah mendengar istilah Convolutional Neural Network. Istilah yang kerap disingkat CNN atau ConvNet ini adalah sebuah kelas dalam Deep Neural Networks, yang biasanya digunakan untuk menganalisis gambaran visual.

Convolutional Neural Network merupakan salah satu jenis algoritme Deep Learning yang dapat menerima input berupa gambar, menentukan aspek atau obyek apa saja dalam sebuah gambar yang bisa digunakan mesin untuk mempelajari dan mengenali gambar, dan membedakan antara satu gambar dengan yang lainnya.

CNN adalah versi resmi dari multilayer perceptrons . Multilayer perceptrons biasanya berarti jaringan yang terhubung penuh, yaitu, setiap neuron dalam satu lapisan terhubung ke semua neuron di lapisan berikutnya. "Koneksi penuh" dari jaringan-jaringan ini membuat mereka rentan terhadap overfitting data. Cara khas regularisasi termasuk menambahkan beberapa bentuk pengukuran bobot untuk fungsi kerugian. CNN mengambil pendekatan berbeda terhadap regularisasi: mereka mengambil keuntungan dari pola hierarkis dalam data dan mengumpulkan pola yang lebih kompleks menggunakan pola yang lebih kecil dan lebih sederhana. Oleh karena itu, pada skala keterhubungan dan kompleksitas, CNN berada di ekstrem bawah.

Jaringan konvolusional diilhami oleh proses biologis dalam hal pola konektivitas antara neuron menyerupai organisasi korteks visual hewan. Neuron kortikal individual merespons rangsangan hanya di daerah terbatas bidang visual yang dikenal sebagai bidang reseptif. Bidang reseptif dari neuron yang berbeda sebagian tumpang tindih sehingga menutupi seluruh bidang visual.

CNN menggunakan pra-pemrosesan yang relatif sedikit dibandingkan dengan algoritma klasifikasi gambar lainnya. Ini berarti bahwa jaringan mempelajari filter yang dalam algoritma tradisional direkayasa oleh tangan Kemandirian ini dari pengetahuan sebelumnya dan upaya manusia dalam desain fitur adalah keuntungan utama.

Arsitektur CNN terbilang mirip dengan pola koneksi neuron atau sel saraf dalam otak manusia. CNN tercipta karena terinspirasi dengan Visual Cortex, yaitu bagian pada otak yang bertugas untuk memroses informasi dalam bentuk visual. Dengan arsitektur seperti itu, CNN dapat dilatih untuk memahami detail sebuah gambar dengan lebih baik. Dengan begitu, CNN dapat menangkap dependensi Spasial dan Temporal dalam sebuah gambar setelah kamu memberikan filter yang relevan.

CNN merupakan salah satu jenis neural network yang biasa digunakan pada data berupa image. CNN dapat digunakan untuk mendekripsi dan mengenali object pada sebuah image. Pada dasarnya CNN sama dengan neural network biasanya yang terdiri dari neuron yang memiliki weight, bias dan activation function seperti yang sudah kita pelajari pada part sebelumnya.

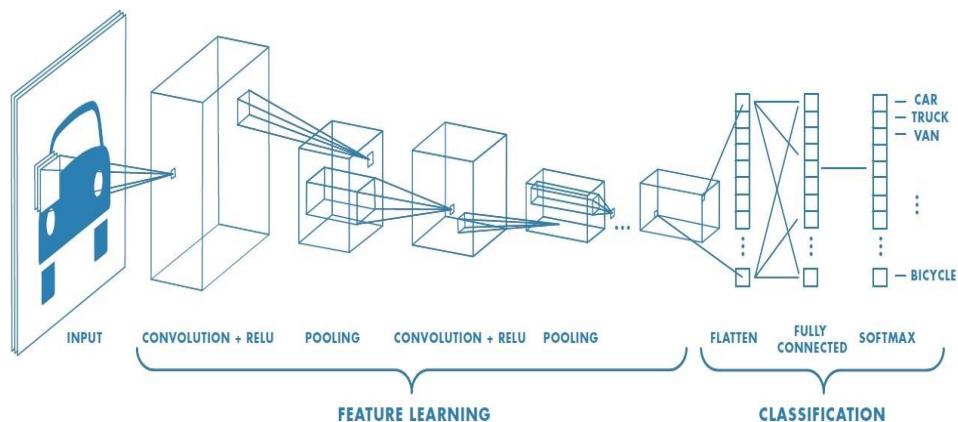
CNN memiliki kesamaan struktur dengan artificial neural network. Pada kasus klasifikasi citra, CNN menerima citra input atau masukan kemudian diproses dan diklasifikasi ke kategori tertentu (mis. pesawat,

kapal, burung, kucing, sapi). Perbedaan CNN dengan ANN adalah CNN memiliki arsitektur tambahan yang dioptimisasi untuk fitur yang ada pada citra input.

Komponen-komponen utama yang ada di dalam Convolutional Neural Networks adalah:

- Input layer
- Convolution Layer
- Activation Function
- Pooling Layer
- Fully Connected Layer

Pada gambar di bawah ini bisa dilihat alur dari proses CNN dalam mengolah citra masukan sampai mengklasifikasikan citra tersebut ke kategori tertentu berdasarkan nilai keluarannya.



Gambar 2. 1 Arsitektur CNN

CNN terdiri dari beberapa layer dan dirancang untuk pengenalan citra kompleks yang efektif. CNN mempunyai performansi yang baik dalam ekstraksi fitur untuk mencapai confidence rate tinggi layaknya cara kerja otak manusia. Dalam proses identifikasi dan klasifikasi terdapat banyak layer yang digunakan untuk mendapatkan hasil identifikasi yang akurat.

Kumpulan layer tersebut membentuk sebuah arsitektur yang kemudian digunakan untuk mengenali sebuah objek.

Arsitektur dari Convolutional Neural Network dibagi menjadi 2 bagian yaitu Feature Extraction Layer (Feature Learning) dan Fully Connected Layer (MLP) atau Classification. Dikatakan Feature Extraction Layer, karena proses yang terjadi pada bagian ini adalah melakukan encoding dari sebuah gambar menjadi features yang berupa angka-angka yang merepresentasikan gambar tersebut (Feature Extraction).

Berikut penjelasan pada layer yang terdapat pada arsitektur Convolution Neural Network :

### 2.1.1 Input Layer

Pada lapisan input layer nilai piksel dari citra ditampung dan akan menjadi masukan. Misalnya pada citra dengan ukuran  $64 \times 64$  dengan 3 channel warna, RGBm (Red, Green, Blue) maka yang menjadi masukan akan adalah piksel array yang berukuran  $64 \times 64 \times 3$



Gambar 2. 2 ILustrasi Input RGB

## 2.1.2 Convolution Layer



Gambar 2. 3 Gambaran Citra RGB

Pada gambar 2.3 merupakan RGB dengan ukuran 32x32 pixels yang sebenarnya adalah multidimensional array dengan ukuran 32x32x3, 3 adalah jumlah channel. Conv. layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (pixels). Contohnya, layer pertama pada feature extraction layer biasanya adalah conv. layer dengan ukuran 5x5x3. Panjangnya 5 pixels dan tingginya 5 pixels dan tebal atau jumlah 3 buah sesuai dengan channel dari image tersebut. Kemudian filter ini akan digeser keseluruh bagian dari gambar. Pada saat dilakukannya pergeseran, maka akan dilakukan dengan operasi matiriks “dot” antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah output atau biasa disebut sebagai activation map atau feature map.

Saat memprogram CNN, input adalah tensor dengan bentuk (jumlah gambar) x (lebar gambar) x (tinggi gambar) x ( kedalaman gambar ).

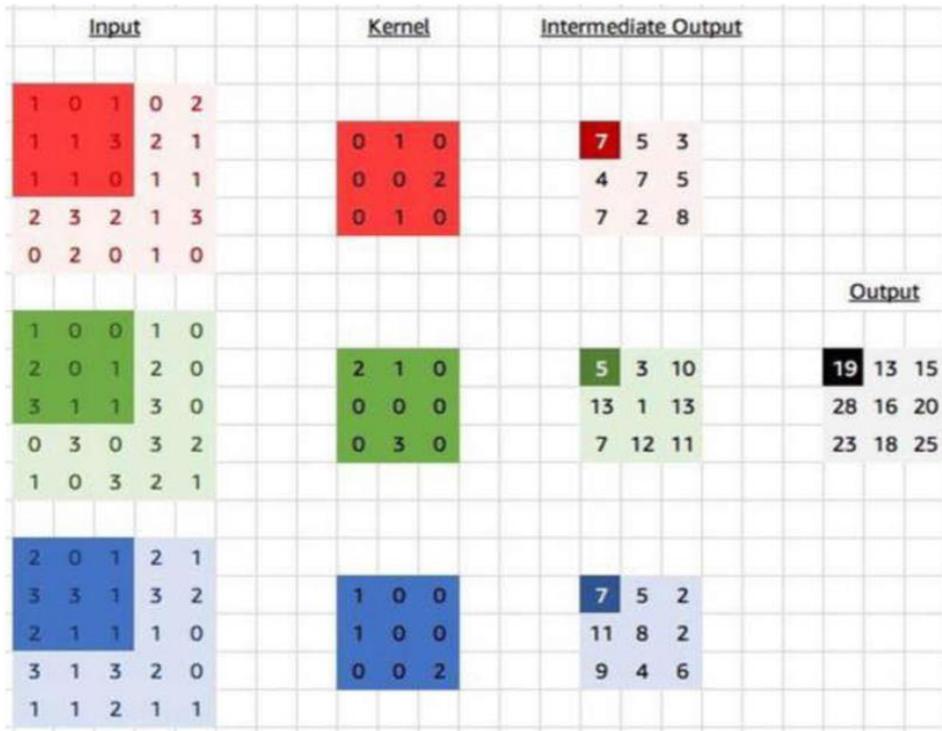
Kemudian setelah melewati lapisan konvolusional, gambar menjadi diabstraksi menjadi peta fitur, dengan bentuk (jumlah gambar) x (lebar peta fitur) x (tinggi peta fitur) x (saluran peta fitur). Lapisan convolutional dalam jaringan saraf harus memiliki atribut berikut:

- Kernel konvolusional ditentukan oleh lebar dan tinggi (parameter-hiper).
- Jumlah saluran input dan saluran output (hyper-parameter).
- Kedalaman filter Konvolusi (saluran input) harus sama dengan saluran angka (kedalaman) dari peta fitur input.

Lapisan konvolusional menggabungkan input dan meneruskan hasilnya ke lapisan berikutnya. Ini mirip dengan respons neuron di korteks visual terhadap rangsangan spesifik. Setiap neuron konvolusional hanya memproses data untuk bidang reseptifnya. Meskipun jaringan neural feedforward terhubung sepenuhnya dapat digunakan untuk mempelajari fitur serta mengklasifikasikan data, itu tidak praktis untuk menerapkan arsitektur ini untuk gambar. Jumlah neuron yang sangat tinggi akan diperlukan, bahkan dalam arsitektur dangkal (kebalikan dari kedalaman), karena ukuran input yang sangat besar yang terkait dengan gambar, di mana setiap piksel adalah variabel yang relevan.

Sebagai contoh, lapisan yang terhubung penuh untuk gambar (kecil) ukuran 100 x 100 memiliki 10.000 bobot untuk setiap neuron di lapisan kedua. Operasi konvolusi membawa solusi untuk masalah ini karena mengurangi jumlah parameter bebas, memungkinkan jaringan menjadi lebih dalam dengan lebih sedikit parameter. Misalnya, terlepas dari ukuran gambar, wilayah ubin dengan ukuran 5 x 5, masing-masing dengan bobot yang sama, hanya memerlukan 25 parameter yang bisa dipelajari. Dengan cara ini, ia menyelesaikan masalah gradien menghilang atau meledak

dalam pelatihan jaringan saraf multi-layer tradisional dengan banyak lapisan dengan menggunakan backpropagation.



Gambar 2. 4 Proses Convolusi Perkalian Dot

Filter akan digeser ke seluruh area dari gambar. Operasi “dot” akan dilakukan pada setiap pergeseran yang dilakukan antara input dan nilai dari filter tersebut sehingga menghasilkan sebuah output atau biasa disebut sebagai activation map atau feature map.

Pada proses konvolusi antara image input dengan kernel (filter) terdapat parameter yang menentukan hasil output antara lain:

### 1. Stride

Parameter yang menentukan berapa jumlah pergeseran filter. Jika stride nya 2 maka convolutional filter akan bergeser sebanyak 2 pixels secara horizontal lalu vertical. Semakin kecil stride maka akan semakin

detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan komputasi yang lebih jika dibandingkan dengan stride yang besar.

Stride mengontrol bagaimana kolom kedalaman di sekitar dimensi spasial (lebar dan tinggi) dialokasikan. Ketika langkahnya adalah 1 maka kita memindahkan filter satu piksel pada satu waktu. Hal ini menyebabkan bidang reseptif yang tumpang tindih antara kolom, dan juga volume output yang besar. Ketika langkahnya adalah 2 maka filter melompat 2 piksel sekaligus ketika mereka meluncur.

## 2. Padding atau Zero Padding

Padding atau Zero Padding adalah parameter yang menentukan jumlah pixels (berisi nilai 0) yang akan ditambahkan di setiap sisi dari input. Padding digunakan dengan tujuan untuk memanipulasi dimensi output dari conv. layer (Feature Map). Tujuan menggunakan padding adalah agar dimensi output tetap sama dengan dimensi input atau tidak berkurang secara drastis.

Dengan dimensi output convolutional layer yang selalu lebih kecil dari inputnya dan hasil output ini akan digunakan kembali sebagai input dari convolutional layer selanjutnya, sehingga makin banyak informasi terbuang. Dapat meningkatkan performa dari model karena convolutional filter akan fokus pada informasi yang sebenarnya yaitu yang berada diantara zero padding tersebut.

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Gambar 2. 5 Zero Padding Pada Matriks 4x4 Menjadi 6x6

Dimensi dari feature map dapat dihitung menggunakan rumus seperti dibawah ini:

$$output = \frac{W - N + 2P}{S} + 1$$

Gambar 2.6 Rumus Menghitung Dimensi Feature Map

Keterangan :

- W = Panjang/Tinggi Input
- N = Panjang/Tinggi Filter
- P = Zero Padding
- S = Stride

### 3. Dropout

Dropout merupakan proses mencegah terjadinya overfitting dan juga mempercepat proses learning. Dropout mengacu kepada menghilangkan neuron yang berupa hidden mapun layer yang visible di dalam jaringan. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak. Setiap neuron akan diberikan probabilitas yang bernilai antara 0 dan 1.

Karena lapisan yang terhubung penuh menempati sebagian besar parameter, ia cenderung overfitting.

Salah satu metode untuk mengurangi overfitting adalah putus sekolah. Pada setiap tahap pelatihan, setiap node "putus" dari jaring dengan probabilitas  $1-p$  atau disimpan dengan probabilitas  $p$ , sehingga jaringan tereduksi tertinggal; tepi masuk dan keluar ke simpul putus juga dihapus. Hanya jaringan tereduksi yang dilatih tentang data pada tahap itu. Node yang dihapus kemudian dimasukkan kembali ke jaringan dengan bobot aslinya. Pada tahap pelatihan, probabilitas bahwa simpul tersembunyi akan turun biasanya 0,5; untuk input node, ini harus jauh lebih rendah, secara intuitif karena informasi langsung hilang ketika input node diabaikan.

Pada waktu pengujian setelah pelatihan selesai, idealnya kami ingin mencari rata-rata sampel dari semua kemungkinan 2 jaringan putus; sayangnya ini tidak layak untuk nilai besar  $n$ . Namun, kami dapat menemukan perkiraan dengan menggunakan jaringan penuh dengan output setiap node tertimbang oleh faktor  $p$ , jadi nilai yang diharapkan dari output dari sembarang simpul sama dengan pada tahap pelatihan. Ini adalah kontribusi terbesar dari metode putus sekolah: meskipun ia menghasilkan secara efektif 2 jaring saraf, dan dengan demikian

memungkinkan untuk kombinasi model, pada waktu pengujian hanya satu jaringan yang perlu diuji.

Dengan menghindari pelatihan semua node pada semua data pelatihan, dropout mengurangi overfitting. Metode ini juga secara signifikan meningkatkan kecepatan pelatihan. Ini membuat kombinasi model praktis, bahkan untuk jaringan saraf yang dalam . Teknik ini tampaknya mengurangi interaksi simpul, mengarahkan mereka untuk mempelajari fitur yang lebih kuat yang lebih baik untuk menggeneralisasikan data baru.

### **2.1.3 Pooling Layer**

Konsep penting lain dari CNN adalah pooling, yang merupakan bentuk non-linear down-sampling Ada beberapa fungsi non-linear untuk menerapkan penggabungan di mana max penyatuhan adalah yang paling umum. Ini membagi gambar input menjadi satu set persegi panjang yang tidak tumpang tindih dan, untuk setiap sub-wilayah tersebut, menghasilkan maksimum.

Secara intuitif, lokasi pasti dari suatu fitur kurang penting daripada lokasi kasarnya relatif terhadap fitur-fitur lainnya. Ini adalah ide di balik penggunaan penyatuhan dalam jaringan saraf convolutional. Lapisan penyatuhan berfungsi untuk secara progresif mengurangi ukuran spasial dari representasi, untuk mengurangi jumlah parameter, jejak memori dan jumlah perhitungan dalam jaringan, dan karenanya juga mengendalikan overfitting Adalah umum untuk secara berkala memasukkan lapisan penyatuhan antara lapisan konvolusional berturut-turut dalam arsitektur CNN. Operasi penyatuhan menyediakan bentuk lain dari terjemahan invarian.

Lapisan pooling beroperasi secara independen pada setiap irisan kedalaman input dan mengubah ukurannya secara spasial. Bentuk yang paling umum adalah lapisan penyatuhan dengan filter ukuran  $2 \times 2$  diterapkan dengan langkah 2 sampel bawah pada setiap irisan kedalaman input dengan 2 sepanjang lebar dan tinggi, membuang 75% dari aktivasi. Dalam hal ini, setiap operasi maks lebih dari 4 angka. Dimensi kedalaman tetap tidak berubah.

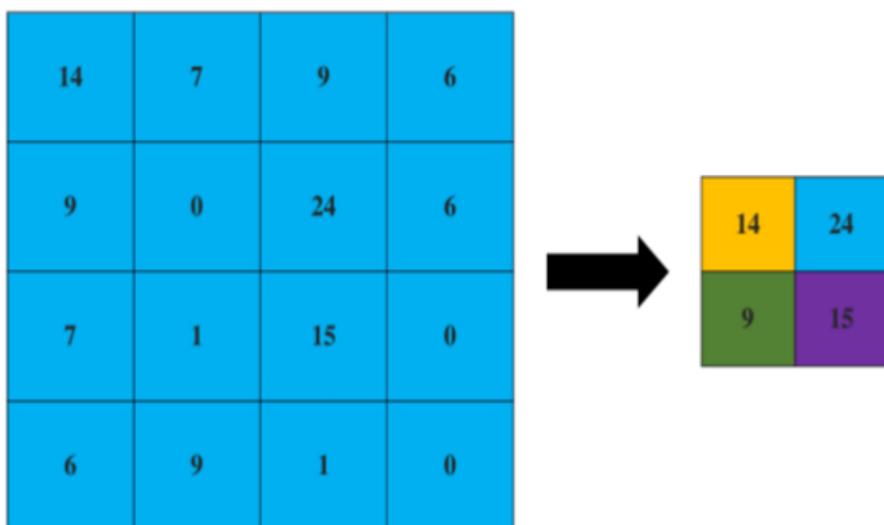
Selain max pooling, unit pooling dapat menggunakan fungsi lain seperti pooling rata - rata atau pool 2- pooling normal. Pooling rata-rata sering digunakan secara historis tetapi baru-baru ini tidak disukai dibandingkan dengan max pooling, yang berkinerja lebih baik dalam praktiknya. Karena pengurangan agresif dalam ukuran representasi, ada tren terbaru untuk menggunakan filter yang lebih kecil atau membuang lapisan gabungan sama sekali.

Jaringan konvolusional dapat mencakup lapisan pengumpulan lokal atau global untuk merampingkan perhitungan yang mendasarinya. Lapisan penggabungan mengurangi dimensi data dengan menggabungkan output gugus neuron pada satu lapisan menjadi satu neuron pada lapisan berikutnya. Penggabungan lokal menggabungkan kluster kecil, biasanya  $2 \times 2$ . Penggabungan global bertindak pada semua neuron dari lapisan konvolusional. Selain itu, penyatuhan mungkin menghitung maksimal atau rata-rata. Max pooling menggunakan nilai maksimum dari masing-masing sekelompok neuron pada lapisan sebelumnya. Penggabungan rata-rata menggunakan nilai rata-rata dari masing-masing sekelompok neuron pada lapisan sebelumnya.

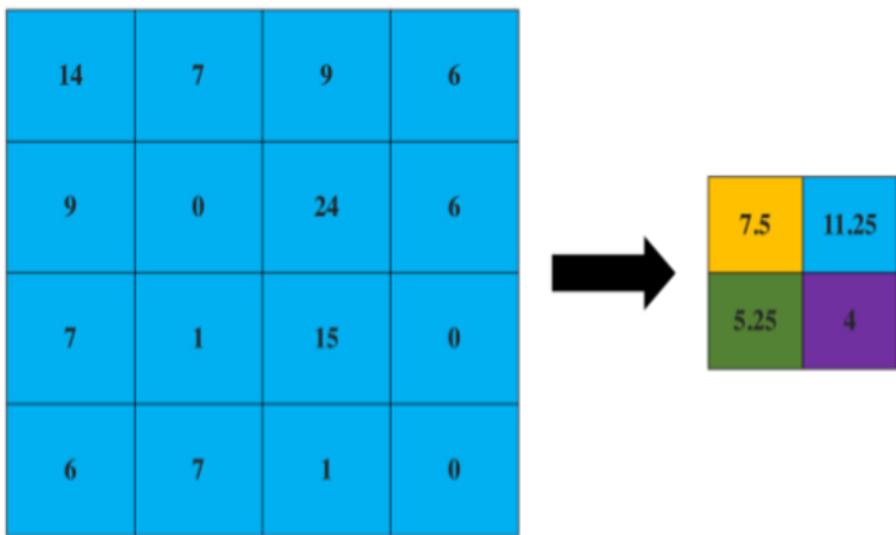
Pooling "Region of Interest" (juga dikenal sebagai pooling RoI) adalah varian dari max pooling, di mana ukuran output tetap dan persegi

panjang input adalah parameter. Pooling adalah komponen penting dari jaringan saraf convolutional untuk deteksi objek berdasarkan arsitektur Fast R-CNN.

Pooling layer biasanya berada setelah conv. Pooling layer pada prinsipnya terdiri dari sebuah filter dengan ukuran dan stride tertentu yang akan bergeser pada seluruh area feature map. Max Pooling dan Average Pooling adalah pooling yang biasa digunakan. Contohnya jika menggunakan Max Pooling  $2 \times 2$  dengan stride 2, maka pada setiap pergeseran filter, nilai maximum pada area  $2 \times 2$  pixel tersebut yang akan dipilih, sedangkan Average Pooling akan memilih nilai rata-ratanya. Layer ini berfungsi untuk mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus di update semakin sedikit dan mengatasi overfitting.



Gambar 2. 6 Maximum Pooling Layer



Gambar 2. 7 Average Pooling Layer

Proses konvolusi dan pooling dapat diulang beberapa kali hingga menghasilkan sebuah arsitektur Convolutional Neural Network yang diinginkan. Diantara convolutional layer dan pooling layer umumnya terdapat activation layer (ReLU, softmax, dll) yang banyak ditemukan pada artificial neural network. Activation layer adalah sebuah node/titik yang mendefinisikan sebuah output berdasarkan input yang diberikan. Pada CNN layer ini dapat didefinisikan sebagai batas (threshold) suatu input agar menghasilkan suatu output. Jika suatu input belum melebihi threshold maka tidak akan ada klasifikasi output.

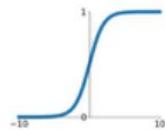
#### 2.1.4 Activation Layer

Lapisan aktivasi adalah layer dimana feature map dimasukkan ke dalam fungsi aktifasi. Layer ini berfungsi untuk mengubah nilai-nilai pada feature map pada range tertentu sesuai dengan fungsi aktifasi yang digunakan. ini bertujuan untuk meneruskan nilai yang menampilkan fitur

dominan dari citra yang masuk ke layer berikutnya. Fungsi aktifasi yang umum digunakan bisa dilihat pada gambar 2.8 sebagai berikut :

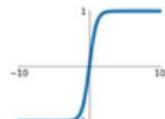
**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



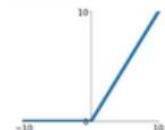
**tanh**

$$\tanh(x)$$



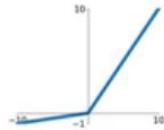
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

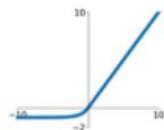


**Maxout**

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

**ELU**

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Gambar 2. 8 Activation Functions

Ketika kita mulai menggunakan jaringan syaraf, kita akan menggunakan fungsi aktivasi secara teratur karena fungsi aktivasi adalah bagian wajib dari jaringan saraf. Tujuan dari fungsi aktivasi adalah untuk menyesuaikan berat dan bias. Di TensorFlow, fungsi aktivasi adalah operasi non-linear yang bekerja pada tensor. Mereka adalah fungsi yang beroperasi dengan cara yang mirip dengan operasi matematika sebelumnya. Fungsi aktivasi melayani banyak tujuan, tetapi beberapa konsep utama adalah bahwa mereka memperkenalkan non-linearitas ke dalam grafik saat menormalkan output. Mulai grafik TensorFlow dengan perintah berikut:

```
import tensorflow as tf
```

```
sess = tf.Session()
```

Fungsi aktivasi hidup di jaringan neural network (nn) di TensorFlow. Selain menggunakan fungsi aktivasi bawaan, kami juga dapat mendesain sendiri menggunakan operasi TensorFlow. Kita dapat mengimpor fungsi

aktivasi yang telah ditentukan (import Tensorflow.nn as nn) atau secara eksplisit dan menulis nn dalam panggilan fungsi kita.

1. Rectified linear unit, yang dikenal sebagai ReLU, adalah cara paling umum dan dasar untuk memperkenalkan non-linearitas ke dalam jaringan saraf. Fungsi ini hanya  $\max(0, x)$ . Ini terus menerus tetapi tidak lancar. Bentuknya sebagai berikut:

```
print(sess.run(tf.nn.relu([-3., 3., 10.]))[ 0. 3. 10.])
```

2. Akan ada saat-saat ketika kita ingin membatasi bagian yang meningkat secara linear dari fungsi aktivasi ReLU sebelumnya. Kita bisa melakukan ini dengan menumpuk fungsi  $\max(0, x)$  ke fungsi  $\min(0)$ . Implementasi yang TensorFlow miliki disebut fungsi ReLU6. Ini didefinisikan sebagai  $\min(\max(0, x), 6)$ . Ini adalah versi dari fungsi hard-sigmoid dan secara komputasi lebih cepat, dan tidak mengalami kehilangan ((infinitesimally near zero)) atau exploding value.

```
print(sess.run(tf.nn.relu6([-3., 3., 10.]))[ 0. 3. 6.])
```

3. Fungsi sigmoid adalah fungsi aktivasi berkelanjutan dan smooth yang paling umum. Ini juga disebut fungsi logistik dan memiliki bentuk  $1 / (1 + \exp(-x))$ . Sigmoid tidak sering digunakan karena kecenderungan untuk menghilangkan istilah backpropagation selama training. Bentuknya seperti dibawah ini :

```
print(sess.run(tf.nn.sigmoid([-1., 0., 1.])))[ 0.26894143 0.5  
0.7310586]
```

Kita harus menyadari bahwa beberapa fungsi aktivasi tidak mempunyai nol pusat, seperti sigmoid. Ini akan mengharuskan kita untuk nol, berarti data sebelum menggunakannya di sebagian besar algoritma grafik komputasi.

4. Fungsi aktivasi smooth lainnya adalah hyper tangent. Fungsi hyper tangent sangat mirip dengan sigmoid yang memiliki rentang antara 0 dan 1, hyper tangent memiliki rentang antara -1 dan 1. Fungsi ini memiliki bentuk  $((\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x)))$ . Ini bisa dilihat sebagai berikut:

```
print(sess.run(tf.nn.tanh([-1.,0.,1.]))) [-0.761594180076159418]
```

Fungsi softsign juga digunakan sebagai fungsi aktivasi. Bentuk fungsi ini adalah  $x / (\text{abs}(x) + 1)$ . Fungsi softsign seharusnya menjadi pendekatan berkelanjutan untuk fungsi sign. Dapat dilihat sebagai berikut:

```
print (sess.run(tf.nn.softsign([-1., 0., -1.])))[-0.5 0. 0.5]
```

5. Fungsi lain, softplus, adalah versi smooth dari fungsi ReLU. Bentuk fungsi ini adalah  $\log(\exp(x) + 1)$ . Dapat dilihat sebagai berikut:

```
print(sess.run(tf.nn.softplus([-1., 0., -1.])))[ 0.31326166 0.69314718  
1.31326163]
```

Softplus pergi ke arah tak terbatas(infinity) sebagai input increases sedangkan softsign pergi ke 1. sebagai input smaller, bagaimanapun softplus mendekati nol dan softsign pergi ke -1.

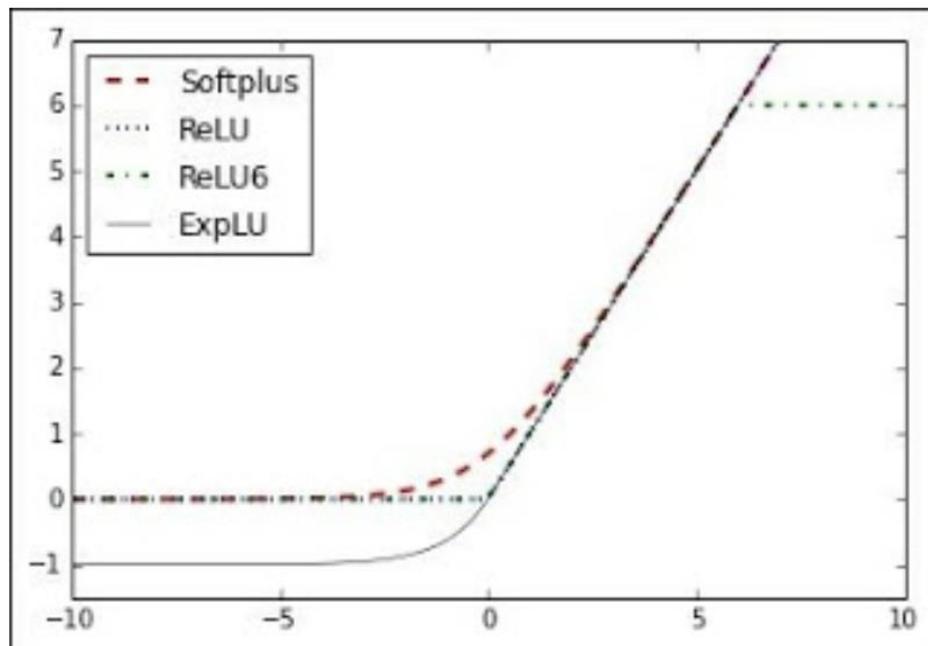
6. The Exponential Linear Unit (ELU) sangat mirip dengan fungsi softplus kecuali bahwa asymptote bawah adalah -1 bukannya 0. Bentuknya  $(\exp(x) + 1)$  jika  $x < 0$  else  $x$ . Dapat dilihat sebagai berikut:
- ```
print(sess.run(tf.nn.elu([-1., 0., -1.])))[-0.63212055 0. 1.]
```

Fungsi aktivasi ini adalah cara memperkenalkan nonlinier pada jaringan syaraf atau grafik komputasional lainnya di masa depan. Penting untuk dicatat di mana di jaringan, kita menggunakan fungsi aktivasi. Jika fungsi aktivasi memiliki rentang antara 0 dan 1

(sigmoid), maka grafik komputasi hanya dapat menghasilkan nilai antara 0 dan 1.

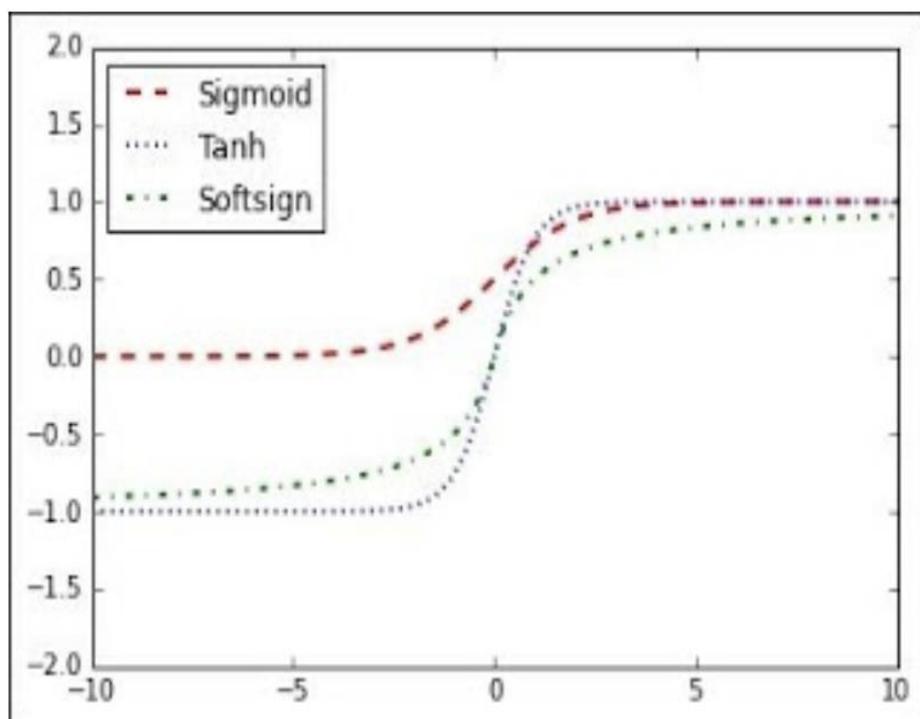
Jika fungsi aktivasi berada di dalam dan tersembunyi di antara node, maka kita ingin mengetahui efek yang dapat terjadi pada tensor saat kita melewati mereka. Jika tensor kita diskalakan memiliki mean nol, kita ingin menggunakan fungsi aktivasi yang mempertahankan sebanyak mungkin variasi di sekitar nol. Ini berarti kita ingin memilih fungsi aktivasi seperti tangen hiperbolik (tanh) atau softsign. Jika tensor semuanya diskalakan menjadi positif, maka kita idealnya memilih fungsi aktivasi yang mempertahankan varians di domain positif.

Berikut ini dua grafik yang menggambarkan fungsi aktivasi yang berbeda. Gambar berikut menunjukkan fungsi-fungsi berikut ReLU, ReLU6, softplus, exponential LU, sigmoid, softsign, dan tangen hiperbolik yang dapat dilihat pada gambar 2.9 dan gambar 2.10 :



Gambar 2. 9 Grafik Activation

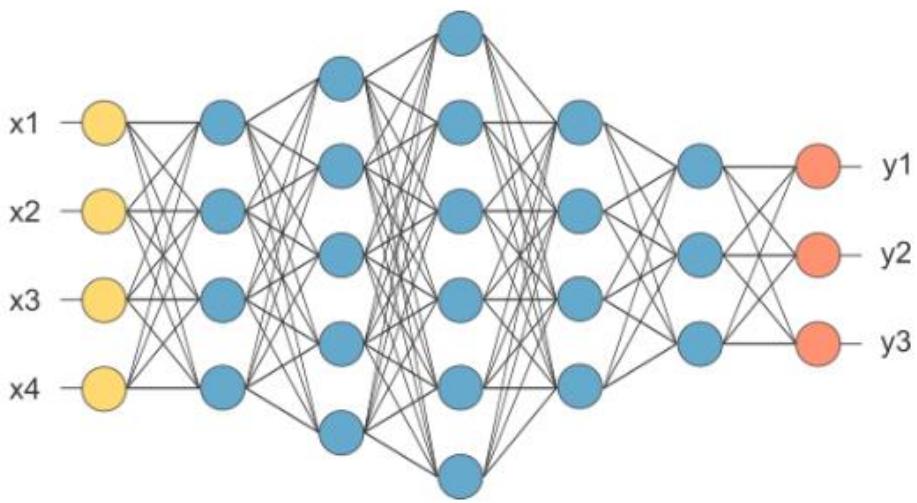
Pada Gambar diatas, kita dapat melihat empat fungsi aktivasi, softplus, ReLU, ReLU6, dan LU eksponensial. Fungsi-fungsi ini meratakan ke kiri dan meningkat secara linear ke kanan, dengan pengecualian ReLU6, yang memiliki nilai maksimum 6.



Gambar 2. 10 Grafik Activation

### 2.1.5 Fully Connected Layer

Jika proses-proses diatas sudah dilewati maka hasil dari pooling layer digunakan menjadi masukan untuk Fully connected layer. Pada lapisan ini terdapat kesamaan struktur dengan Artificial Neural Network pada umumnya yaitu memiliki input layer, hidden layer dan output layer yang masing-masing memiliki neuron-neuron yang saling terhubung dengan neuron-neuron di layer tetangganya. Contoh Fully Connected Layer terdapat pada gambar pada gambar 2.11 :



Gambar 2. 11 Ilustrasi Fully Connected Layer

Dari gambar diatas, dimana layer 1 akan dilakukan feedforwarding menuju layer 2 dengan menggunakan fungsi aktivasi ReLU. Pada layer 2 akan dilakukan klasifikasi dengan menggunakan softmax. Setiap neuron menerima weight yang diperioritaskan pada kelas yang paling sesuai. Pada akhirnya neuron akan melakukan pemilihan pada kelas terbaik. kelas dengan nilai terbaik diberikan klasifikasi sesuai dengan kelasnya.

Fully connected layer bertujuan untuk mengambil hasil dari proses konvolusi, pooling dan menggunakan hasil parameter tersebut dalam bentuk hidden neuron untuk melakukan klasifikasi label dari foto atau video. Output dari konvolusi dan pooling berbentuk linier dan dibuat dalam nilai single vector yang mana tiap nilai single vector merepresentasikan feature yang berasal dari kelas yang ada. Misalkan kelas plate dalam penelitian ini, apabila ada gambar dengan ciri-ciri warna dan bentuk mirip dengan plate, maka secara probabilitas akan lebih besar untuk dikategorikan sebagai kelas plate. Pada layer terakhir di dalam Fully Connected layer akan digunakan fungsi aktifasi sigmoid atau softmax untuk menentukan

klasifikasi atau output dari citra inputan atau masukan dari Input Layer CNN.

### **2.1.6 Sejarah Convolution Neural Network (CNN)**

#### **1. Bidang reseptif dalam korteks visual**

Karya Hubel dan Wiesel pada 1950-an dan 1960-an menunjukkan bahwa korteks visual kucing dan monyet mengandung neuron yang secara individual merespons wilayah kecil bidang visual . Asalkan mata tidak bergerak, wilayah ruang visual di mana rangsangan visual mempengaruhi penembakan neuron tunggal dikenal sebagai bidang reseptifnya. Sel tetangga memiliki bidang reseptif yang sama dan tumpang tindih. Ukuran dan lokasi bidang reseptif bervariasi secara sistematis di seluruh korteks untuk membentuk peta lengkap ruang visual. Korteks di setiap belahan mewakili bidang visual kontralateral. Makalah mereka tahun 1968 mengidentifikasi dua jenis sel visual dasar di otak:

- Sel-sel sederhana, yang outputnya dimaksimalkan oleh tepi lurus yang memiliki orientasi khusus dalam bidang reseptifnya.
- Sel kompleks, yang memiliki bidang reseptif yang lebih besar, yang outputnya tidak sensitif terhadap posisi tepi yang tepat di bidang.

Hubel dan Wiesel juga mengusulkan model kaskade dari kedua jenis sel untuk digunakan dalam tugas-tugas pengenalan pola.

#### **2. Neocognitron, asal dari arsitektur CNN**

"Neocognitron" diperkenalkan oleh Kunihiko Fukushima pada tahun 1980. Itu terinspirasi oleh karya Hubel dan Wiesel yang disebutkan di atas. Neocognitron memperkenalkan dua tipe dasar lapisan dalam CNN: lapisan konvolusional, dan lapisan downsampling. Lapisan konvolusional berisi unit yang bidang reseptifnya menutupi tambalan lapisan sebelumnya. Vektor bobot (himpunan parameter adaptif) dari unit tersebut sering

disebut filter. Unit dapat berbagi filter. Lapisan downsampling berisi unit yang bidang reseptifnya mencakup tambalan dari lapisan konvolusional sebelumnya. Unit seperti itu biasanya menghitung rata-rata aktivasi unit dalam tambalannya. Downsampling ini membantu untuk mengklasifikasikan objek dalam adegan visual dengan benar bahkan ketika objek digeser.

Dalam varian neokognitron yang disebut cresceptron, alih-alih menggunakan rata-rata spasial Fukushima, J. Weng et al. memperkenalkan metode yang disebut max-pooling di mana unit downsampling menghitung maksimum aktivasi unit dalam tambalannya. Max-pooling sering digunakan dalam CNN modern. Beberapa algoritma pembelajaran yang diawasi dan tidak diawasi telah diusulkan selama beberapa dekade untuk melatih bobot neokognitron. Namun, hari ini, arsitektur CNN biasanya dilatih melalui backpropagation. Neocognitron adalah CNN pertama yang membutuhkan unit yang terletak di berbagai posisi jaringan untuk memiliki bobot bersama. Neokognitron diadaptasi pada tahun 1988 untuk menganalisis sinyal yang bervariasi waktu.

### 3. Jaringan saraf tunda waktu

Waktu tunda jaringan saraf (TDNN) diperkenalkan pada tahun 1987 oleh Alex Waibel et al. dan merupakan jaringan konvolusional pertama, karena mencapai shift invarian. Itu dilakukan dengan memanfaatkan pembagian berat badan dalam kombinasi dengan pelatihan Backpropagation Dengan demikian, meski juga menggunakan struktur piramidal seperti pada neokognitron, ia melakukan optimalisasi bobot secara global, alih-alih yang lokal.

TDNN adalah jaringan konvolusional yang berbagi bobot di sepanjang dimensi temporal. Mereka memungkinkan sinyal ucapan

diproses secara invarian. Pada 1990, Hampshire dan Waibel memperkenalkan varian yang melakukan konvolusi dua dimensi. Karena TDNN ini beroperasi pada spektrogram, sistem pengenalan fonem yang dihasilkan adalah tidak sama untuk keduanya, bergeser dalam waktu dan frekuensi. Penerjemahan invarian ini mengilhami dalam pemrosesan gambar dengan CNN. Ubin output neuron dapat mencakup tahap waktunya. TDNN sekarang mencapai kinerja terbaik dalam pengenalan suara jarak jauh.

#### 4. Max pooling

Pada tahun 1990 Yamaguchi et al. memperkenalkan konsep max pooling. Mereka melakukannya dengan menggabungkan TDNNs dengan max pooling untuk mewujudkan sistem pengenal kata terisolasi yang independen. Dalam sistem mereka, mereka menggunakan beberapa TDNN per kata, satu untuk setiap suku kata . Hasil dari setiap TDNN pada sinyal input digabungkan menggunakan max pooling dan output dari layer pooling kemudian diteruskan ke jaringan yang melakukan klasifikasi kata yang sebenarnya.

#### 5. Pengenalan gambar dengan CNN dilatih oleh gradient descent

Sebuah sistem untuk mengenali nomor Kode ZIP yang ditulis tangan melibatkan konvolusi di mana koefisien kernel telah dirancang dengan susah payah. Yann LeCun et al. (1989) menggunakan back-propagation untuk mempelajari koefisien kernel konvolusi secara langsung dari gambar angka tulisan tangan. Dengan demikian pembelajaran sepenuhnya otomatis, dilakukan lebih baik daripada desain koefisien manual, dan cocok untuk berbagai masalah pengenalan gambar dan tipe gambar yang lebih luas. Pendekatan ini menjadi dasar dari visi komputer modern.

## 6. LeNet-5

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

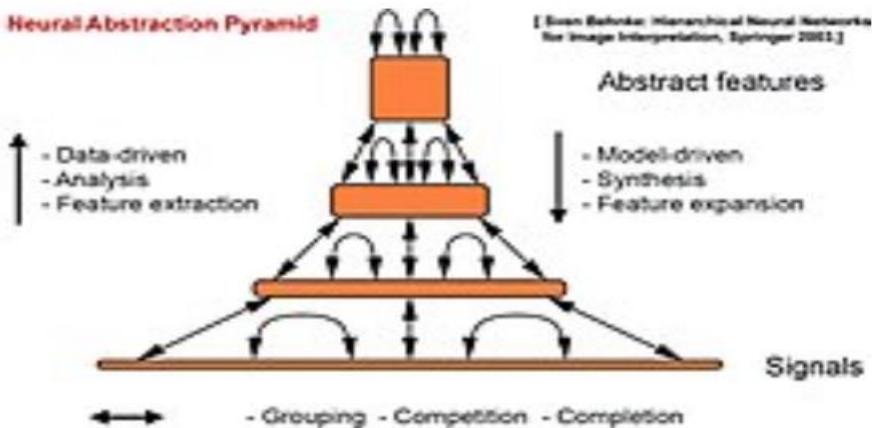
## 7. Jaringan Saraf Shift-invarian

Demikian pula, pergeseran jaringan saraf invarian diusulkan oleh W. Zhang et al. untuk pengenalan karakter gambar pada tahun 1988. Arsitektur dan algoritma pelatihan dimodifikasi pada tahun 1991 dan diterapkan untuk pemrosesan gambar medis dan deteksi otomatis kanker payudara pada mammogram.

Desain berbasis konvolusi yang berbeda diusulkan pada tahun 1988 untuk aplikasi dekomposisi sinyal konvolusi elektromiografi satu dimensi melalui de-konvolusi. Desain ini dimodifikasi pada tahun 1989 untuk desain berbasis de-konvolusi lainnya.

## 8. Piramida abstraksi saraf

Arsitektur umpan-maju dari jaringan saraf convolutional diperluas dalam piramida abstraksi saraf oleh koneksi lateral dan umpan balik. Jaringan konvolusional berulang yang dihasilkan memungkinkan penggabungan informasi kontekstual yang fleksibel untuk secara iteratif menyelesaikan ambiguitas lokal. Berbeda dengan model sebelumnya, output seperti gambar pada resolusi tertinggi dihasilkan, misalnya, untuk segmentasi semantik, rekonstruksi gambar, dan tugas lokalisasi objek.



Gambar 2. 12 Piramida Abstraksi Saraf

## 9. Implementasi GPU

Meskipun CNN diciptakan pada 1980-an, terobosan mereka pada 2000-an membutuhkan implementasi cepat pada unit pemrosesan grafis atau GPU. Pada tahun 2004, ditunjukkan oleh KS Oh dan K. Jung bahwa jaringan saraf standar dapat sangat dipercepat pada GPU. Implementasinya 20 kali lebih cepat dari implementasi yang setara pada CPU. Pada 2005, makalah lain juga menekankan nilai GPGPU untuk pembelajaran mesin.

Implementasi GPU pertama dari CNN dijelaskan pada tahun 2006 oleh K. Chellapilla et al. Implementasinya 4 kali lebih cepat dari implementasi yang setara pada CPU. Pekerjaan selanjutnya juga menggunakan GPU, awalnya untuk jenis jaringan saraf lain (berbeda dari CNN), terutama jaringan saraf yang tidak diawasi.

Pada 2010, Dan Ciresan et al. di IDSIA menunjukkan bahwa bahkan jaringan saraf standar yang dalam dengan banyak lapisan dapat dengan cepat dilatih pada GPU dengan mengawasi pembelajaran melalui metode lama yang dikenal sebagai backpropagation. Jaringan mereka mengungguli metode pembelajaran mesin sebelumnya pada patokan digit tulisan tangan MNIST. Pada 2011, mereka memperluas pendekatan GPU

ini ke CNN, mencapai faktor akselerasi 60, dengan hasil yang mengesankan.

Pada tahun 2011, mereka menggunakan CNN pada GPU untuk memenangkan kontes pengenalan gambar di mana mereka mencapai kinerja manusia super untuk pertama kalinya. Antara 15 Mei 2011 dan 30 September 2012, CNN mereka memenangkan tidak kurang dari empat kompetisi gambar. Pada tahun 2012, mereka juga secara signifikan meningkatkan kinerja terbaik dalam literatur untuk beberapa basis data gambar, termasuk basis data MNIST, basis data NORB, dataset HWDB1.0 (karakter Cina) dan dataset CIFAR10 (dataset dari 60000 32x32 gambar berlabel RGB).

Selanjutnya, CNN berbasis GPU yang serupa oleh Alex Krizhevsky et al. memenangkan ImageNet Large Scale Visual Recognition Challenge 2012. CNN yang sangat dalam dengan lebih dari 100 lapisan oleh Microsoft memenangkan kontes ImageNet 2015.

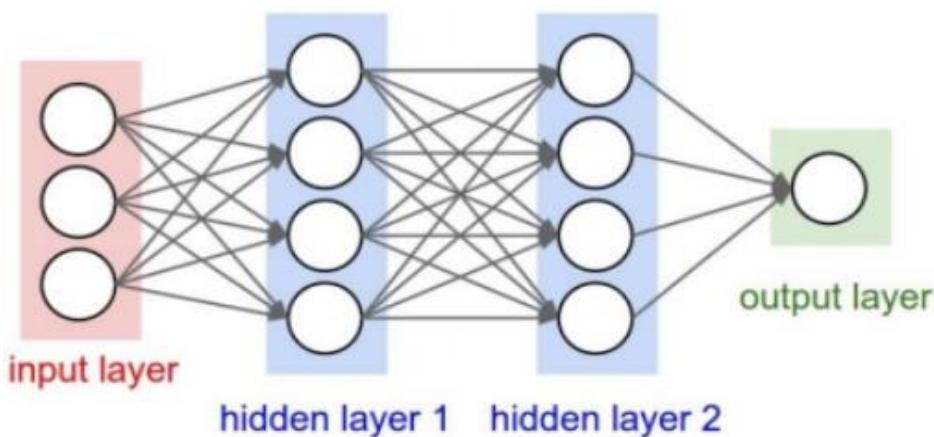
## 10. Implementasi Intel Xeon Phi

Dibandingkan dengan pelatihan CNN menggunakan GPU, tidak banyak perhatian diberikan kepada prosesor Intel Xeon Phi. Perkembangan penting adalah metode paralelisasi untuk melatih jaringan saraf convolutional pada Intel Xeon Phi, bernama Controlled Hogwild dengan Arbitrary Order of Synchronization (CHAOS). CHAOS mengeksplorasi paralelisme tingkat benang dan SIMD yang tersedia pada Intel Xeon Phi.

### **2.1.7 Perkembangan Convolution Neural Network**

Convolutional Neural Network merupakan salah satu metode machine learning dari pengembangan Multi Layer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. Metode ini termasuk ke dalam jenis

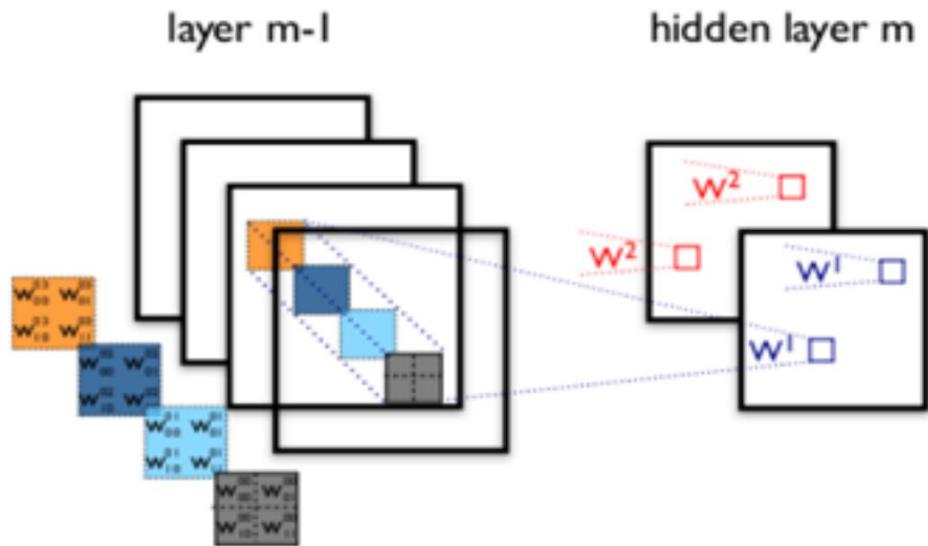
Deep Neural Network karena dalamnya tingkat jaringan dan banyak diimplementasikan dalam data citra. CNN mempunyai dua metode yaitu klasifikasi menggunakan feedforward dan tahap pembelajaran menggunakan backpropagation. CNN memiliki cara kerja yang hampir sama dengan MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.



Gambar 2. 13 Layer MLP

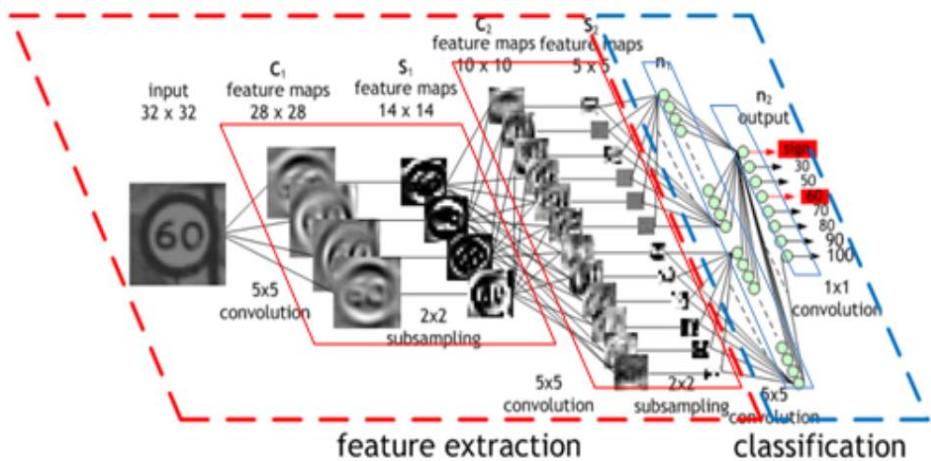
Layer MLP memiliki  $i$  layer (kotak merah dan biru) dengan masing-masing layer berisi  $j_i$  neuron (lingkaran putih). MLP menerima input data satu dimensi dan mempropagasi data tersebut pada jaringan hingga menghasilkan output. Pada neuron dua layer yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Pada setiap data yang diinputkan pada layer dilakukan operasi linear dengan nilai bobot yang ada, kemudian hasil komputasi akan ditransformasi menggunakan operasi non linear yang disebut sebagai fungsi aktivasi. Data melakukan dipropagasi pada CNN adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. CNN melakukan

operasi linear menggunakan operasi konvolusi, dengan bobot yang tidak lagi satu dimensi saja. namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar berikut Dimensi bobot pada CNN adalah:



Gambar 2. 14 Neuron Input x Neuron Output x Tinggi x Lebar

Metode CNN berkembang lebih lanjut dari metode MLP karena menggunakan metode yang mirip dengan dimensi yang lebih banyak. Pada metode CNN, input dari layer sebelumnya bukan array 1 dimensi melainkan array 2 dimensi. Jika di analogikan dengan fitur dari wajah manusia, layer pertama merupakan refleksi goresan-goresan berbeda arah, pada layer kedua fitur seperti bentuk mata, hidung, dan mulut mulai terlihat, hal ini karena di lakukan pooling/penggabungan dari layer pertama yang masih berupa goresan-goresan, pada layer ketiga akan terbentuk kombinasi fitur-fitur mata, hidung, dan mulut yang nantinya akan disimpulkan dengan wajah orang tertentu.

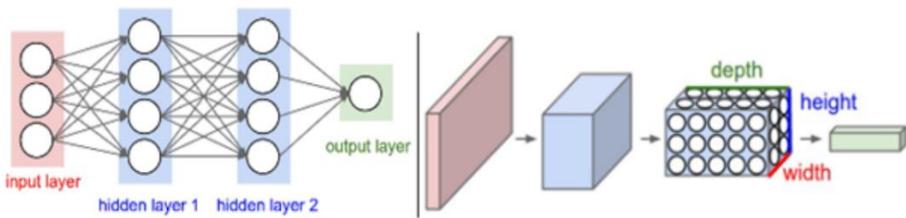


Gambar 2. 15 Contoh dan Arsitektur Dari CNN

Terdapat beberapa lapisan tersembunyi (hidden layers) dari sebuah input berupa vector tunggal Pada CNN. Gambar 2.16 terdapat input berupa citra yang dijadikan vektor tunggal  $32 \times 32$ . Setiap hidden layer terdapat beberapa neuron layaknya empat feature maps  $C_1$  pada gambar tersebut. Neuron yang tedapat pada  $C_1$  dihubungkan dengan neuron di  $S_1$ , dan seterusnya. Layer yang terakhir terhubung dengan lapisan-lapisan tersembunyi sebelumnya disebut dengan output layer dan merepresentasikan hasil akhir klasifikasi kelas.  $n_2$  yang ditunjukkan oleh Gambar 2.15 yang merepresentasikan hasil pada output layer, seperti 30, 50, 60, dll.

CNN memiliki kelebihan yaitu menggunakan dimensi  $> 1$  akan memengaruhi keseluruhan skala dalam suatu objek. Semua skala yang terdapat pada objek sangat penting agar input tidak kehilangan informasi spasialnya yang akan diekstraksi fitur dan diklasifikasikan. Proses ini menambah tingkat akurasi dan optimum algoritma CNN. Seperti pada kubus yang memiliki skala pada panjang, lebar, dan tinggi. Bila menggunakan Neural Network biasa, mungkin hanya memuat skala

panjang dan tinggi. Namun CNN bisa memuat semua informasi dari keseluruhan skala yang bisa mengklasifikasikan objek dengan lebih akurat karena bisa menggunakan skala lebarnya juga (yang mungkin tidak akan terlihat oleh Neural Network lainnya yang berdimensi dua).



Gambar 2. 16 Perbedaan Arsitektur NN Dengan CNN

## 2.2 Pengantar Region - Convolution Neural Network (R-CNN)

Masalah yang dicoba diselesaikan oleh sistem R-CNN adalah menemukan objek dalam gambar (deteksi objek). Apa yang Anda lakukan untuk menyelesaikan ini? Anda bisa mulai dengan pendekatan sliding window. Saat menggunakan metode ini, Anda cukup melihat keseluruhan gambar dengan ukuran persegi yang berbeda dan melihat gambar yang lebih kecil dalam metode brute force. Masalahnya adalah Anda akan memiliki sejumlah besar gambar yang lebih kecil untuk dilihat. Untuk keberuntungan kami, orang-orang pintar lainnya mengembangkan algoritma untuk memilih dengan cerdas apa yang disebut proposal kawasan. Untuk menyederhanakan konsep ini: "Proposal kawasan hanyalah sebagian kecil dari gambar asli, yang kami pikir dapat berisi objek yang kami cari."

### 2.2.1 Region Proposal

Ada berbagai algoritma proposal wilayah yang dapat kita pilih. Ini adalah algoritma "normal" yang bekerja di luar kotak. Kami tidak harus melatih mereka atau apa pun. Dalam kasus makalah ini, mereka

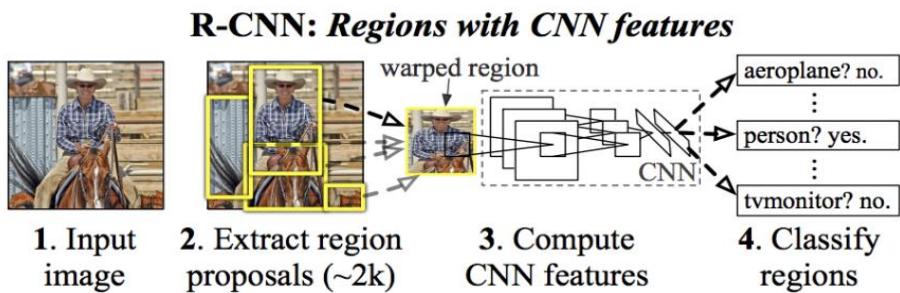
menggunakan metode pencarian selektif untuk menghasilkan proposal wilayah. Saya menemukan penjelasan yang sangat bagus dan terperinci tentang bagaimana, algoritma bekerja di sini . Namun perlu diingat:

R-CNN adalah agnostik terhadap metode proposal wilayah. Anda dapat memilih metode apa pun yang Anda suka dan itu akan berhasil baik. Ini akan membuat hampir 2.000 wilayah berbeda yang harus kita perhatikan. Ini terdengar seperti jumlah yang besar, tetapi masih sangat kecil dibandingkan dengan pendekatan jendela geser kasar.

Untuk menghindari masalah memilih sejumlah besar daerah, Ross Girshick et al mengusulkan metode di mana kami menggunakan pencarian selektif untuk mengekstrak hanya 2000 wilayah dari gambar dan dia menyebutnya proposal wilayah. Karena itu, sekarang, alih-alih mencoba mengklasifikasikan sejumlah besar wilayah, Anda dapat bekerja dengan 2000 wilayah. Proposal 2000 wilayah ini dihasilkan menggunakan algoritma pencarian selektif yang ditulis di bawah ini.

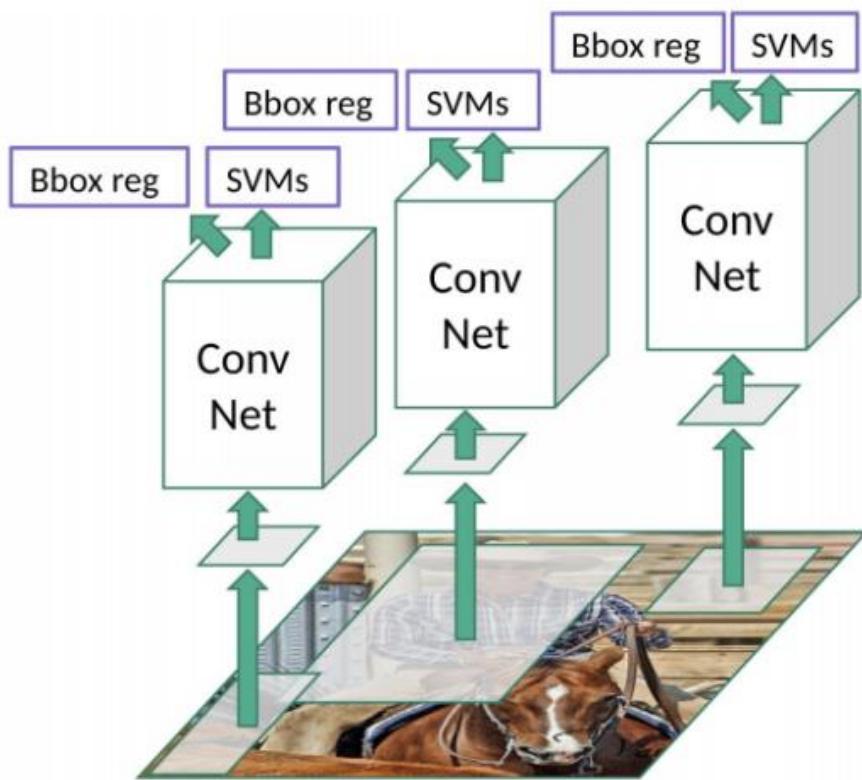
#### Pencarian Selektif :

- Menghasilkan sub-segmentasi awal, kami menghasilkan banyak kandidat daerah
- item Menggunakan algoritma serakah untuk secara rekursi menggabungkan daerah yang sama menjadi yang lebih besar
- Menggunakan daerah yang dihasilkan untuk menghasilkan proposal daerah kandidat akhir



Gambar 2. 17 Arsitektur R-CNN

Proposal wilayah kandidat 2000 ini dibengkokkan menjadi kotak dan dimasukkan ke dalam jaringan saraf convolutional yang menghasilkan vektor fitur 4096-dimensi sebagai output. CNN bertindak sebagai ekstraktor fitur dan lapisan padat output terdiri dari fitur yang diekstraksi dari gambar dan fitur yang diekstraksi dimasukkan ke dalam SVM untuk mengklasifikasikan keberadaan objek dalam proposal wilayah kandidat. Selain memprediksi keberadaan objek dalam proposal wilayah, algoritme juga memprediksi empat nilai yang merupakan nilai offset untuk meningkatkan ketepatan kotak pembatas. Misalnya, mengingat proposal wilayah, algoritme akan memperkirakan keberadaan seseorang tetapi wajah orang tersebut di dalam proposal wilayah tersebut dapat dipotong setengahnya. Oleh karena itu, nilai offset membantu menyesuaikan kotak pembatas proposal wilayah.



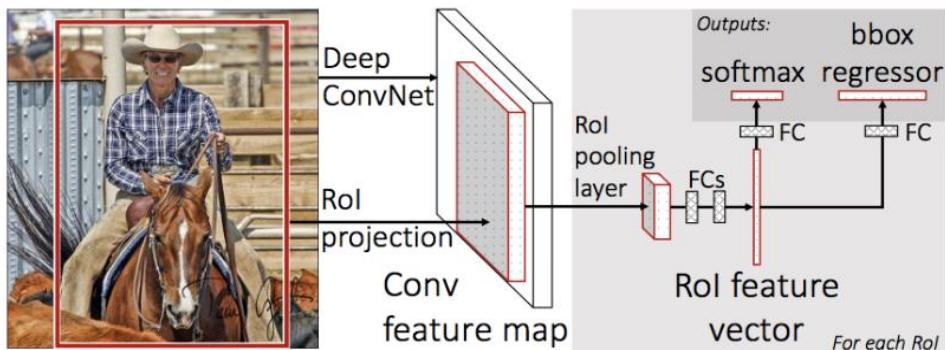
Gambar 2. 18 Proses R-CNN

### 2.2.2 Masalah Dengan R-CNN

Berikut beberapa masalah pada metode R-CNN :

- Masih membutuhkan banyak waktu untuk melatih jaringan karena Anda harus mengklasifikasikan 2.000 proposal wilayah per gambar.
- Itu tidak dapat diimplementasikan real time karena dibutuhkan sekitar 47 detik untuk setiap gambar uji.
- Algoritme pencarian selektif adalah algoritma tetap. Karena itu, tidak ada pembelajaran yang terjadi pada tahap itu. Hal ini dapat mengarah pada pembuatan proposal daerah kandidat yang buruk.

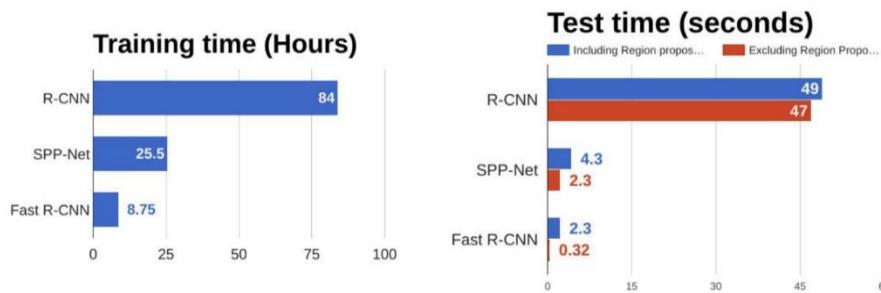
## 2.3 Pengantar Fast Region - Convolution Neural Network (Fater R-CNN)



Gambar 2. 19 Fast R-NN

R-CNN memecahkan beberapa kelemahan R-CNN untuk membangun algoritma deteksi objek yang lebih cepat dan disebut Fast R-CNN. Pendekatannya mirip dengan algoritma R-CNN. Tapi, alih-alih mengumpulkan proposal wilayah ke CNN, kami memberi makan gambar input ke CNN untuk menghasilkan peta fitur convolutional. Dari peta fitur konvolusional, kami mengidentifikasi wilayah proposal dan membengkokkannya ke dalam kotak dan dengan menggunakan lapisan pool RoI kami membentuk kembali menjadi ukuran tetap sehingga dapat dimasukkan ke dalam lapisan yang terhubung sepenuhnya.

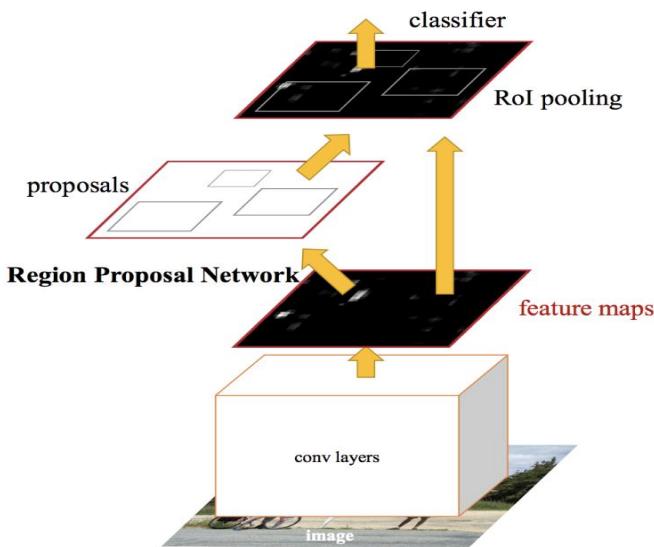
Dari vektor fitur RoI, kami menggunakan lapisan softmax untuk memprediksi kelas wilayah yang diusulkan dan juga nilai offset untuk kotak pembatas. Alasan "Fast R-CNN" lebih cepat daripada R-CNN adalah karena Anda tidak harus memasukkan 2.000 proposal wilayah ke jaringan saraf convolutional setiap waktu. Sebaliknya, operasi konvolusi dilakukan hanya sekali per gambar dan peta fitur dihasilkan darinya.



Gambar 2. 20 Perbandingan Algoritma Objek Deteksi

Dari gambar grafik 2.20 dapat disimpulkan bahwa Fast R-CNN secara signifikan lebih cepat dalam sesi pelatihan dan pengujian dibandingkan R-CNN. Ketika Anda melihat kinerja Fast R-CNN selama waktu pengujian, termasuk proposal wilayah memperlambat algoritma secara signifikan bila dibandingkan dengan tidak menggunakan proposal wilayah. Oleh karena itu, proposal kawasan menjadi hambatan dalam algoritma Fast R-CNN yang memengaruhi kinerjanya.

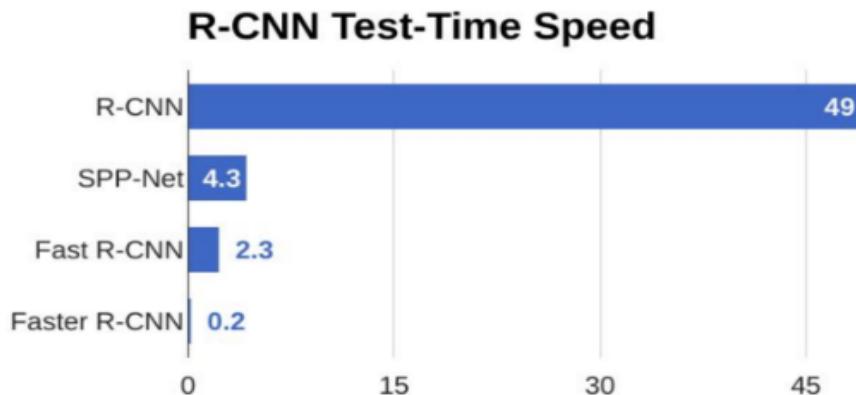
#### 2.4 Faster Region - Convolution Neural Network (Faster R-CNN)



Gambar 2. 21 Arsitektur Faster R-CNN

Kedua algoritma di atas (R-CNN dan Cepat R-CNN) menggunakan pencarian selektif untuk mengetahui proposal kawasan. Pencarian selektif adalah proses yang lambat dan memakan waktu yang mempengaruhi kinerja jaringan. Oleh karena itu, Shaoqing Ren et al datang dengan algoritma deteksi objek yang menghilangkan algoritma pencarian selektif dan memungkinkan jaringan mempelajari proposal wilayah.

Mirip dengan Fast R-CNN, gambar disediakan sebagai input ke jaringan convolutional yang menyediakan peta fitur convolutional. Alih-alih menggunakan algoritma pencarian selektif pada peta fitur untuk mengidentifikasi proposal wilayah, jaringan yang terpisah digunakan untuk memprediksi proposal wilayah. Proposal wilayah yang diprediksi kemudian dibentuk kembali menggunakan lapisan pool RoI yang kemudian digunakan untuk mengklasifikasikan gambar dalam wilayah yang diusulkan dan memprediksi nilai offset untuk kotak pembatas.



Gambar 2. 22 Perbandingan Kecepatan Uji Waktu Dari Algoritma Objek Deteksi

Dari grafik di atas, Anda dapat melihat bahwa R-CNN lebih cepat jauh lebih cepat daripada pendahulunya. Oleh karena itu, bahkan dapat

digunakan untuk deteksi objek real-time. Pada awalnya metode R-CNN bertujuan untuk fokus pada mengembangkan kehandalan akurasi dalam mendeteksi objek. Sebelum faster R-CNN diperkenalkan, berikut perkembangan metode R-CNN seiring berjalannya waktu.



Gambar 2. 23 Faster R-CNN

Metode faster RCNN menggunakan RPN (Region Proposal Network) untuk menentukan kandidat boundary box. Setiap kandidat boundary box akan masuk tahap klasifikasi untuk mendeteksi kelas yang terdapat pada boundary box tersebut. RPN ini merupakan pengganti Selective Search pada metode sebelumnya yaitu Fast R-CNN. Dibandingkan SSD dan YOLO, Faster R-CNN ini dinilai lebih lambat karena terlalu banyak kandidat box yang perlu masuk bagian klasifikasi.

Tahapan Faster R-CNN adalah sebagai berikut :

1. Kalkulasi kandidat bounding box menggunakan RPN (Region Proposal Network)

Berikut tahapan RPN :

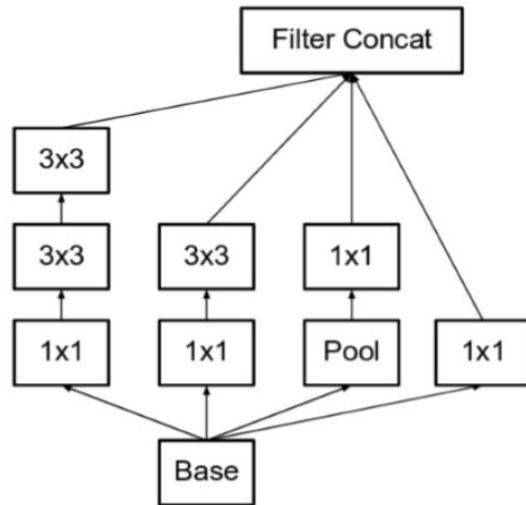
- Input gambar pertama kali akan masuk kedalam CNN untuk mereduksi dimensi gambar serta untuk memperoleh feature map
  - Pada layer terakhir dari CNN diatas dilakukan konvolusi lagi menggunakan sliding window dengan ukuran 3x3
  - Untuk setiap sliding window menghasilkan sejumlah n tetap anchor box
  - Lalu dilakukan seleksi untuk setiap box yang dihasilkan sesuai dengan skor adanya objek pada box tersebut
2. Jalankan CNN untuk setiap bounding box yang dihasilkan dari RPN

3. Setiap output yang dihasilkan CNN menjadi input SVM (Support Vector Machine) untuk klasifikasi kelasnya dan juga sebagai input untuk linear regression untuk memperbaiki bounding box agar lebih presisi

Faster RCNN merupakan salah satu metode deep learning yang digunakan untuk mengenali suatu suatu objek pada citra. Pengenalan dilakukan dengan menelusuri ciri-ciri yang dimiliki oleh objek pada citra. Penelusuran dilakukan melalui sejumlah layer (seperti yang dilakukan pada neural network) melalui proses konvolusi atau yang lebih dikenal dengan nama Convolutional Neural Network (CNN).

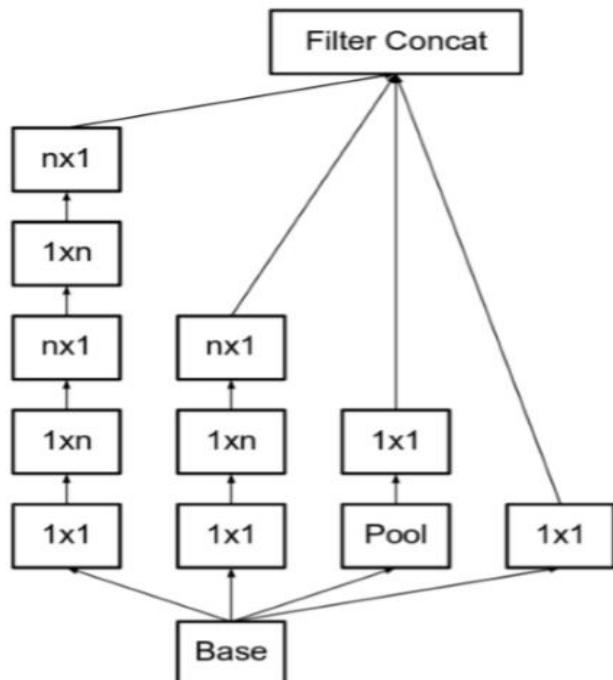
CNN memiliki berbagai arsitektur, salah satunya dalam penelitian ini adalah Inception V2. Arsitektur dari Inception V2 dirancang untuk mengurangi kompleksitas CNN, yang dilakukan dengan cara menyusun arsitektur yang lebih melebar dari pada mendalam. Inception V2 memiliki 3 modul yang ditunjukkan oleh Gambar 4. Modul pertama (Gambar 4.a) menggantikan konvolusi  $5 \times 5$  menjadi  $3 \times 3$ . Selanjutnya pemfaktoran konvolusi dilakukan (ditunjukkan pada Gambar 4.b). Terakhir modul diubah lebih melebar untuk mengurangi kompleksitas jaringan konvolusi (ditunjukkan pada Gambar 4.c)

a. Modul Pertama Inception V2



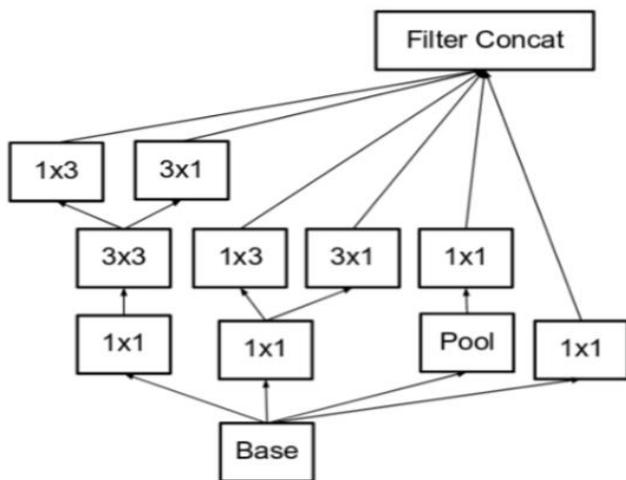
Gambar 2. 24 Modul Pertama Inception V2

b. Modul kedua Inception V2



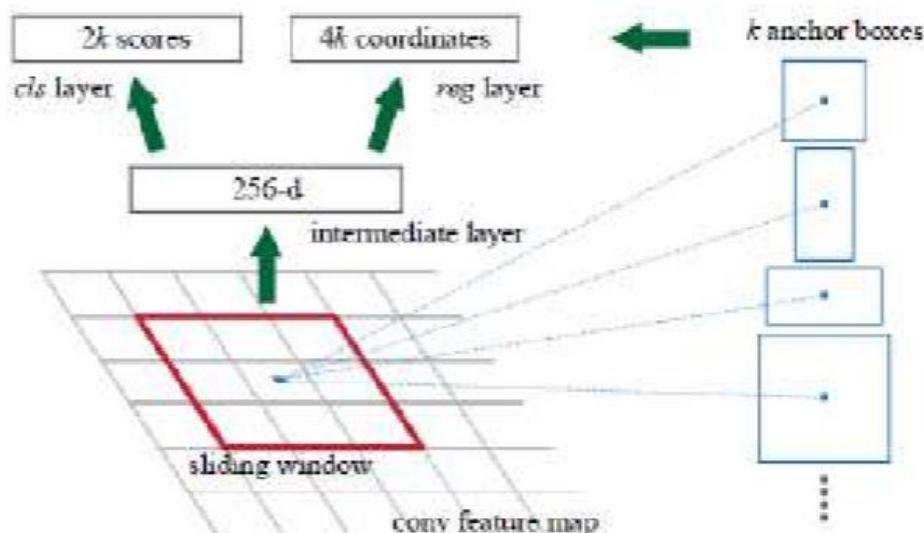
Gambar 2. 25 Modul kedua Inception V2

c. Modul Ketiga Inception V2



Gambar 2. 26 Modul Ketiga Inception V2

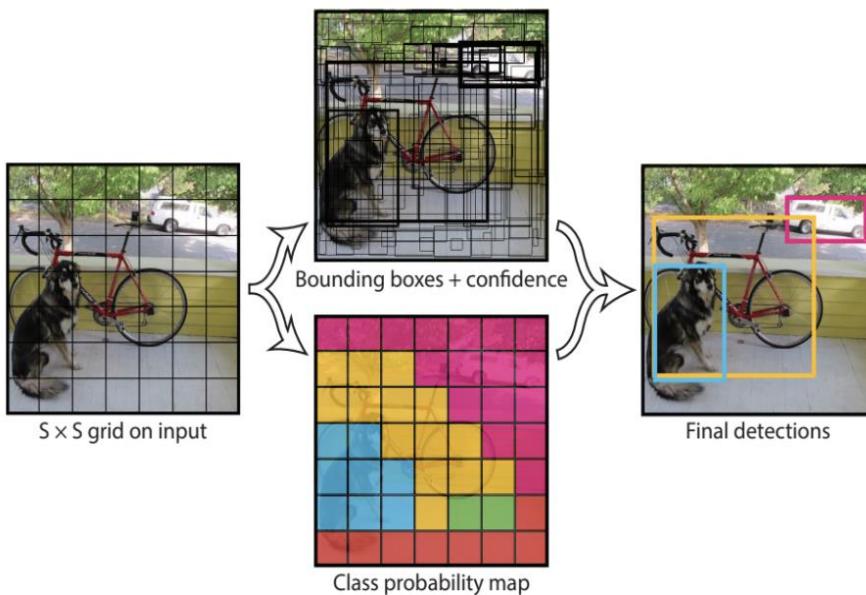
Selanjutnya, berbeda dengan metode RCNN sebelumnya, maka Faster-RCNN membuat perubahan dengan memunculkan Region Proposal Network (RPN) ditunjukkan oleh Gambar berikut :



Gambar 2. 27 Region Proposal Network (RPN)

## 2.5 Pengantar YOLO - You Only Look Once

Semua algoritme deteksi objek sebelumnya menggunakan wilayah untuk melokalkan objek dalam gambar. Jaringan tidak melihat gambar lengkap. Sebagai gantinya, bagian dari gambar yang memiliki probabilitas tinggi mengandung objek. YOLO atau You Only Look Once adalah algoritma deteksi objek yang jauh berbeda dari algoritma berbasis wilayah yang terlihat di atas. Di YOLO, satu jaringan konvolusional memprediksi kotak pembatas dan probabilitas kelas untuk kotak-kotak ini.



Gambar 2. 28 Cara Kerja Yolo

Cara kerja YOLO adalah bahwa kami mengambil gambar dan membaginya menjadi kisi  $S \times S$ , di dalam masing-masing kisi tersebut kami mengambil kotak pembatas. Untuk setiap kotak pembatas, jaringan mengeluarkan probabilitas kelas dan mengimbangi nilai untuk kotak pembatas. Kotak pembatas yang memiliki probabilitas kelas di atas nilai ambang dipilih dan digunakan untuk menemukan objek di dalam gambar.

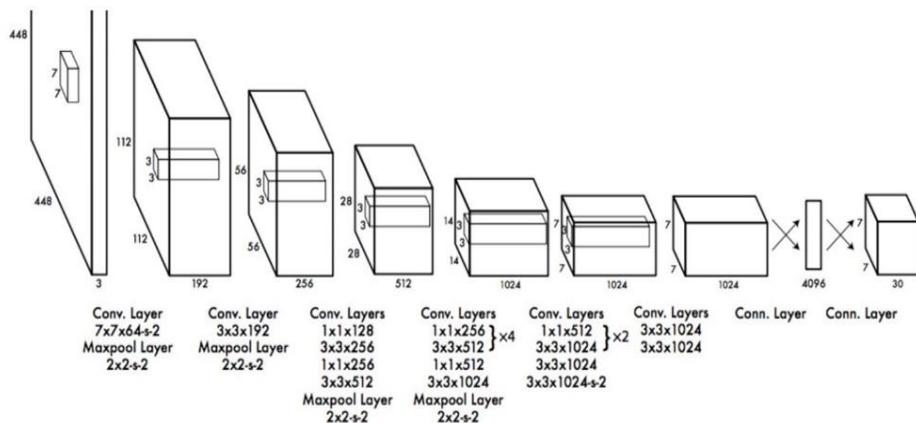
YOLO adalah urutan besarnya lebih cepat (45 frame per detik) daripada algoritma deteksi objek lainnya. Keterbatasan algoritma YOLO adalah bahwa ia berjuang dengan benda-benda kecil di dalam gambar, misalnya mungkin mengalami kesulitan dalam mendeteksi kawanan burung. Ini karena kendala spasial dari algoritma.

Target dari metode YOLO adalah untuk dapat mendeteksi objek secara real time. Secara umum metode YOLO mempunyai tahapan yang paling simpel dibanding SSD maupun faster R-CNN. Sebagian besar tahapan YOLO seperti halnya klasifikasi gambar menggunakan CNN biasa. YOLO melakukan kalkulasi bounding box dengan satu skala fitur map.

Tahapan Yolo adalah sebagai berikut :

- Input gambar ditandai kedalam sejumlah  $S \times S$  grid box
- Setiap grid box akan membentuk sejumlah B “bounding box” yang tetap, dengan berbagai ukuran yang bervariasi. Variasi bounding box sama untuk setiap grid box. Setiap bounding box yang terbentuk dari grid box tertentu berada tepat ditengah grid box tersebut. Ukuran bounding box bisa lebih besar dari ukuran grid box nya.
- Jika pada bounding box terdapat objek, maka grid box yang membentuk bounding box tersebut bertanggung jawab untuk mendeteksi objek tersebut.
- Satu grid box hanya boleh mendeteksi satu objek saja. Sehingga jika lebih dari satu bounding box untuk satu grid box tertentu mendeteksi adanya objek, maka hanya diperbolehkan untuk memilih satu objek dengan satu bounding box saja yang memiliki tingkat keyakinan paling tinggi.

- “Non max supression” digunakan untuk menentukan saat terdapat banyak bounding box terdeteksi untuk satu objek yang sama.



Gambar 2. 29 Arsitektur Yolo

Tipe YOLO :

- YOLO
- YOLO v2
- YOLO9000
- YOLO v3

## **BAB III**

---

### **PENJELASAN TOOLS DAN BAHASA PEMOGRAMAN YANG DIGUNAKAN**

#### **3.1 Tools Yang Digunakan**

##### **3.1.1 Sublime**

Aplikasi Sublime Text merupakan aplikasi editor yang digunakan untuk kode dan teks yang dapat berjalan diberbagai platform operating system dengan menggunakan teknologi Phyton API. Aplikasi ni diciptakan karena terinspirasi dari aplikasiVim, Aplikasi ini sangatlah fleksibel dan powerfull. Aplikasi ini dapat dikembangkan dengan menggunakan sublime-packages.

Sublime Text bukanlah aplikasi opensource dan juga aplikasi yang dapat digunakan dan didapatkan secara gratis, namun beberapa fitur pengembangan fungsionalitas (packages) dari aplikasi ini merupakan hasil dari temuan dan memperoleh dukungan penuh dari komunitas serta memiliki linsensi aplikasi gratis. Aplikasi ini mendukung berbagai bahasa pemrograman dan mampu menyajikan fitur syntax highlight hampir di semua bahasa pemrogramman yang didukung ataupun dikembangkan oleh komunitas.

```

D:\INTERNSHIP_TELKOM\plate_detection\detection\detection.py (object_detection, SCRIPT MIA, plate_detection) - Sublime Text (UNREGISTERED)
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERES detector.py Object.detection.inference Object.detection.imagepy Object.detection.webcam.py
> object_detection 18
> SCRIPT MIA 19
> plate_detection 20
> < detector 21
>   < __pycache__ 22
>     /> detector.py 23
>       frozen_inference_graph.py 24
>       labeling_pbtxt 25
>     /> test_image 26
>     /> utils 27
>     < __pycache__ 28
>       /> plot.py 29
>       Object_detection_imagepy 30
>       Object_detection_video.py 31
>     Object_detection_webcam.py 32
> requirements.txt 33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

# Mengarahkan path ke frozen.pb yang berisikan model yang
# akan digunakan untuk sistem objek deteksi
PATH_TO_CKPT = os.path.sep.join([folder_detector, 'frozen_inference_graph.pb'])

# Mengarahkan path ke labelmap.pbtxt untuk mengetahui jumlah class
PATH_TO_LABELS = os.path.sep.join([folder_detector, 'labelmap.pbtxt'])

# Jumlah kelas yang bisa diidentifikasi oleh objek detektor adalah 1 yaitu plate
NUM_CLASSES = 1

# Muat model Tensorflow ke dalam memori untuk persiapan sessions tensorflow.
# Session ini yang dipanggil sama detector
self.detection_graph = tf.Graph()
with self.detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

self.sess = tf.Session(graph=self.detection_graph)

# Inialisasi variabel untuk penimpanan sementara saat testing
# Tetapan tensor input dan output (mis. Data) untuk classifier objek Input tensor adalah gambar
self.image_tensor = self.detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensor adalah kotak deteksi, skor, dan kelas
# Setiap kotak membakli bagian dari gambar di mana objek tertentu terdeteksi
self.detection_boxes = self.detection_graph.get_tensor_by_name('detection_boxes:0')

# Setiap skor membakli tingkat kepercayaan untuk masing-masing objek.
# Skor ditampilkan pada gambar hasil, bersama dengan label kelas.
self.detection_scores = self.detection_graph.get_tensor_by_name('detection_scores:0')
self.detection_classes = self.detection_graph.get_tensor_by_name('detection_classes:0')

# Timlah chiec_vann ferderable

```

Gambar 3. 1 Tampilan Sublime

Berikut beberapa fitur yang diunggulkan dari aplikasi Sublime Text:

- Goto Anything

Fitur yang sangat membantu dalam membuka file ataupun menjelajahi isi dari file hanya dengan beberapa keystrokes.

- Multiple Selections

Fitur ini memungkinkan user untuk mengubah secara interaktif banyak baris sekaligus, mengubah nama variabel dengan mudah, dan memanipulasi file lebih cepat dari sebelumnya.

- Command Pallete

Dengan hanya beberapa keystrokes, user dapat dengan cepat mencari fungsi yang diinginkan, tanpa harus menavigasi melalui menu.

- Distraction Free Mode

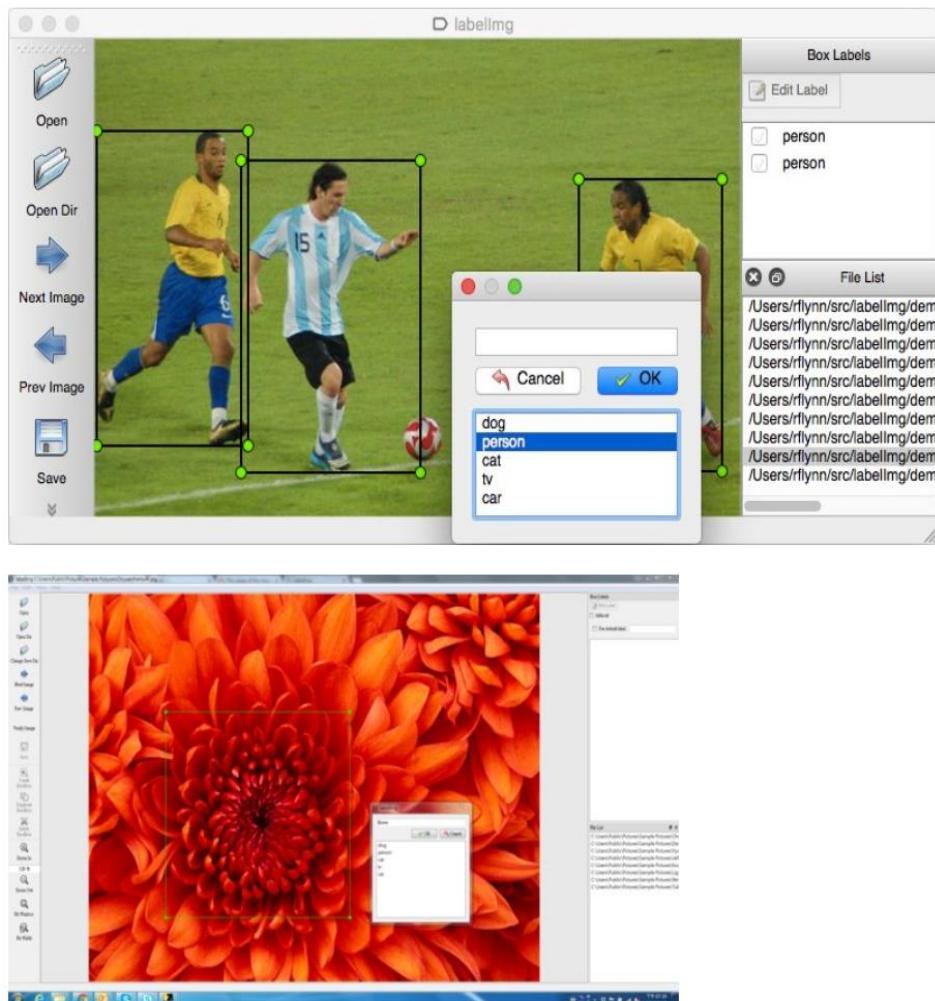
Bila user memerlukan fokus penuh pada aplikasi ini, fitur ini dapat membantu user dengan memberikan tampilan layar penuh.

- Split Editing

- Dapatkan hasil yang maksimal dari monitor layar lebar dengan dukungan editing perpecahan. Melakukan editing di sisi file dengan sisi, atau mengedit dua lokasi di satu file. Anda dapat mengedit dengan banyak baris dan kolom yang user inginkan.
- Instant Project Switch  
Mengambil seluruh file yang dimasukkan kedalam project pada aplikasi ini terhubung dengan fitur Goto Anything untuk menjelajahi semua file yang ada ataupun untuk beralih ke file dalam project lainnya dengan cepat.
- Plugin API  
Aplikasi ini memiliki plugin API berbasis Phyton sehingga membuat aplikasi ini sangat tangguh.
- Customize Anything  
Aplikasi sublime mempunyai dan memberikan user fleksibilitas dalam hal pengaturan fungsional dalam aplkasi ini.
- Cross Platform  
Aplikasi ini dapat berjalan hampir disemua operating system modern seperti Windows, OS X, dan Linux based operating system.

### 3.1.2 LabelImg

Salah satu alat Anotasi Gambar untuk Penglihatan Komputer adalah LabelImg. LabelImg adalah alat anotasi gambar grafis open source yang dapat Anda gunakan untuk memberi label pada kotak pembatas objek dalam gambar. LabelImg adalah alat anotasi gambar grafis. Ini ditulis dalam Python dan menggunakan Qt untuk antarmuka grafisnya. Anotasi disimpan sebagai file XML dalam format PASCAL VOC, format yang digunakan oleh ImageNet Selain itu, ia juga mendukung format YOLO.



Gambar 3. 2 Tampilan LabelImg

LabelImg adalah alat anotasi gambar grafis. Ini ditulis dalam Python dan menggunakan Qt untuk antarmuka grafis nya. Anotasi disimpan sebagai file XML dalam format PASCAL VOC, format yang digunakan oleh ImageNet. Untuk mendapatkan aplikasi Labelimg tersebut kita bisa langsung men download nya pada google.

Pada aplikasi LabelImg tersebut terdapat beberapa menu seperti menu untuk membuka file yang akan kita beri label, menu untuk menyimpan hasil dari pekerjaan kita, menu untuk next dan prev pada gambar, menu

untuk membuat rectBox, menu untuk men duplicate RectBox dan masih banyak lagi menu lainnya.

### 3.1.3 CUDA

CUDA (Compute Unified Device Architecture) adalah platform komputasi paralel dan model antarmuka pemrograman aplikasi (API) yang dibuat oleh Nvidia. Ini memungkinkan pengembang perangkat lunak dan insinyur perangkat lunak untuk menggunakan unit pemrosesan grafis (GPU) yang diaktifkan CUDA untuk pemrosesan tujuan umum - suatu pendekatan yang disebut GPGPU (komputasi Tujuan Umum pada Unit Pemrosesan Grafik). CUDA adalah sebuah Platform yang memiliki lapisan perangkat lunak yang memberikan akses langsung ke set instruksi virtual GPU dan elemen komputasi paralel, untuk eksekusi kernel komputasi.

CUDA adalah platform komputasi paralel dan model antarmuka pemrograman aplikasi yang dibuat oleh Nvidia dan memungkinkan pengembang perangkat lunak dapat menggunakan untuk melakukan suatu kesatuan proses grafik yang diaktifkan CUDA. Oleh karena itu dalam model CUDA terdapat CUDA yang digunakan untuk mengalokasikan memori pada perangkat dengan tujuan untuk melakukan peluncuran kernel sedangkan CUDA yang digunakan untuk memproses satuan grafik. Berdasarkan referensi tersebut, CUDA digunakan oleh peneliti untuk membantu proses kerja yang terjadi pada tensorflow dalam memproses sebuah grafik yang terdapat pada sebuah gambar.

Platform CUDA dirancang untuk bekerja dengan bahasa pemrograman seperti C, C ++, dan Fortran. Aksesibilitas ini memudahkan spesialis dalam pemrograman paralel untuk menggunakan sumber daya GPU, berbeda dengan API sebelumnya seperti Direct3D dan OpenGL, yang membutuhkan keterampilan tingkat lanjut dalam pemrograman

grafis. GPU yang didukung CUDA juga mendukung kerangka kerja pemrograman seperti OpenACC dan OpenCL dan HIP dengan mengkompilasi kode tersebut ke CUDA. Ketika CUDA pertama kali diperkenalkan oleh Nvidia, namanya adalah singkatan dari Compute Unified Device Architecture, tetapi Nvidia kemudian membatalkan penggunaan umum dari akronim tersebut.

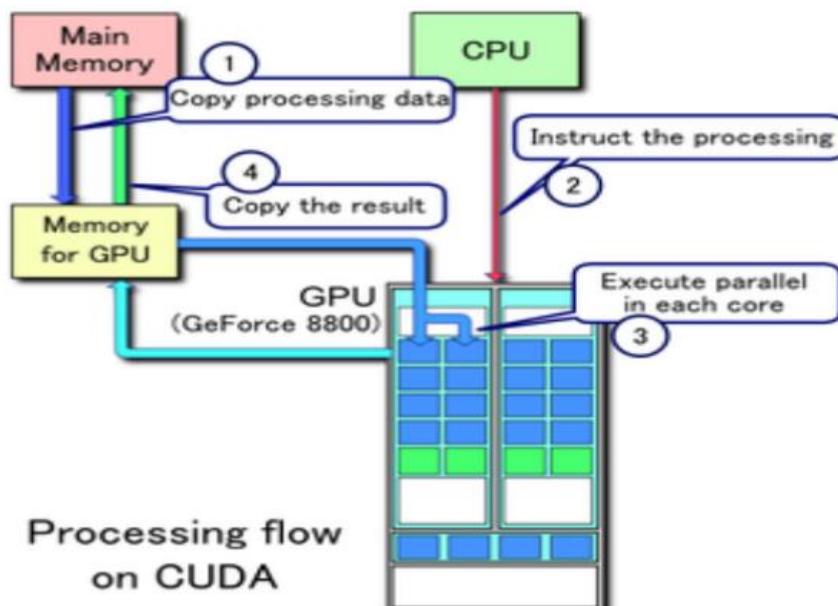
CUDA bisa diakses oleh pengembang perangkat lunak melalui perpustakaan yang dipercepat CUDA, arahan penyusun seperti OpenACC , dan ekstensi ke bahasa pemrograman standar industri termasuk C , C ++ dan Fortran . Pemrogram C / C ++ dapat menggunakan 'CUDA C / C ++', dikompilasi dengan nvcc , kompiler C / C ++ berbasis LLVM Nvidia. Pemrogram Fortran dapat menggunakan 'CUDA Fortran', yang dikompilasi dengan kompiler PGI CUDA Fortran dari The Portland Group.

Selain perpustakaan, arahan kompiler, CUDA C / C ++ dan CUDA Fortran, platform CUDA mendukung antarmuka komputasi lainnya, termasuk OpenCL Khronos Group, Microsoft DirectCompute, OpenGL Compute Shaders, dan C ++ AMP. Pembungkus pihak ketiga juga tersedia untuk Python, Perl, Fortran, Jawa, Ruby, Lua, Common Lisp, Haskell, R, MATLAB, IDL, Julia dan dukungan asli di Mathematica.\

ada industri game komputer , GPU digunakan untuk rendering grafik, dan untuk perhitungan fisika game (efek fisik seperti puing-puing, asap, api, cairan); contohnya termasuk PhysX dan Bullet . CUDA digunakan untuk mempercepat aplikasi non-grafis dalam biologi komputasi, kriptografi , dan bidang lainnya dengan urutan besarnya atau lebih.

CUDA memiliki beberapa keunggulan dibandingkan perhitungan tradisional untuk keperluan umum pada GPU (GPGPU) menggunakan API grafik:

- Pembacaan yang tersebar - kode dapat membaca dari alamat yang berubah-ubah dalam memori
- Memori virtual terpadu (CUDA 4.0 dan lebih tinggi)
- Memori terpadu (CUDA 6.0 ke atas)
- Memori bersama - CUDA memperlihatkan wilayah memori bersama cepat yang dapat dibagi di antara utas. Ini dapat digunakan sebagai cache yang dikelola pengguna, memungkinkan bandwidth yang lebih tinggi daripada yang dimungkinkan menggunakan pencarian tekstur
- Unduhan dan pembacaan lebih cepat ke dan dari GPU
- Dukungan penuh untuk operasi integer dan bitwise, termasuk pencarian tekstur integer

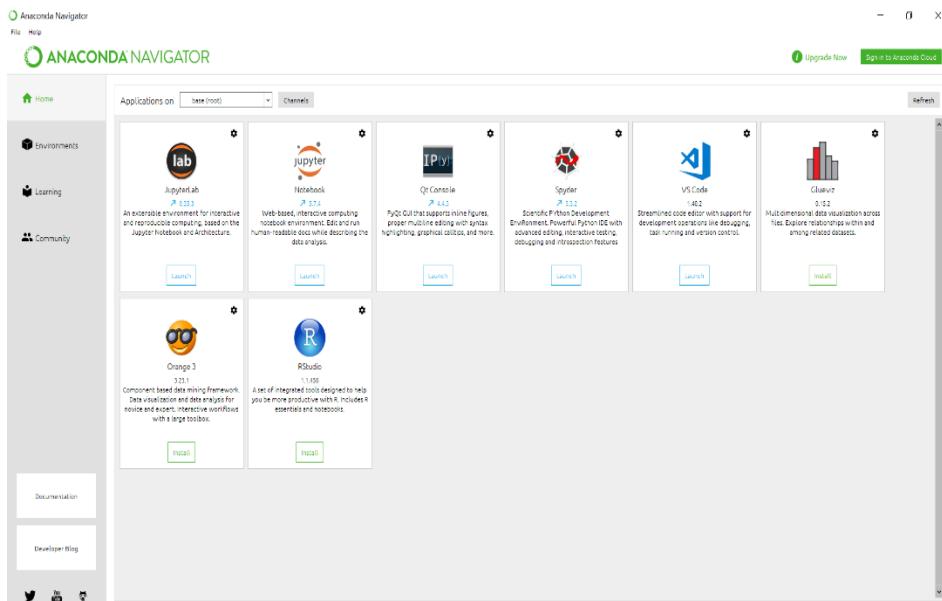


Gambar 3. 3 Aliran Pemrosesan CUDA

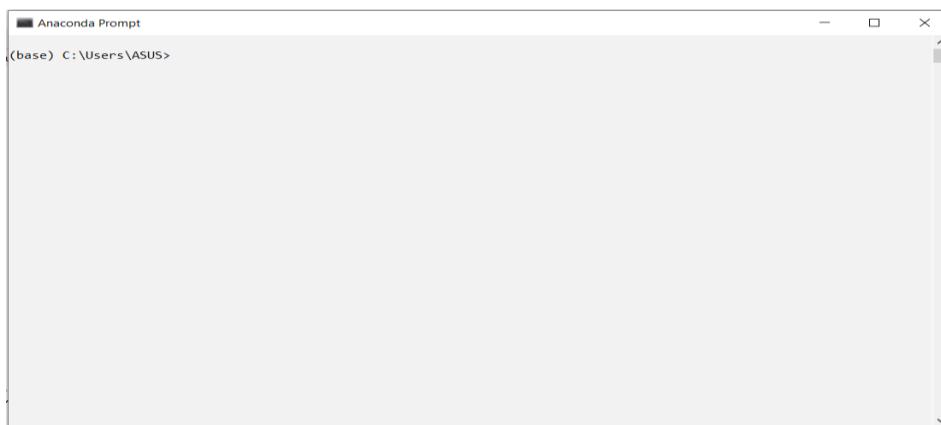
### **3.1.4 Anaconda**

Tools Anaconda merupakan platform ilmu data berbasis di sekitar bahasa pengkodean Python. Tujuan utama dari paket ini adalah untuk memungkinkan organisasi untuk berhasil mengamankan, menafsirkan, skala dan menyimpan data yang sangat penting untuk operasi sehari-hari mereka. Diperkirakan lebih dari 4,5 juta pengguna telah mengunduh paket ini. Melakukan sains data Python / R dan pembelajaran mesin di Linux, Windows, dan Mac OS X. Dengan lebih dari 15 juta pengguna di seluruh dunia, ini adalah standar industri untuk pengembangan, pengujian, dan pelatihan tentang mesin tunggal, memungkinkan para ilmuwan data individu untuk:

- Cepat mengunduh 1.500+ paket sains data Python / R
- Kelola perpustakaan, dependensi, dan lingkungan dengan Conda
- Mengembangkan dan pembelajaran mesin kereta api dan model pembelajaran yang mendalam dengan scikit-belajar , TensorFlow , dan Theano
- Menganalisis data dengan skalabilitas dan kinerja dengan Dask , NumPy , panda , dan Numba
- Visualisasikan hasil dengan Matplotlib, Bokeh, Datashader, dan Holoviews



Gambar 3. 4 Tampilan Anaconda Desktop



Gambar 3. 5 Tampilan Anaconda Desktop

Anaconda merupakan bundel perangkat lunak tingkat perusahaan yang menyediakan sejumlah opsi inovatif untuk pengguna akhir. Beberapa manfaat inti termasuk peningkatan kolaborasi antardepartemen, kemampuan untuk mereproduksi data, skalabilitas superior dan berbagai lapisan keamanan. Fungsi utama lainnya dari paket ini adalah memungkinkan organisasi untuk mengelola dan menafsirkan data besar

dengan lebih baik; faktor kunci untuk sukses dalam lingkungan bisnis modern. Ini mempekerjakan sejumlah sumber data untuk menjamin redundansi. Ini termasuk (tetapi tidak terbatas pada) penyimpanan berbasis cloud, SQL, NoSQL, dan File Datar.

Anaconda memiliki sifat yang modular, sehingga dapat disesuaikan tergantung kebutuhan organisasi yang bersangkutan. Karena hal itu mendorong kolaborasi waktu nyata, tingkat efisiensi internal juga akan ditingkatkan. User dapat menikmati dukungan teknis real-time selama penyebaran kode open-source dan karena Anaconda sepenuhnya kompatibel dengan bahasa Python, kurva pembelajaran secara keseluruhan telah berkurang secara dramatis.

## **3.2 Bahasa Pemograman**

### **3.2.1 Python**

Bahasa pemrograman yang interpretatif dan multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode salah satunya adalah Bahasa Python. Memiliki fungsi sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas yang memiliki fungsionalitas pustaka standar yang besar serta komprehensif. Bahasa pemograman python juga didukung oleh komunitas yang besar. Bahasa ini banyak digunakan padaprogrammer karena dikenal dengan bahasa pemrograman yang mudah dipelajari, karena struktur sintaknya rapi dan mudah dipahami.

Bahasa pemograman ini juga mensupport multi paradigma pemrograman, utamanya namun tidak dibatasi pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Beberapa fitur yang tersedia pada Bahasa python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori

otomatis. Python pada umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Pemograman python juga dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

Python saat ini bias dijalankan di berbagai platform system operasi, diantaranya adalah sebagai berikut :

- Linux/Unix
- Windows
- Mac OS X
- Java Virtual Machine
- OS/2
- Amiga
- Palm
- Symbian (untuk produk-produk Nokia)

Bahasa pemrograman ini dihubungkan dengan beberapa lisensi yang berbeda dari beberapa versi. Lihat sejarahnya di Python Copyright. Namun pada prinsipnya Python dapat diperoleh dan dipergunakan secara bebas, bahkan untuk kepentingan komersial.

## **BAB IV**

---

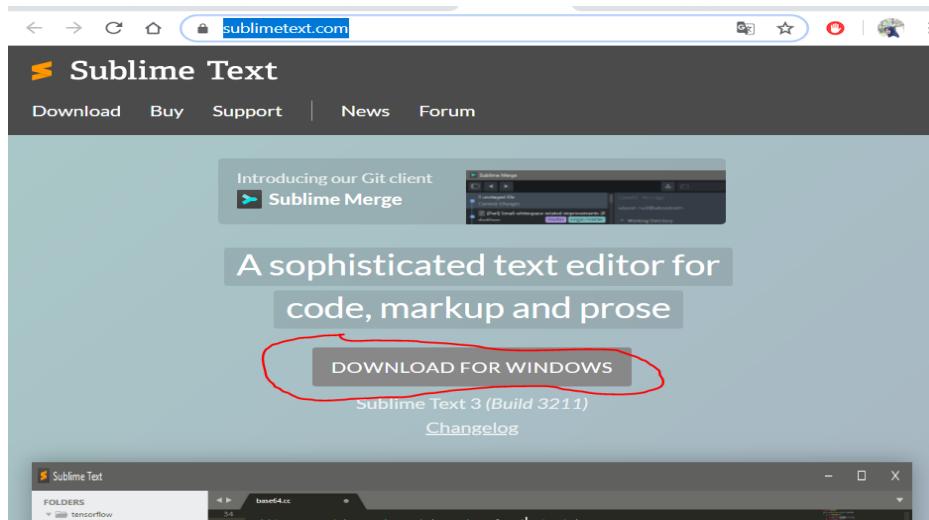
### **INSTALASI TOOLS YANG DIGUNAKAN**

#### **4.1 Tools Yang Digunakan**

##### **4.1.1 Instalasi Sublime**

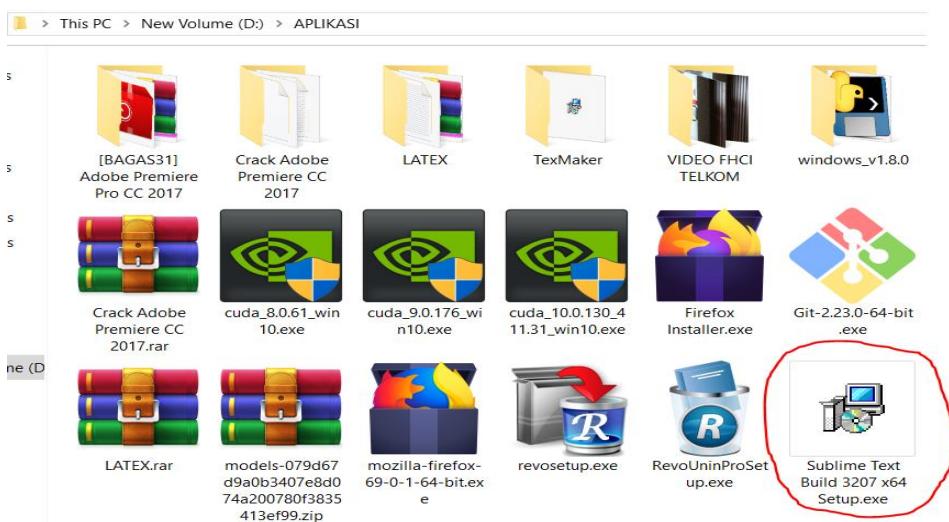
Berikut adalah langkah-langkah untuk melakukan instalasi Sublime :

1. Tahapan yang pertama untuk melakukan instalasi maka download dahulu file .exe pada link berikut :  
<https://www.sublimetext.com/>.



Gambar 4. 1 Download file .exe

2. Tahapan yang kedua setelah selesai mendownload maka buka directory dimana file .exe tersimpan, kemudian Double klik file .exe hasil download tadi :



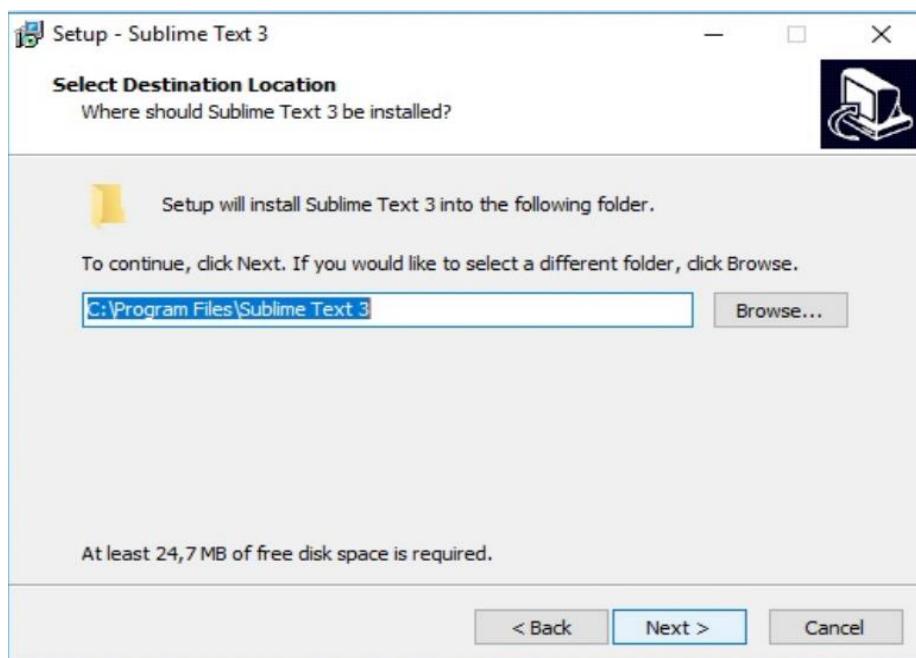
Gambar 4. 2 Double Klik File .exe

3. Tahapan ketiga jika sudah double klik file .exe maka akan muncul tampilan berikut kemudian klik next :



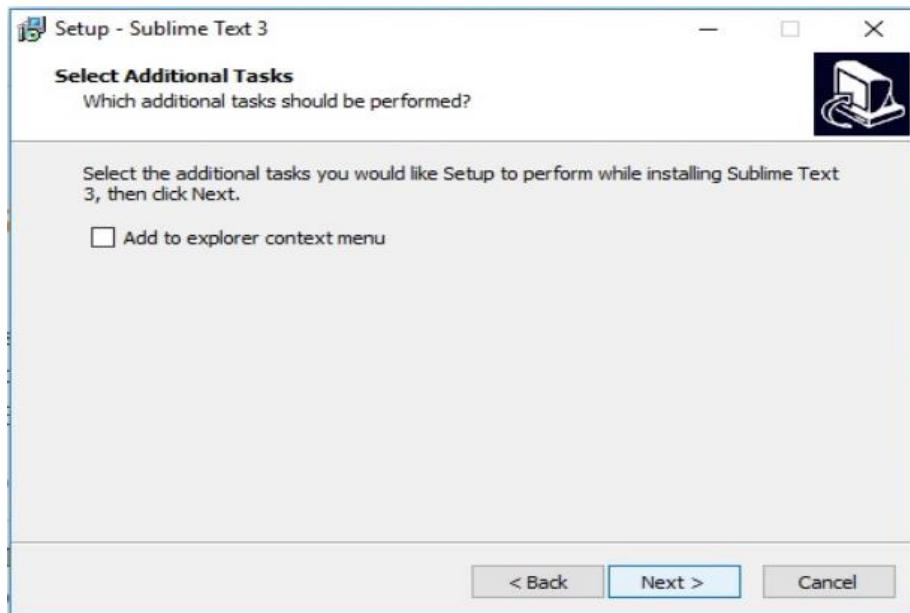
Gambar 4. 3 Proses Instalasi Sublime

4. Selanjutnya klik next



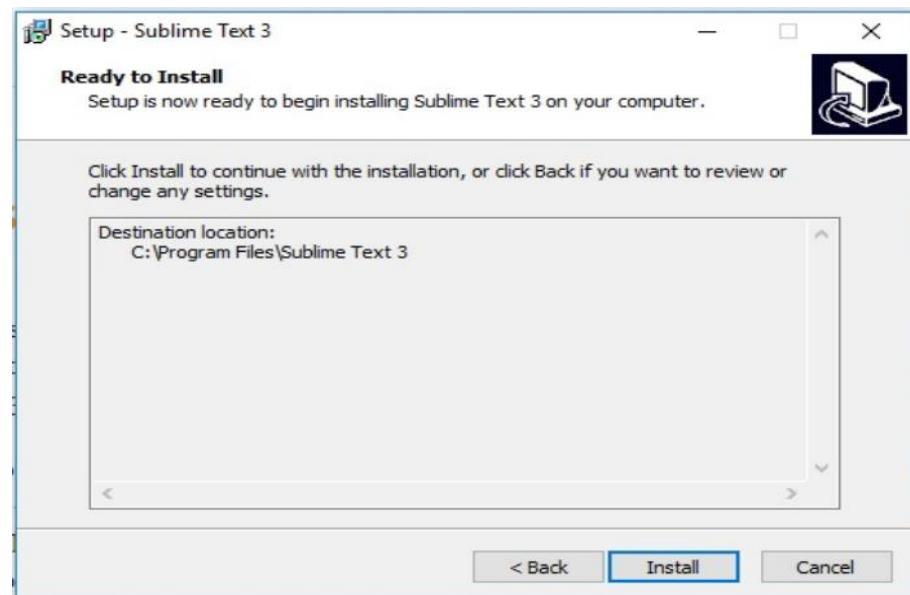
Gambar 4. 4 Proses Instalasi Sublime

5. Selanjutnya klik next



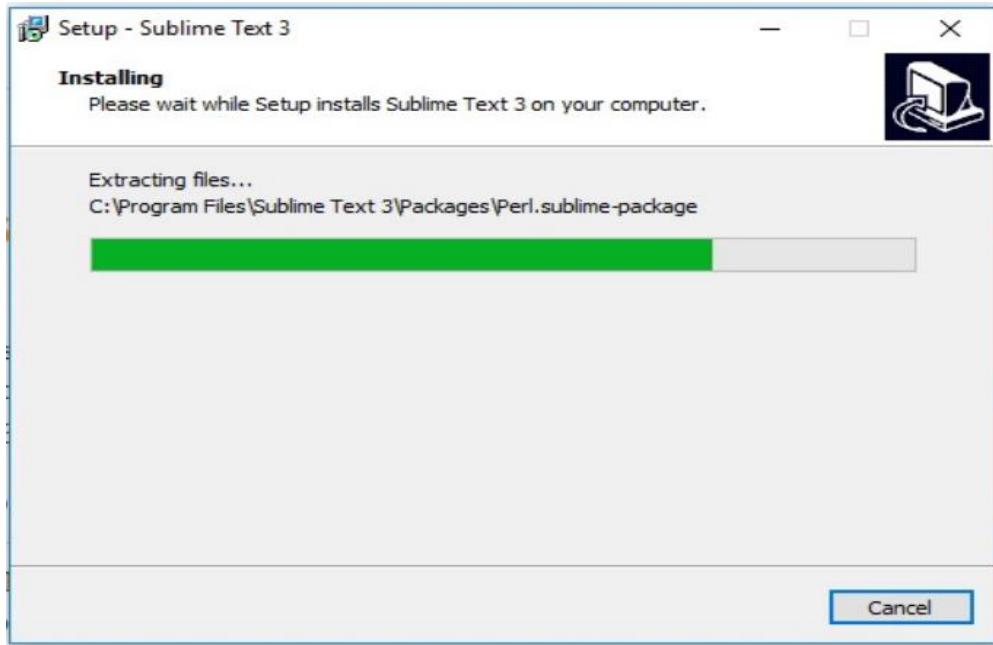
Gambar 4. 5 Proses Instalasi Sublime

6. Selanjutnya klik next



Gambar 4. 6 Proses Instalasi Sublime

7. Tunggu sampai proses instalasi selesai, Setelah proses ini selesai maka sumblime telah bisa digunakan.



Gambar 4. 7 Proses Instalasi Sublime

#### 4.1.2 Instalasi LabelImg

Berikut adalah langkah-langkah untuk melakukan instalasi LabelImg:

1. Untuk melakukan instalasi LabelImg maka pertama-tama melakukan download file .exe pada link berikut ini :  
<https://tzutalin.github.io/labelImg/>  
Sesuai kan dengan system operasi yang anda gunakan .
2. Anaconda Prompt dan buka direktori labelImg

```
Conda install pyqt = 5
pyrcc5 -o libs / resources.py resources.qrc
python labelImg.py
python labelImg.py [IMAGE_PATH] [FILE KELAS PRE-DEFINISI]
```

Gambar 4. 8 Instalasi LabelImg

### 3. Pemakaian

- Langkah (PascalVOC)

- Bangun dan luncurkan dengan menggunakan instruksi di atas.
- Klik 'Ubah folder anotasi tersimpan standar' di Menu / File
- Klik 'Buka Dir'
- Klik 'Buat Kotak Panjang'
- Klik dan lepaskan mouse kiri untuk memilih wilayah untuk memberi anotasi pada kotak segi empat
- Anda dapat menggunakan mouse kanan untuk menarik kotak persegi panjang untuk menyalin atau memindahkannya

Anotasi akan disimpan ke folder yang Anda tentukan. Anda bisa merujuk ke hotkey di bawah ini untuk mempercepat alur kerja Anda.

- Langkah (YOLO)

- Dalam data/predefined\_classes.txt menentukan daftar kelas yang akan digunakan untuk pelatihan Anda.
- Bangun dan luncurkan dengan menggunakan instruksi di atas.
- Tepat di bawah tombol "Simpan" di bilah alat, klik tombol "PascalVOC" untuk beralih ke format YOLO.
- Anda dapat menggunakan Open / OpenDIR untuk memproses gambar tunggal atau ganda. Setelah selesai dengan satu gambar, klik simpan.

File txt dalam format YOLO akan disimpan di folder yang sama dengan gambar Anda dengan nama yang sama. File bernama "classes.txt" disimpan ke folder itu juga. "classes.txt" mendefinisikan daftar nama kelas yang dirujuk oleh label YOLO Anda.

catatan:

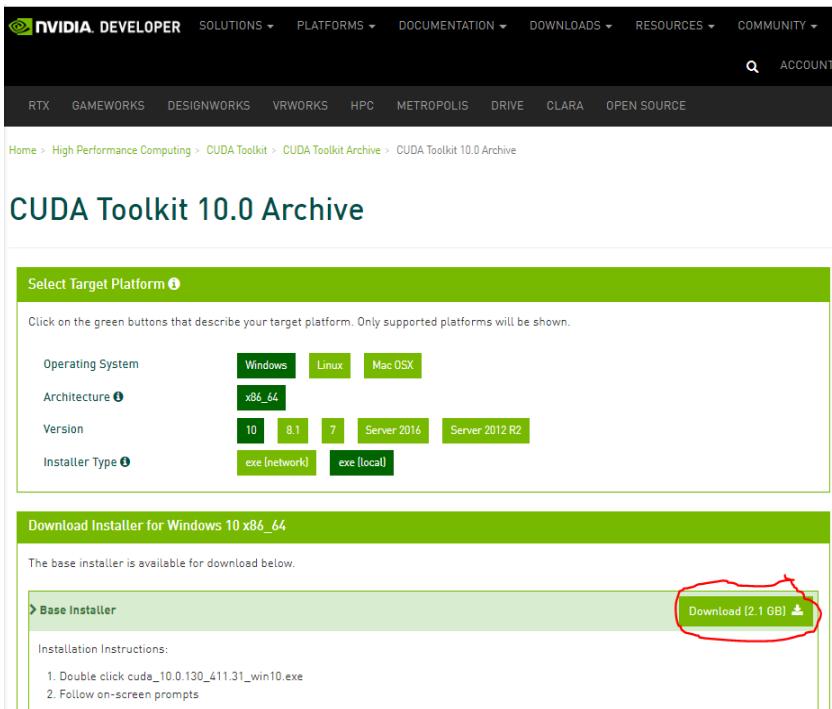
- Daftar label Anda tidak akan berubah di tengah pemrosesan daftar gambar. Saat Anda menyimpan gambar, classes.txt juga akan diperbarui, sedangkan anotasi sebelumnya tidak akan diperbarui.
- Anda seharusnya tidak menggunakan fungsi "kelas default" saat menyimpan ke format YOLO, itu tidak akan dirujuk.
- Saat menyimpan sebagai format YOLO, bendera "sulit" dibuang.

#### 4.1.3 Instalasi CUDA

Berikut adalah langkah-langkah untuk melakukan instalasi CUDA :

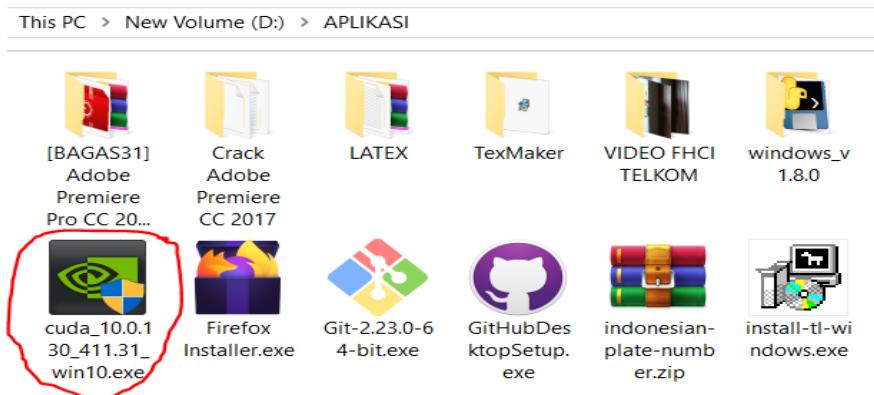
1. Langkah pertama yaitu download terlebih dahulu file .exe pada link berikut : <https://developer.nvidia.com/cuda-downloads>

Nanti kalian akan diminta memilih versi yang akan kalian download, pilih saja sesuai OS dan arsitektur komputer kalian.



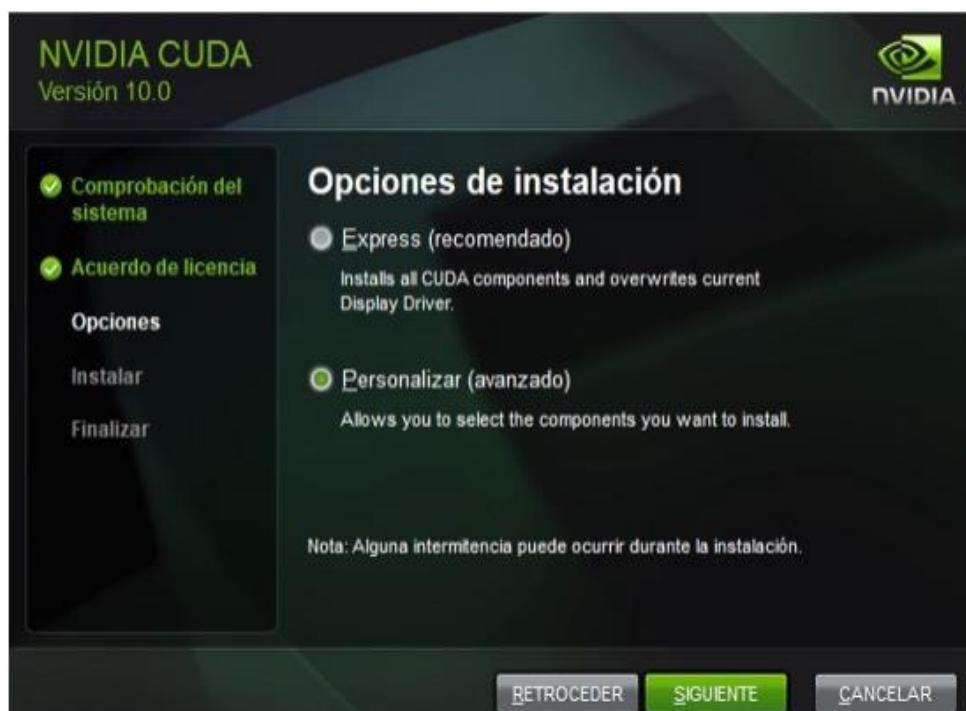
Gambar 4. 9 Download File .exe CUDA

2. Kemudian Proses instalasi dilakukan hanya dengan double klik file hasil download tadi



Gambar 4. 10 Melakukan Double Klik

3. Proses Instalasi CUDA hanya dengan klik next sampai prosesnya selesai, proses ini memakan waktu yang cukup lama.



Gambar 4. 11 Proses Instalasi CUDA

- Setelah Cuda toolkit ter-install maka selanjutnya adalah memindahkan cuDNN v7.0 ke folder instalasi selanjutnya mari kita buka hasil download cuDNN setelah itu masuk ke folder instalasi dimana(biasanya) terletak di D:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0

#### 4.1.4 Instalasi Anaconda

Berikut adalah langkah-langkah untuk melakukan instalasi Anaconda:

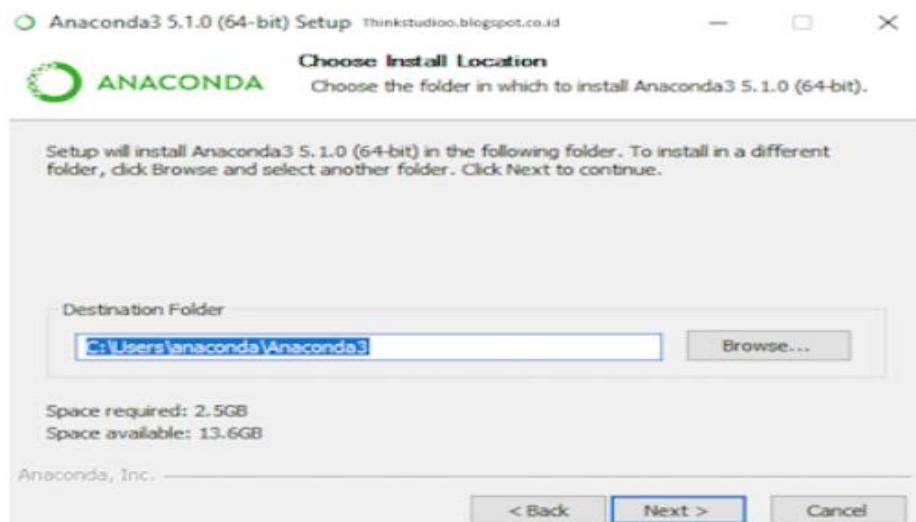
- Langkah pertama yaitu melakukan downlod file .exe pada link berikut ini :

<https://repo.continuum.io/archive/>

| Filename                           | Size   | Last Modified       | MD5                              |
|------------------------------------|--------|---------------------|----------------------------------|
| Anaconda2-5.1.0-Linux-ppc64le.sh   | 267.3M | 2018-02-15 09:08:49 | e894dcc547a1c7d67deb04f6bba7223a |
| Anaconda2-5.1.0-Linux-x86.sh       | 431.3M | 2018-02-15 09:08:51 | e26fb9d3e53049f6e32212270af6b987 |
| Anaconda2-5.1.0-Linux-x86_64.sh    | 533.0M | 2018-02-15 09:08:50 | 5b1b5784cae93cf696e11e66983d8756 |
| Anaconda2-5.1.0-MacOSX-x86_64.pkg  | 588.0M | 2018-02-15 09:08:52 | 4f9c197dfe6d3dc7e50a8611b4d3cfa2 |
| Anaconda2-5.1.0-MacOSX-x86_64.sh   | 505.9M | 2018-02-15 09:08:53 | e9845ccf67542523c5be09552311666e |
| Anaconda2-5.1.0-Windows-x86.exe    | 419.8M | 2018-02-15 09:08:55 | a09347a53e04a15ee965300c2b95dfde |
| Anaconda2-5.1.0-Windows-x86_64.exe | 522.6M | 2018-02-15 09:08:54 | b16d6d6858fc7decf671ac71e6d7cfdb |
| Anaconda3-5.1.0-Linux-ppc64le.sh   | 285.7M | 2018-02-15 09:08:56 | 47b5b2b17b7dbac0d4d0f0a4653f5b1c |
| Anaconda3-5.1.0-Linux-x86.sh       | 449.7M | 2018-02-15 09:08:58 | 793a94ee85baf64d0ebb67a0c49af4d7 |
| Anaconda3-5.1.0-Linux-x86_64.sh    | 551.2M | 2018-02-15 09:08:57 | 966406059cf7ed89cc82eb475ba506e5 |
| Anaconda3-5.1.0-MacOSX-x86_64.pkg  | 594.7M | 2018-02-15 09:09:06 | 6ed496221b843d1b5fe8463d3136b649 |
| Anaconda3-5.1.0-MacOSX-x86_64.sh   | 511.3M | 2018-02-15 09:10:24 | 047e12523fd287149ecd80c803598429 |
| Anaconda3-5.1.0-Windows-x86.exe    | 435.5M | 2018-02-15 09:10:28 | 7a2291ab99178a4cdec530861494531f |
| Anaconda3-5.1.0-Windows-x86_64.exe | 537.1M | 2018-02-15 09:10:26 | 83a8b1edcb21fa0ac481b23f65b604c6 |
| Anaconda2-5.0.1-Linux-x86.sh       | 413.2M | 2017-10-24 12:13:07 | ae155b192027e23189d723a897782fa3 |

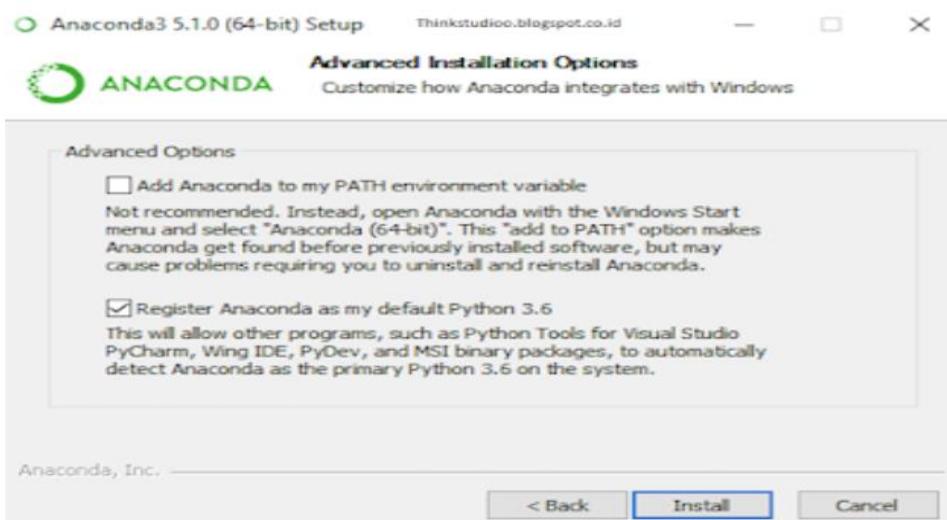
Gambar 4. 12 Download File Anaconda

- Setelah didownload sudah selesai lakukan double klik pada installer Anaconda. Kemudian klik next.
- Selanjutnya read lisensi dan klik I Agree.
- Kemudian pilih lokasi yang diinginkan, kalo saran saya dibiarkan default saja kemudian klik next.



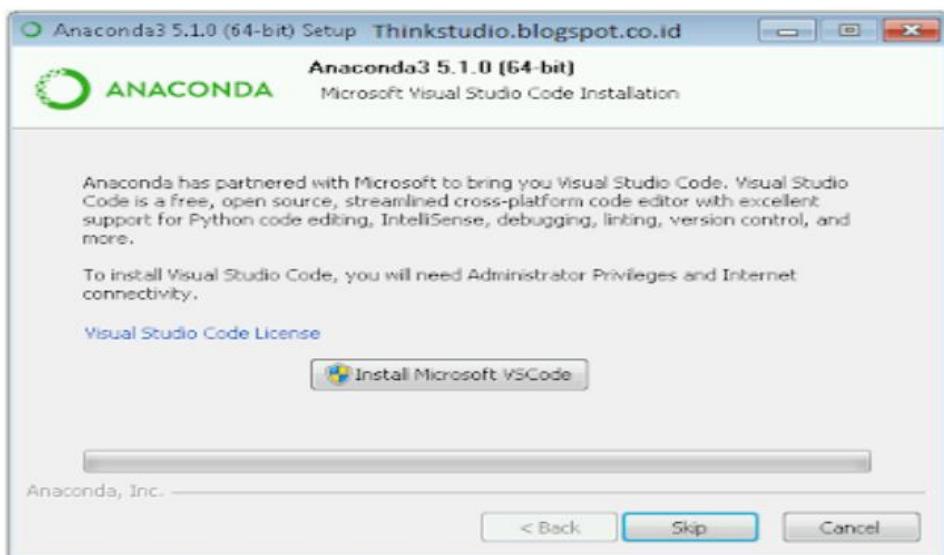
Gambar 4. 13 Proses Instalasi Anaconda

5. Kemudian pilih add anaconda to PATH atau tidak. Pilih apakah akan mendaftarkan Anaconda sebagai default Python 3.6? Kecuali kalian berencana menginstal dan menjalankan beberapa versi Anaconda, atau beberapa versi Python, biarkan default dan biarkan kotak ini dicentang. kemudian klik next.



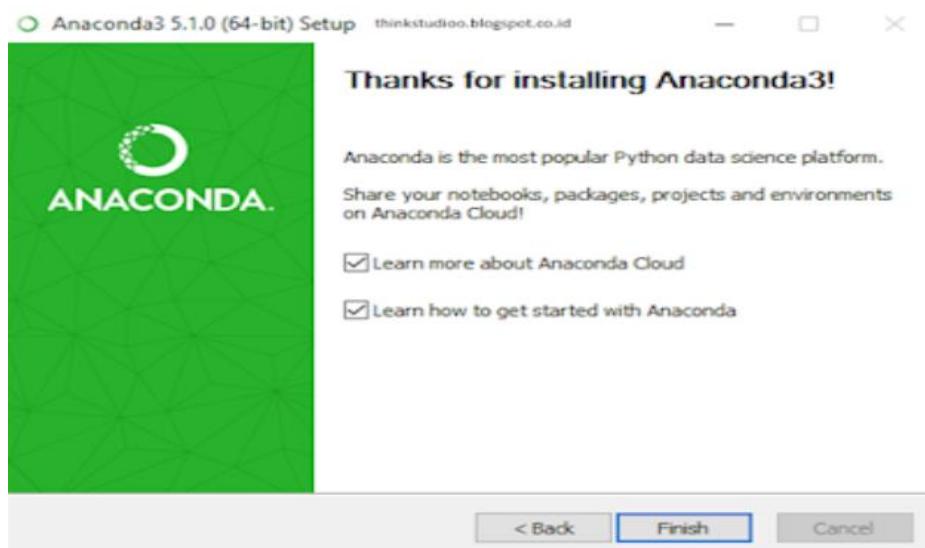
Gambar 4. 14 Proses Instalasi

6. Klik tombol Install. Jika Kalian ingin melihat packages Anaconda yang sedang dipasang, klik Show Details.
7. Kemudian Klik Next
8. Untuk menginstal VS Code, klik tombol Install Microsoft VS Code. Setelah instalasi selesai, klik tombol Next Atau untuk menginstal Anaconda tanpa VS code, klik tombol skip. Memasang VS code dengan pemasang Anaconda membutuhkan koneksi internet. Pengguna offline mungkin dapat menemukan pemasang offline VS Code dari Microsoft.



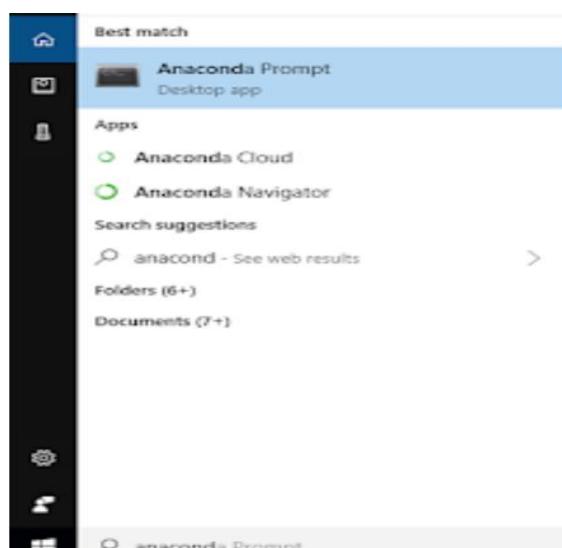
Gambar 4. 15 Install VS Code

9. Setelah instalasi yang sukses, Kalian akan melihat kotak dialog "Thanks for installing Anaconda3"



Gambar 4. 16 Instalasi Selesai

10. Bila instalasi sudah selesai kemudian verifikasi dengan membuka Anaconda Navigator, program yang disertakan dengan Anaconda. Search di windows kalian anaconda navigator. Jika Navigator terbuka, Kalian telah berhasil menginstal Anaconda. Jika tidak, periksa bahwa Kalian telah menyelesaikan setiap langkah di atas.



Gambar 4. 17 Buka Aplikasi Anaconda

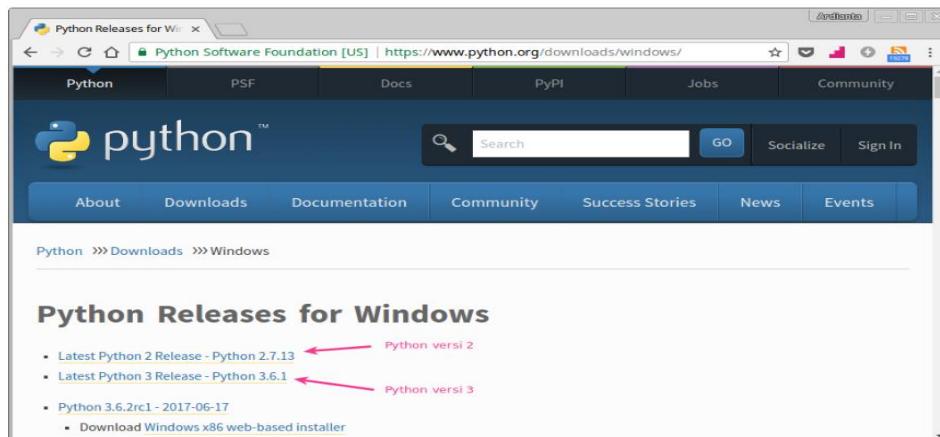
## 4.2 Bahasa Pemograman Yang Digunakan

### 4.2.1 Instalasi Python

Berikut adalah langkah-langkah untuk melakukan instalasi Bahasa pemograman python :

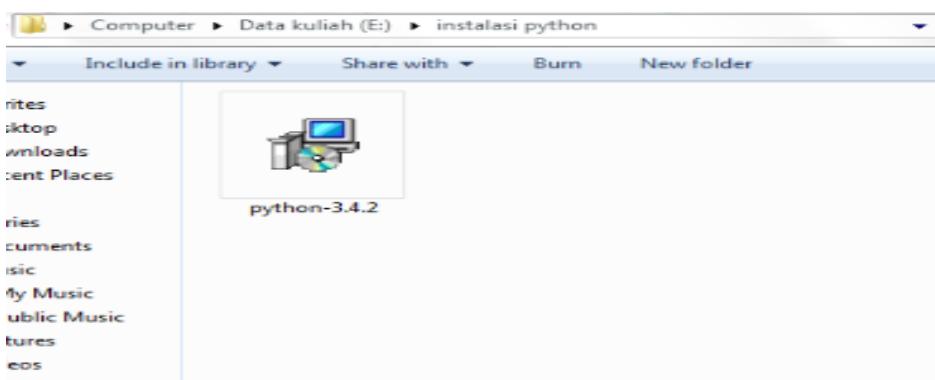
1. Untuk melakukan instalasi python maka download terlebih dahulu file python pada link berikut ini :

<https://www.python.org/downloads/windows/>



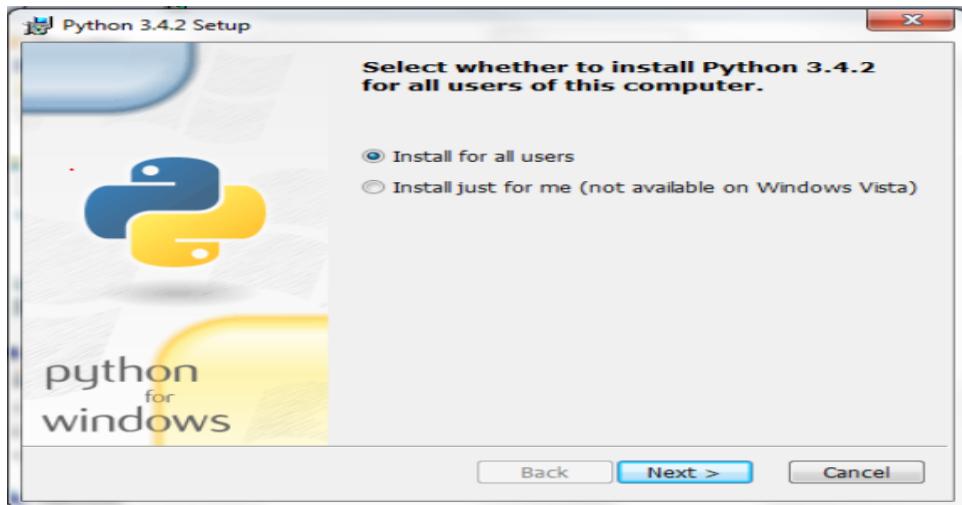
Gambar 4. 18 Downlod File Python

2. Setelah download file sudah selesai maka kita akan mendapatkan file python. File ini akan melakukan instalasi ke sistem operasi computer kita. Lakukan Double klik pada file untuk mengeksekusinya :



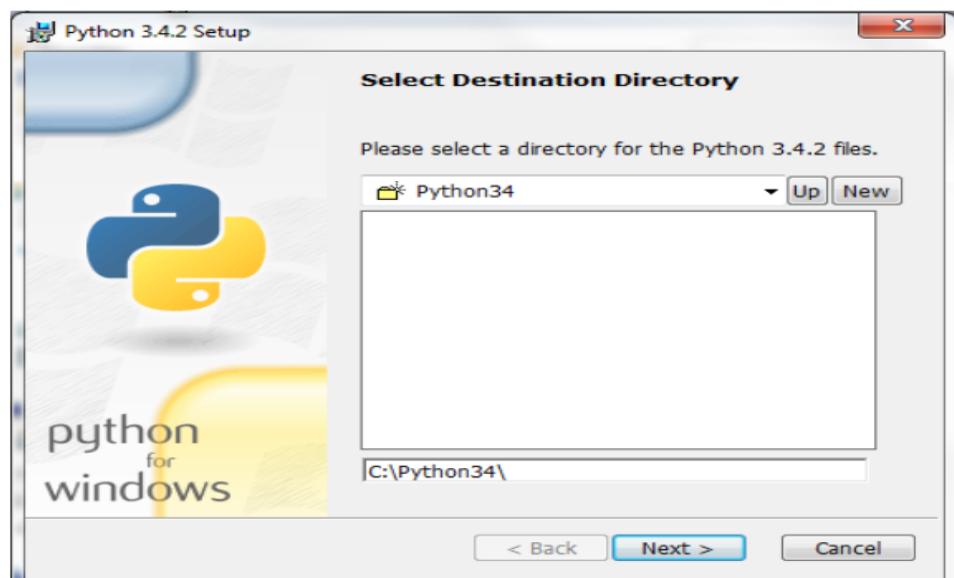
Gambar 4. 19 Eksekusi File Python

3. Kemudian selanjutnya pilih siapa saja yang boleh memakai python.  
Pilih saja “Intsall for all users” agar bias digunakan untuk semua user di computer.



Gambar 4. 20 Install Python

4. Kemudian tentukan lokasi python yang akan diinstall, kemudian next.



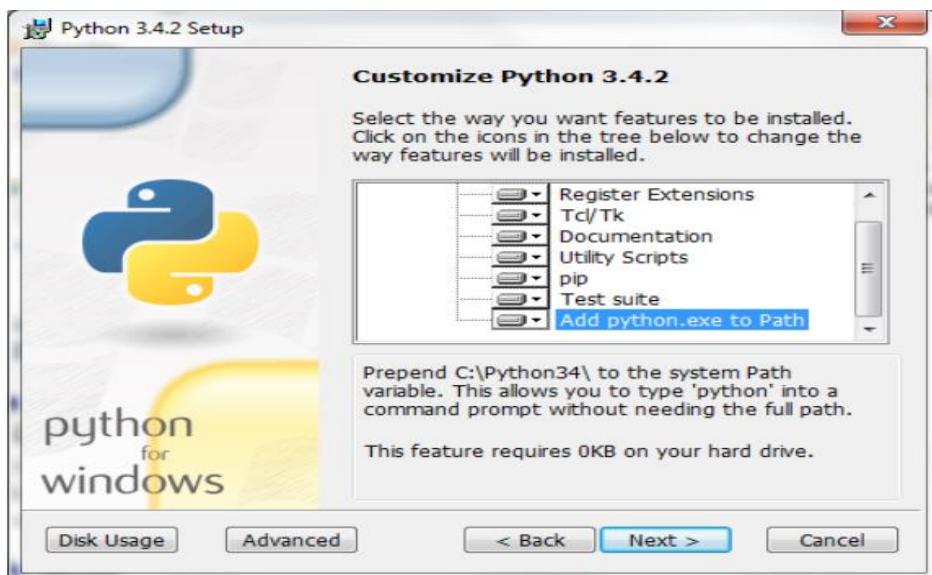
Gambar 4. 21 Tentukan Lokasi Intallasi

5. Pada tahap ini tentukan fitur-fitur yang akan diinstall, aktifkan “Add python.exe to path” agar perintah python dikenali pada CMD (Command Prompt).



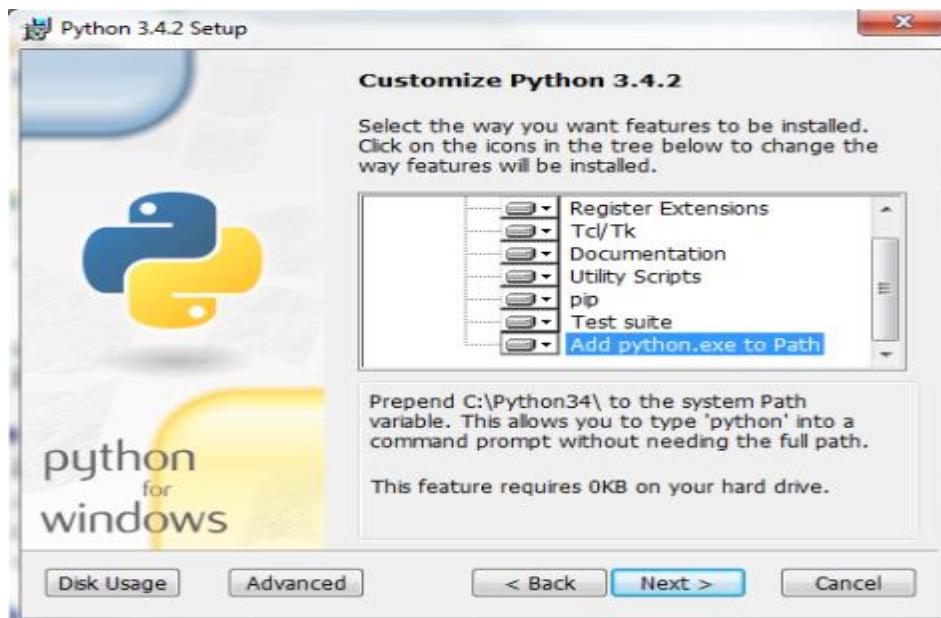
Gambar 4. 22 Aktifkan “Add python.exe to path”

6. Setelah diaktifkan, akan menjadi seperti ini:



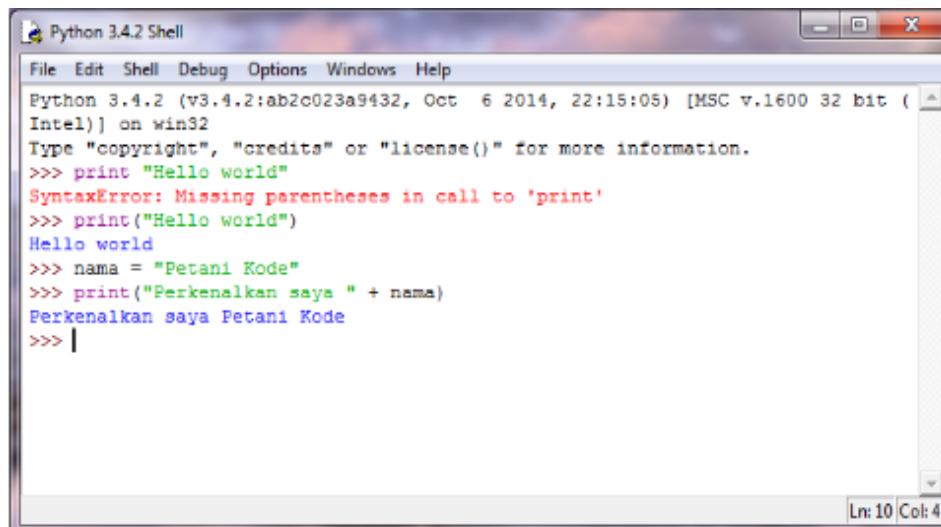
Gambar 4. 23 Hasil Aktivasi

7. Klik finish untuk menyelesaikan



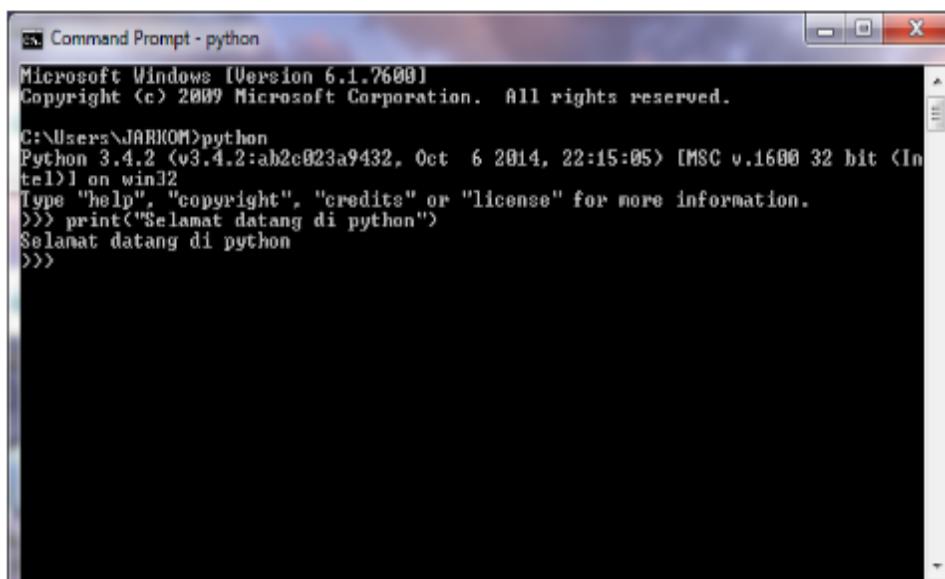
Gambar 4. 24 Instalasi Selesai

8. Untuk melakukn uji coba python Pertama, kita coba dulu membuka Python Shell. Silahkan buka Start Menu kemudian cari Python Shell.



Gambar 4. 25 Melakukan Uji Coba

Kemudian uji coba Python dengan CMD, ketik perintah python untuk masuk ke Python Shell dari CMD.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\JARHOM>python
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:05) [MSC v.1600 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Selamat datang di python")
Selamat datang di python
>>>
```

Gambar 4. 26 Uji Coba Python CMD

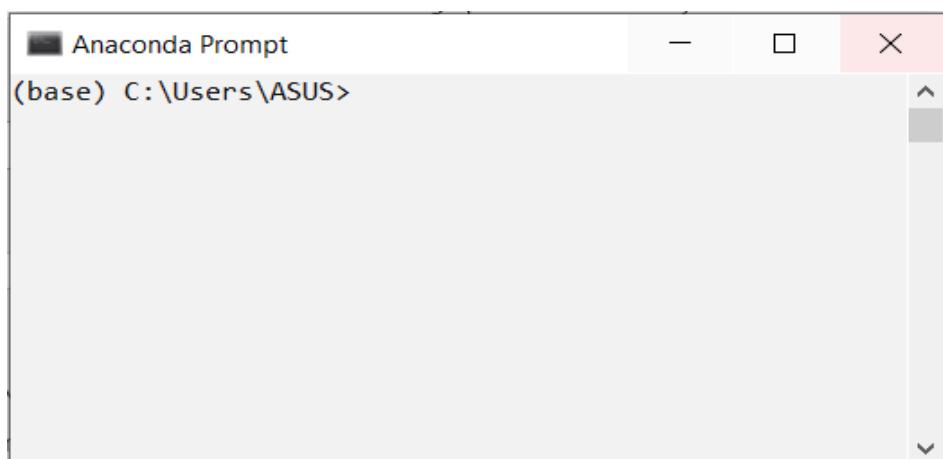
## **BAB V**

### **PENJELASAN, INSTALASI LIBRARY & FRAMEWORK YANG DIGUNAKAN**

#### **5.1 Membuat Environment Pada Anacoda**

Untuk melakukan instalasi framework dan library yang dibutuhkan maka kita harus membuat environment kemudian menginstal library dan framework tersebut ke dalam environment yang sudah dibuat. Hal ini bertujuan untuk memudahkan pengaturan environment sesuai kebutuhan program yang sedang dibuat. Apabila suatu saat nanti kita ingin menggunakan versi yang berbeda dari Tensorflow dan Keras yang telah terinstall tersebut maka hal yang perlu dilakukan hanyalah membuat environment yang baru sehingga tidak akan berdampak kepada environment sebelumnya yang telah diatur spesifik sesuai kebutuhan proyek yang lama. Berikut langkah-langkah untuk membuat environment pada anaconda :

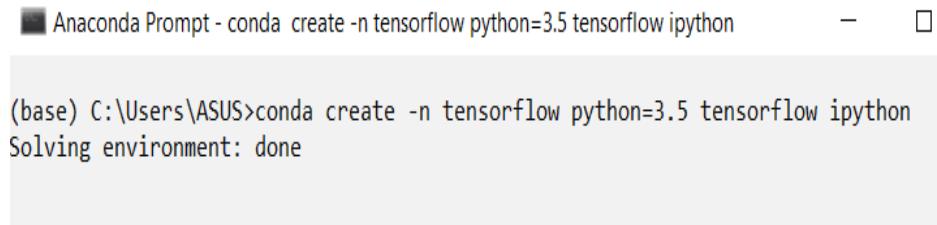
1. Buka anaconda command prompt, kemudian akan tampil anaconda prompt sebagai berikut :



Gambar 5. 1 Membuka Anaconda Commannd Prompt

2. Untuk membuat environment baru dalam Anaconda Anda dapat menggunakan perintah sebagai berikut :

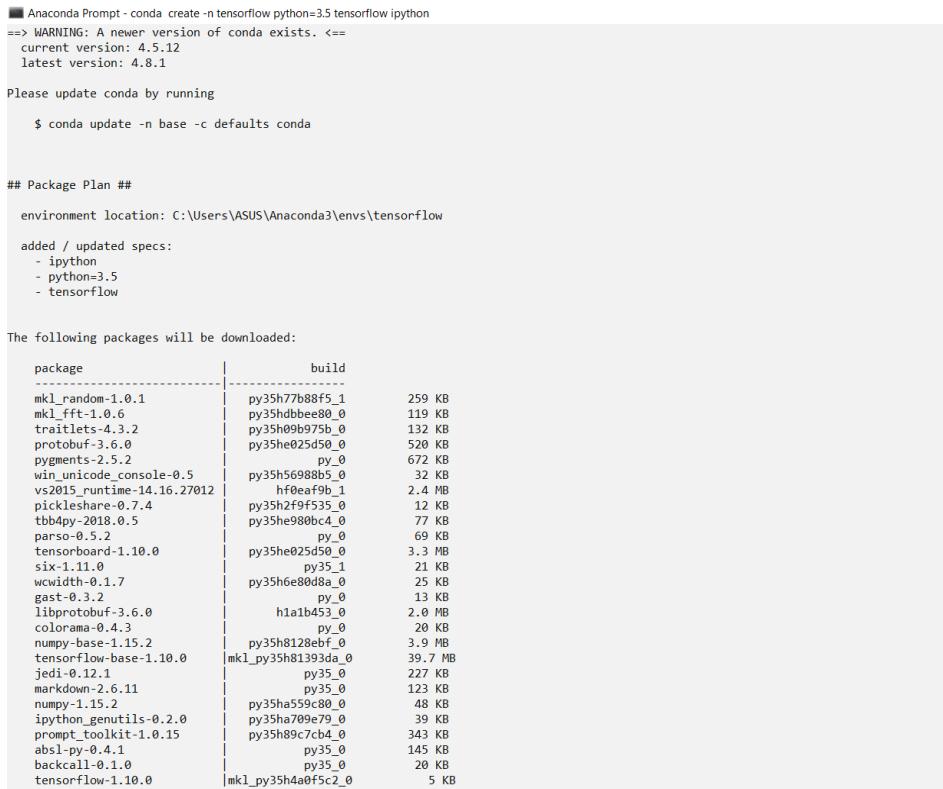
```
conda create -n "nama environment" python=3.5 tensorflow ipython
```



```
(base) C:\Users\ASUS>conda create -n tensorflow python=3.5 tensorflow ipython
Solving environment: done
```

Gambar 5. 2 Membuat environment

3. Tunggu hingga proses pembuatan environment-nya selesai kemudian jalankan perintah y untuk menginstall package yang disediakan anaconda.



```
conda create -n tensorflow python=3.5 tensorflow ipython
==> WARNING: A newer version of conda exists. <==
    current version: 4.5.12
    latest version: 4.8.1

Please update conda by running

$ conda update -n base -c defaults conda

## Package Plan ##

environment location: C:\Users\ASUS\Anaconda3\envs\tensorflow

added / updated specs:
- ipython
- python=3.5
- tensorflow

The following packages will be downloaded:

  package          build
mkl_random-1.0.1      py35h77b88f5_1    259 KB
mkl_fft-1.0.6        py35hdbbee80_0    119 KB
traitlets-4.3.2       py35hb9b975b_0    132 KB
protobuf-3.6.0         py35he025d50_0    520 KB
pygments-2.5.2          py_0                672 KB
win_unicode_console-0.5 py35hb6988bb5_0    32 KB
vs2015_runtime-14.16.27012 hf0eaf9b_1    2.4 MB
pickleshare-0.7.4      py35h2f9f535_0    12 KB
tbb4py-2018.0.5        py35he980bc4_0    77 KB
parso-0.5.2             py_0                69 KB
tensorboard-1.10.0      py35he025d50_0    3.3 MB
six-1.11.0              py35_1                21 KB
wcwidth-0.1.7           py35h680d8a_0    25 KB
gast-0.3.2               py_0                13 KB
libprotobuf-3.6.0        h1a1b453_0    2.0 MB
colorama-0.4.3            py_0                20 KB
numpy-base-1.15.2        py35h8128ebf_0    3.9 MB
tensorflow-base-1.10.0   mkl_py35h81393da_0 39.7 MB
jedi-0.12.1               py35_0                227 KB
markdown-2.6.11            py35_0                123 KB
numpy-1.15.2              py35ha559c80_0    48 KB
ipython_genutils-0.2.0     py35ha709e79_0    39 KB
prompt_toolkit-1.0.15      py35hb89c7cb4_0    343 KB
absl-py-0.4.1               py35_0                145 KB
backcall-0.1.0              py35_0                20 KB
tensorflow-1.10.0          mkl_py35h4a0ff5c2_0   5 KB
```

Gambar 5. 3 Proses Membuat Environment

```
Anaconda Prompt - conda create -n tensorflow python=3.5 tensorflow ipython
  _tfflow_select: 2.3.0-mkl
  astropy: 0.7.1-py35_0
  astor: 0.7.1-py35_0
  backcall: 0.1.0-mklv35_0
  blas: 1.0-mkl
  certifi: 2018.8.24-py35_1
  colorama: 0.4.3-py_0
  decorator: 4.4.2-py_0
  gast: 0.3.2-py_0
  grpcio: 1.12.1-py35h1a1b453_0
  icc_rt: 2019.0.0-h0c432a_1
  intel-openmp: 2019.4-245
  ipython: 0.5.1-py35_0
  ipython_genutils: 0.12.1-py35h709e79_0
  jedi: 2019.0.5_0
  libmklml: 3.6.0-h1a1b453_0
  libprotobuf: 3.6.0-py35hb5f5c2_0
  markdown: 2.0.0-py35_0
  mkl: 2018.0.0
  mkl_fft: 1.0.6-py35hbdbbe80_0
  mkl_random: 1.0.1-py35h7b88ff5_1
  numpy: 1.15.2-py35h5a559c80_0
  numpy-base: 1.15.2-py35h8128ebff_0
  parse: 0.5.2-py35_0
  pickleshare: 0.7.4-py35hf9ff535_0
  pip: 10.0.1-py35_0
  prompt_toolkit: 1.0.15-py35h8997cb4_0
  protobuf: 3.6.1-py35he025d50_0
  pygments: 2.3.2-py35_0
  python: 3.5.6-he925d598_0
  setuptools: 40.2.0-py35_0
  simplegeneric: 0.8.1-py35_2
  six: 1.11.0-py35_1
  tk: 2018.0.0-py35h9793_0
  tbb4py: 2018.0.5-py35he980bc4_0
  tensorboard: 1.10.0-py35he025d50_0
  tensorflow: 1.10.0-mkl_pyy35h4a0ff5c2_0
  tensorflow-base: 1.10.0-mkl_pyy35h81393da_0
  termcolor: 1.1.0-py35_0
  treelib: 4.3.2-py35hb9b975b_0
  vc: 14.1-h9510ff6_4
  vs2015_runtime: 14.16.27012-hf0eaaf9b_1
  wcmwidth: 0.1.7-py35h6e80d8a_0
  wcwidth: 0.1.6-py35_0
  wheel: 0.33.1-py35_0
  win_unicode_console: 0.5-py35h6988bb5_0
  wincertstore: 0.2-py35hfebbdb8_0
  zlib: 1.2.11-h62dc9d7_3

Proceed ([y]/n)?
```

Gambar 5. 4 Proses Membuat Environment

4. Jika pembuatan environment Berhasil maka akan seperti tampilan berikut :

```
Anaconda Prompt
mkl_fft-1.0.6 | 119 KB | ## | 100%
treelib-1.3.2 | 132 KB | ## | 100%
protobuf-3.6.0 | 528 KB | ## | 100%
pygments-2.5.2 | 672 KB | ## | 100%
win_unicode_console | 32 KB | ## | 100%
vs2015_runtime-14.16 | 2.4 MB | ## | 100%
tbb4py-2018.0.5 | 77 KB | ## | 100%
parse-0.5.2 | 69 KB | ## | 100%
tempfile-3.10.0 | 3.3 KB | ## | 100%
six-1.12.0 | 2 KB | ## | 100%
wcwidth-0.1.7 | 25 KB | ## | 100%
gast-0.3.2 | 13 KB | ## | 100%
libprotobuf-3.6.0 | 2.0 MB | ## | 100%
colorama-0.4.3 | 20 KB | ## | 100%
numpy-base-1.15.2 | 3.9 MB | ## | 100% numpy-base-1.15.2 | 3.9 MB | #####
#####
numpy-base-1.15.2 | 3.9 MB | ## | 100% numpy-base-1.15.2 | 3.9 MB | #####
#####
numpy-base-1.15.2 | 3.9 MB | ## | 100% tensorflow-base-1.10.0 | 1.1 MB | #####
#####
tensorflow-base-1.10.0 | 1.1 MB | ## | 100% tensorflow-base-1.10.0 | 1.1 MB | #####
#####
jedi-0.12.1 | 227 KB | ## | 100%
markdown-2.6.11 | 123 KB | ## | 100%
numpy-1.15.2 | 48 KB | ## | 100%
ipython_genutils-0.2 | 39 KB | ## | 100%
prompt_toolkit-1.0.1 | 343 KB | ## | 100%
absl_py-0.4.1 | 145 KB | ## | 100%
backcall-0.1.0 | 20 KB | ## | 100%
tempfile-3.10.0 | 5 KB | ## | 100%
six-1.12.0 | 2 KB | ## | 100%
astor-0.7.1 | 44 KB | ## | 100%
grpcio-1.12.1 | 1.4 MB | ## | 100%
termcolor-1.1.0 | 8 KB | ## | 100%
tbb-2019.4 | 173 KB | ## | 100%
simplegeneric-0.8.1 | 10 KB | ## | 100%
ipython-6.5.0 | 1.1 MB | ## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
# $ conda activate tensorflow
#
# To deactivate an active environment, use
# $ conda deactivate

(base) C:\Users\ASUS>
```

Gambar 5. 5 Instalasi Selesai

5. Aktifkan environment yang dibuat tadi dengan perintah berikut.

```
(base) C:\Users\ASUS>activate tensorflow  
(tensorflow) C:\Users\ASUS>
```

Gambar 5. 6 Aktivasi Environment

6. Setelah environment aktif maka selanjutnya kita hanya perlu melakukan instalasi framework dan library yang dibutuhkan untuk objek deteksi.

### 5.1.1 Framework Tensorflow

TensorFlow merupakan open source framework yang dapat digunakan untuk mengembangkan, melatih, dan menggunakan model deteksi objek. Sistem ini sudah banyak diterapkan pada berbagai produk Google antara lain pencarian image, deteksi wajah dan plat nomor kendaraan pada Google Streetview, Google Assistant, Waymo atau Self Driving Car, dan lain-lain.

Tensorflow bekerja dengan komputasional untuk membuat model machine learning. TensorFlow menyediakan berbagai toolkit yang memungkinkan Anda membuat model pada tingkat abstraksi yang Anda suka. Anda dapat menggunakan API dengan tingkat yang lebih rendah untuk membuat model dengan menentukan serangkaian operasi matematis. Sebagai alternatif, Anda dapat menggunakan API dengan tingkat yang lebih tinggi (seperti tf.estimator ) untuk menentukan arsitektur yang telah ditetapkan, seperti regresi linier atau jaringan neural.

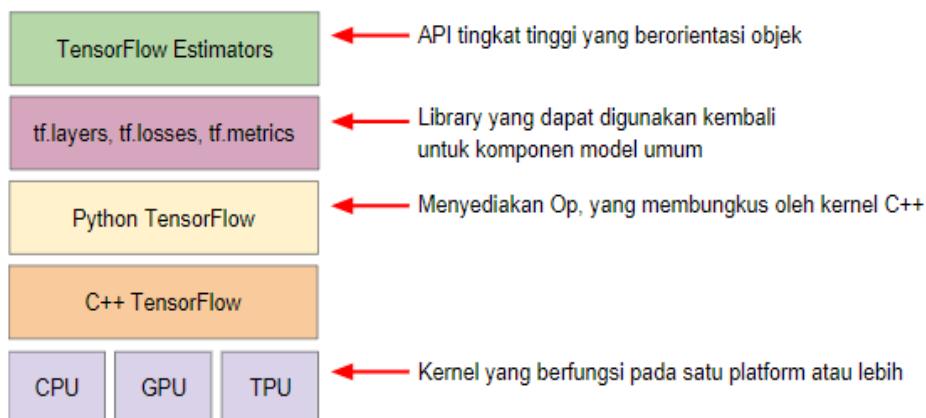
Framework Tensorflow ini digunakan pada proses pembuatan system objek deteksi agar memudahkan implementasi Algoritma dan penggunaan Bahasa pemrograman kemudian terdapat GPU untuk mempercepat proses training. TensorFlow sebagai kerangka machine learning yang dapat

digunakan untuk mengolah banyak data atau ingin mempelajari kecerdasan buatan (artificial intelligence) secara mendalam.

TensorFlow adalah pustaka perangkat lunak sumber terbuka untuk perhitungan numerik berkinerja tinggi. Arsitekturnya yang fleksibel memungkinkan penyebaran komputasi dengan mudah di berbagai platform (CPU, GPU, TPU), dan dari desktop ke cluster server hingga perangkat seluler dan tepi.

Awalnya dikembangkan oleh para peneliti dan insinyur dari tim Google Brain dalam organisasi AI Google, ia hadir dengan dukungan kuat untuk pembelajaran mesin dan pembelajaran mendalam dan inti perhitungan numerik yang fleksibel digunakan di banyak domain ilmiah lainnya.

Gambar berikut menunjukkan hierarki toolkit TensorFlow saat ini :



Gambar 5. 7 Tensorflow

Tabel berikut berisi ringkasan tujuan dari berbagai lapisan:

Tabel 5. 1 Hierarki toolkit TensorFlow

| Toolkit                  | Deskripsi                                   |
|--------------------------|---------------------------------------------|
| Estimator (tf.estimator) | API tingkat tinggi yang berorientasi objek. |

|                                |                                    |
|--------------------------------|------------------------------------|
| tf.layers/tf.losses/tf.metrics | library untuk komponen model umum. |
| TensorFlow                     | API dengan tingkat lebih rendah    |

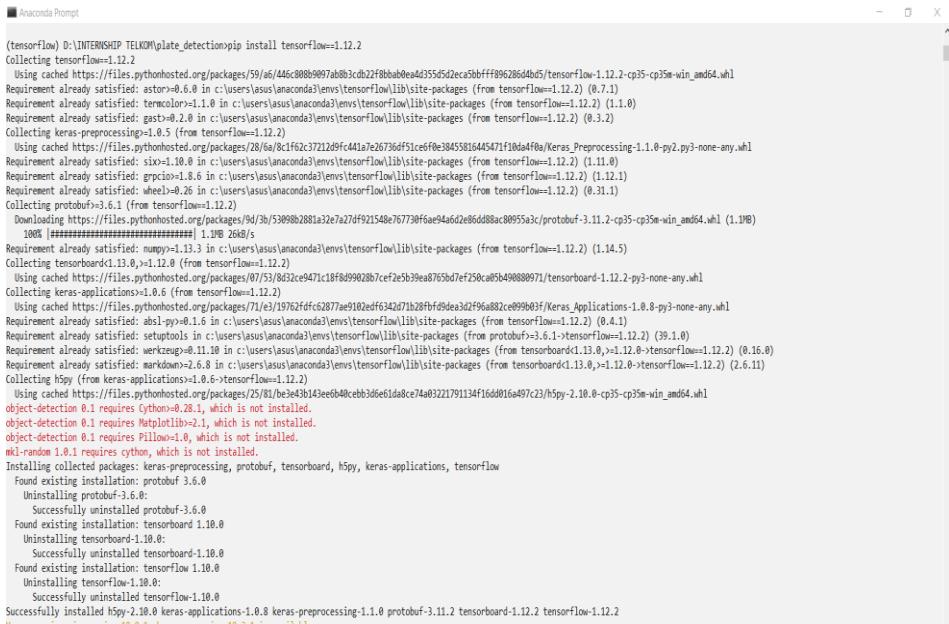
TensorFlow terdiri dari dua komponen berikut:

- buffer protokol grafik
  - waktu proses yang menjalankan grafik (terdistribusi)
- komponen ini masing-masing bersifat analog terhadap kode Python dan penafsir Python. Sama seperti pemrograman Python diterapkan pada beberapa platform hardware untuk menjalankan kode Python, TensorFlow dapat menjalankan grafik pada beberapa platform hardware, termasuk CPU, GPU, dan TPU.

API mana yang harus Anda gunakan? Anda harus menggunakan tingkat abstraksi tertinggi yang dapat memecahkan masalah. Tingkat abstraksi yang lebih tinggi lebih mudah digunakan, tetapi juga (berdasarkan desain) kurang fleksibel. Sebaiknya mulai dengan API tingkat tertinggi terlebih dahulu dan pastikan semuanya berfungsi. Jika Anda memerlukan fleksibilitas tambahan untuk beberapa masalah pemodelan khusus, berpindahlah ke satu tingkat lebih rendah. Perhatikan bahwa setiap tingkat dibuat menggunakan API di tingkat yang lebih rendah, sehingga menurunkan hierarki seharusnya cukup mudah.

Untuk melakukan instalasi tensorflow, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut

“pip install tensorflow” (versi tensorflow menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal tensorflow, dapat dilihat pada gambar 5.8 :



```
(tensorflow) D:\INTERNSHIP TELKOMplate_detection>pip install tensorflow==1.12.2
Collecting tensorflow==1.12.2
  Using cached https://files.pythonhosted.org/packages/59/a6/446c0808997ab0b0cd22f9bbab0e4d355f5d2eca5bffff806286dd4b5/tensorflow-1.12.2-cp35-cp35m-win_amd64.whl
Requirement already satisfied: astor>=0.6.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.7.1)
Requirement already satisfied: tensorboard<1.1.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (1.1.0)
Requirement already satisfied: gast>=0.2.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.3.2)
Collecting keras-preprocessing<1.0.5 (from tensorflow==1.12.2)
  Using cached https://files.pythonhosted.org/packages/59/a6/446c0808997ab0b0cd22f9bbab0e4d355f5d2eca5bffff806286dd4b5/tensorflow-1.12.2-cp35-cp35m-win_amd64.whl
Requirement already satisfied: six>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (1.11.0)
Requirement already satisfied: grpcio<1.8.6 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.31.1)
Collecting protobuf<3.6.1 (from tensorflow==1.12.2)
  Downloading https://files.pythonhosted.org/packages/9d/b3/530d8b2881a32e7a27df921548e76773ff6aae04ad2e06dd88ac80955a3c/protobuf-3.11.2-cp35-cp35m-win_amd64.whl (1.1MB)
  100% [=====] 1.1MB 260B/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (1.14.5)
Collecting tensorboard<1.13.0,>=1.12.2 (from tensorflow==1.12.2)
  Using cached https://files.pythonhosted.org/packages/07/53/8d32ca947c18fb499028b7cef2e5b39e8765bd7ef250ca05b49088971/tensorboard-1.12.2-py3-none-any.whl
Requirement already satisfied: tensorflow<1.12.2 (from tensorflow==1.12.2)
  Using cached https://files.pythonhosted.org/packages/71/e1/19762fd4c52877ae910edff6342d71b28fbf9d9ea3d2f96a882ce099b803f/Keras_Applications-1.8.8-py3-none-any.whl
Requirement already satisfied: absl-py<1.5.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.4.1)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf<3.6.1->1.12.2) (39.1.0)
Requirement already satisfied: werkzeug<0.11.10 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow<1.13.0,>=1.12.0->tensorflow<1.12.2) (0.16.0)
Requirement already satisfied: markdown<2.6.8 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow<1.13.0,>=1.12.0->tensorflow<1.12.2>) (2.6.11)
Requirement already satisfied: h5py (from keras-applications<1.0.6->tensorflow<1.12.2>)
  Using cached https://files.pythonhosted.org/packages/81/be393b433eeb040ceb6d6e61da8c74a0322179134f16d0016a497c23/h5py-2.10.0-cp35-cp35m-win_amd64.whl
object-detection 0.1 requires Cython<0.28.1, which is not installed.
object-detection 0.1 requires Matplotlib<2.1, which is not installed.
object-detection 0.1 requires Pillow<5.0, which is not installed.
mkl-random 1.0.1 requires cython, which is not installed.
Installing collected packages: keras-preprocessing, protobuf, tensorboard, h5py, keras-applications, tensorflow
Found existing installation: protobuf 3.6.0
  Uninstalling protobuf-3.6.0:
    Successfully uninstalled protobuf-3.6.0
Found existing installation: tensorboard 1.10.0
  Uninstalling tensorboard-1.10.0:
    Successfully uninstalled tensorboard-1.10.0
Found existing installation: tensorflow 1.10.0
  Uninstalling tensorflow-1.10.0:
    Successfully uninstalled tensorflow-1.10.0
Successfully uninstalled tensorflow-1.10.0
Successfully installed h5py-2.10.0 keras-applications-1.1.0 protobuf-3.11.2 tensorboard-1.12.2 tensorflow-1.12.2
```

Gambar 5. 8 Instalasi Tensorflow

### 5.1.2 Library Keras

Keras adalah pustaka neural-network open-source yang ditulis dengan Python. Itu mampu berjalan di atas TensorFlow, Microsoft Cognitive Toolkit, R, Theano, atau PlaidML. Dirancang untuk memungkinkan eksperimen cepat dengan jaringan saraf yang dalam, ia berfokus untuk menjadi ramah pengguna, modular, dan dapat dikembangkan. Itu dikembangkan sebagai bagian dari upaya penelitian proyek ONEIROs (Sistem Operasi Robot Cerdas Elektronik Neuro-ujung terbuka), dan penulis dan pengelola utamanya adalah François Chollet, seorang insinyur Google. Chollet juga adalah penulis model jaringan saraf dalam Xception.

Keras berisi banyak implementasi blok bangunan jaringan saraf yang biasa digunakan seperti lapisan, tujuan, fungsi aktivasi, pengoptimal, dan sejumlah alat untuk membuat bekerja dengan data gambar dan teks lebih mudah untuk menyederhanakan pengkodean yang diperlukan untuk menulis kode Jaringan Neural Network. Kode ini dihosting di GitHub, dan

forum dukungan komunitas menyertakan halaman masalah GitHub, dan saluran Slack.

Selain jaringan saraf standar, Keras memiliki dukungan untuk jaringan saraf konvolusional dan berulang. Ini mendukung lapisan utilitas umum lainnya seperti dropout, normalisasi batch, dan pengumpulan. Keras memungkinkan pengguna untuk menghasilkan model yang dalam pada telepon pintar (iOS dan Android), di web, atau di Java Virtual Machine. Hal ini juga memungkinkan penggunaan pelatihan terdistribusi model pembelajaran mendalam tentang kelompok Unit Pemrosesan Grafis (GPU) dan unit pemroses Tensor (TPU) terutama dalam hubungannya dengan CUDA. Untuk melakukan instalasi keras, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install keras” (versi keras menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal keras, dapat dilihat pada gambar 5.9:

```
Anaconda Prompt
Uninstalling tensorflow-1.10.0:
Successfully uninstalled tensorflow-1.10.0
Found existing installation: tensorflow 1.10.0
Uninstalling tensorflow-1.10.0:
Successfully uninstalled tensorflow-1.10.0
Successfully installed h5py-2.10.0 keras-applications-1.0.8 keras-preprocessing-1.1.0 protobuf-3.11.2 tensorboard-1.12.2 tensorflow-1.12.2
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install keras==2.3.1
Collecting keras==2.3.1
  Using cached https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bcd0fe9e589a60ceb/Keras-2.3.1-py2.py3-none-any.whl
Collecting pyyaml (from keras==2.3.1)
  Downloading https://files.pythonhosted.org/packages/8a/53/a649f98cd5ab851dc48677b8ae775113756286d040622381d385afcd5a32/PyYAML-5.3-cp35-cp35m-win_amd64.whl (208kB)
    100% [##] 215kB 30kB/s
Requirement already satisfied: numpy<1.9.1 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.14.5)
Requirement already satisfied: keras-applications<1.0.6 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.0.8)
Collecting scipy<0.14 (from keras==2.3.1)
  Downloading https://files.pythonhosted.org/packages/ef/ae/78bbaf498bba92e5ce5903b096b75b5e1f9f82a742fc37a6595892f1ffca/scipy-1.4.1-cp35-cp35m-win_amd64.whl (30.8MB)
    100% [##] 30.8MB 44kB/s
Requirement already satisfied: keras-preprocessing<1.0.5 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.1.0)
Requirement already satisfied: six<1.9.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.11.0)
Requirement already satisfied: h5py in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (2.10.0)
object-detection 0.1 requires Cython<0.28.1, which is not installed.
object-detection 0.1 requires Matplotlib>2.1, which is not installed.
object-detection 0.1 requires Pillow<1.0, which is not installed.
mkl-random 1.0.1 requires cython, which is not installed.
Installing collected packages: pyyaml, scipy, keras
Successfully installed keras-2.3.1 pyyaml-5.3 scipy-1.4.1
```

Gambar 5. 9 Instalasi Keras

### **5.1.3 Library Open-CV**

OpenCV (Open source computer vision) adalah library fungsi pemrograman yang terutama ditujukan untuk computer vision. Awalnya dikembangkan oleh Intel, kemudian didukung oleh Willow Garage kemudian Itseez (yang kemudian diakuisisi oleh Intel). Perpustakaan adalah lintas-platform dan gratis untuk digunakan di bawah lisensi BSD open-source. OpenCV mendukung kerangka pembelajaran yang dalam TensorFlow, Torch / PyTorch dan Caffe. OpenCV digunakan dalam objek deteksi sebagai library yang ditujukan untuk mengolah citra dinamis secara real-time.

Resmi diluncurkan pada tahun 1999, proyek OpenCV pada awalnya merupakan inisiatif Intel Research untuk memajukan aplikasi intensif CPU, bagian dari serangkaian proyek termasuk penelusuran sinar waktu nyata dan dinding tampilan 3D. Kontributor utama untuk proyek ini termasuk sejumlah pakar optimisasi di Intel Rusia, serta Tim Perpustakaan Kinerja Intel. Pada hari-hari awal OpenCV, tujuan proyek digambarkan sebagai:

- Memajukan penelitian visi dengan menyediakan tidak hanya kode terbuka tetapi juga dioptimalkan untuk infrastruktur visi dasar. Tidak ada lagi menciptakan kembali roda.
- Menyebarluaskan pengetahuan visi dengan menyediakan infrastruktur umum yang dapat dibangun oleh pengembang, sehingga kode akan lebih mudah dibaca dan dapat ditransfer.
- Aplikasi komersial berbasis visi maju dengan membuat kode portabel, dioptimalkan kinerja tersedia secara gratis - dengan lisensi yang tidak memerlukan kode untuk terbuka atau bebas sendiri.

Versi alpha pertama dari OpenCV dirilis ke publik di Konferensi IEEE tentang Visi Komputer dan Pengenalan Pola pada tahun 2000, dan lima beta dirilis antara tahun 2001 dan 2005. Versi 1.0 pertama dirilis pada tahun 2006. Versi 1.1 "pra-rilis" dirilis pada Oktober 2008.

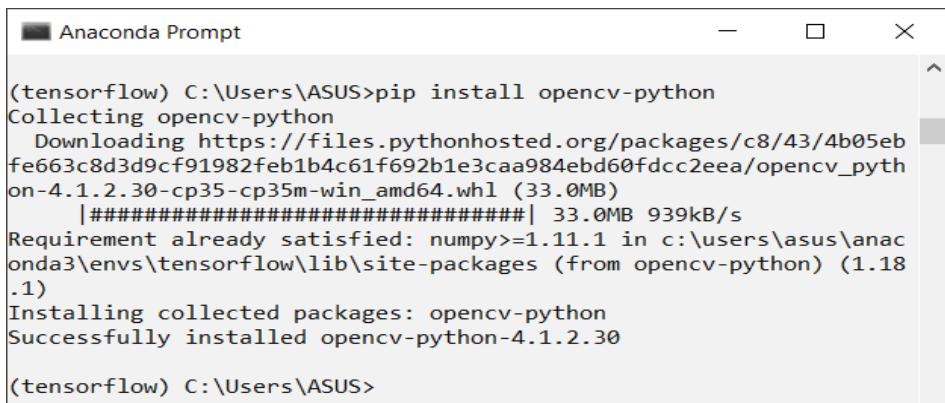
Rilis utama kedua dari OpenCV adalah pada Oktober 2009. OpenCV 2 mencakup perubahan besar pada antarmuka C++, yang bertujuan lebih mudah, pola yang lebih aman, fungsi baru, dan implementasi yang lebih baik untuk yang sudah ada dalam hal kinerja (terutama pada multi sistem inti). Rilis resmi sekarang terjadi setiap enam bulan dan pengembangan sekarang dilakukan oleh tim Rusia independen yang didukung oleh perusahaan komersial.

Pada Agustus 2012, dukungan untuk OpenCV diambil alih oleh yayasan nirlaba OpenCV.org, yang mengelola pengembang dan situs pengguna. Pada Mei 2016, Intel menandatangani perjanjian untuk mengakuisisi Itseez, pengembang OpenCV terkemuka.

OpenCV ditulis dalam C++ dan antarmuka utamanya adalah dalam C++, tetapi ia masih mempertahankan antarmuka C yang lebih lama dan lebih komprehensif. Ada binding di Python, Java dan MATLAB / OCTAVE. API untuk antarmuka ini dapat ditemukan dalam dokumentasi online. Wrappers dalam bahasa lain seperti C#, Perl, Ch, Haskell, dan Ruby telah dikembangkan untuk mendorong adopsi oleh khalayak yang lebih luas.

Sejak versi 3.4, OpenCV.js adalah pengikatan JavaScript untuk subset fungsi OpenCV yang dipilih untuk platform web. Semua pengembangan dan algoritma baru di OpenCV sekarang dikembangkan di antarmuka C++. Untuk melakukan instalasi OpenCV, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah

berikut “pip install opencv-python”. Berikut tampilan cmd apabila berhasil menginstal OpenCV, dapat dilihat pada gambar 5.10 :



```
Anaconda Prompt
(tensorflow) C:\Users\ASUS>pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/c8/43/4b05ebf663c8d3d9cf91982feb1b4c61f692b1e3caa984ebd60fdcc2eea/opencv_python-4.1.2.30-cp35-cp35m-win_amd64.whl (33.0MB)
    #####| 33.0MB 939kB/s
Requirement already satisfied: numpy>=1.11.1 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from opencv-python) (1.18.1)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.1.2.30
(tensorflow) C:\Users\ASUS>
```

Gambar 5. 10 Instalasi OpenVC

#### 5.1.4 Library Pandas

Pandas merupakan library berlisensi open source dan BSD yang menyediakan struktur data dan analisis data yang mudah digunakan dan berkinerja tinggi untuk bahasa pemrograman Python. Dengan penggunaan sistem dataframe, kita dapat memuat sebuah file ke dalam tabel virtual ala spreadsheet dengan menggunakan Pandas. Dengan menggunakan Pandas, kamu dapat mengolah suatu data dan mengolahnya seperti join, distinct, group by, agregasi, dan teknik seperti pada SQL. Hanya saja dilakukan pada tabel yang dimuat dari file ke RAM.

Pandas juga disebut sebagai library analisis data yang memiliki struktur data yang diperlukan untuk membersihkan data mentah ke dalam sebuah bentuk yang cocok untuk analisis (yaitu tabel). Pandas berfungsi untuk melakukan tugas penting seperti menyelaraskan data untuk perbandingan dan penggabungan set data, penanganan data yang hilang, dll, itu telah menjadi sebuah librari de facto untuk pemrosesan data tingkat tinggi dalam Python (yaitu statistik). Pandas didesain untuk menangani

data finansial, dikarenakan alternatif umum adalah menggunakan spreadsheet (misalnya Microsoft Excel).

Struktur dasar data pada pandas dinamakan DataFrame, yaitu sebuah koleksi kolom berurutan dengan nama dan jenis, dengan demikian merupakan sebuah tabel yang tampak seperti database dimana sebuah baris tunggal mewakili sebuah contoh tunggal dan kolom mewakili atribut tertentu. Harus dicatat di sini bahwa elemen dalam berbagai kolom mungkin berapa jenis yang berbeda.

Fitur dataframe yang terdapat pada pandas memudahkan untuk membaca sebuah file dan menjadikannya table, kita juga dapat mengolah suatu data dengan menggunakan operasi seperti join, distinct, group by, agregasi, dan teknik lainnya yang terdapat pada SQL. Berikut beberapa format file yang dapat dibaca menggunakan Pandas, seperti file .txt, .csv, tsv dan lainnya. Untuk melakukan instalasi pandas, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install pandas” (versi pandas menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal pandas, dapat dilihat pada gambar 5.11 :

```
■ Anaconda Prompt
Requirement already satisfied: numpy>=1.9.1 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.14.5)
Requirement already satisfied: keras-applications>1.0.6 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.0.8)
Collecting scipy>0.14 (from keras==2.3.1)
  Downloading https://files.pythonhosted.org/packages/e1/ae/78bba498bb492e5ce5903b096b75b5e1f9f82a742fc37a6595892ff1ffca/scipy-1.4.1-cp35-cp35m-win_amd64.whl (30.8MB)
    100% [##] 30.8MB 44KB/s
Requirement already satisfied: keras-preprocessing>1.0.5 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.1.0)
Requirement already satisfied: six>1.9.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (1.11.0)
Requirement already satisfied: h5py in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras==2.3.1) (2.10.0)
object-detection 0.1 requires python>0.28.1, which is not installed.
object-detection 0.1 requires Matplotlib<2.1, which is not installed.
object-detection 0.1 requires numpy>1.13.1, which is not installed.
mkl-random 1.0.1 requires cython, which is not installed.
Installing collected packages: pyyaml, scipy, keras
Successfully installed keras-2.3.1 pyyaml-5.3 scipy-1.4.1
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install pandas==0.22.0
Collecting pandas==0.22.0
  Downloading https://files.pythonhosted.org/packages/79/59/06c496c847724d00864a9f6e7c11c98561d8901a9853ff7c758c3d2842c68/pandas-0.22.0-cp35-cp35m-win_amd64.whl (9.0MB)
    100% [##] 9.0MB 3.3MB/s
Requirement already satisfied: numpy>=1.15.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from pandas==0.22.0) (1.14.5)
Collecting python-dateutil>2 (from pandas==0.22.0)
  Downloading https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae890709de0b306af2be5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any.whl (227kB)
    100% [##] 235kB 3.3MB/s
Collecting pytz>=2018.4 (from pandas==0.22.0)
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a8788e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl (509kB)
    100% [##] 512kB 2.2MB/s
Requirement already satisfied: six>1.5 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from python-dateutil>2->pandas==0.22.0) (1.11.0)
mkl-random 1.0.1 requires cython, which is not installed.
object-detection 0.1 requires Python>0.28.1, which is not installed.
object-detection 0.1 requires Matplotlib<2.1, which is not installed.
object-detection 0.1 requires numpy>1.13.1, which is not installed.
Installing collected packages: python-dateutil, pytz, pandas
Successfully installed pandas-0.22.0 python-dateutil-2.8.1 pytz-2019.3
```

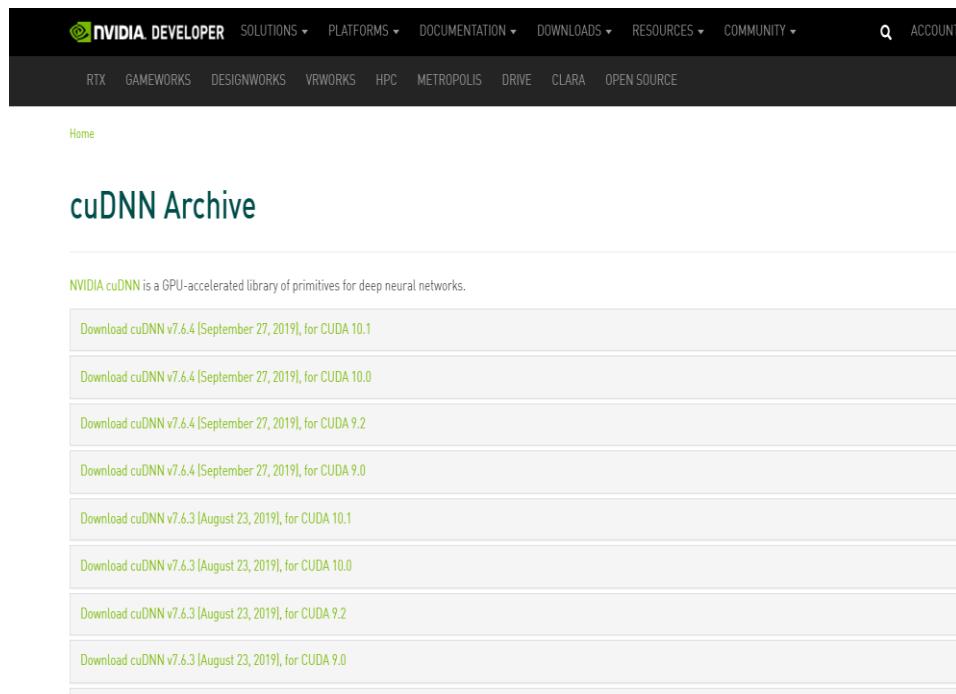
Gambar 5. 11 Instalasi Pandas

### 5.1.5 Library cudnn

CuDNN adalah perpustakaan yang membantu mempercepat kerangka kerja pembelajaran yang dalam, seperti TensoeFlow atau Theano. Pustaka Deep Natural Network NVDIA CUDA (cuDNN) adalah pustaka primitif yang diperuntukkan bagi DNN untuk GPU. CuDNN menyediakan implementasi yang sangat disesuaikan untuk rutin standar seperti konvolusi maju dan kata kunci, pooling, normalisasi, dan lapisan aktivasi. CuDNN adalah bagian dari SDK pembelajaran NVIDIA.

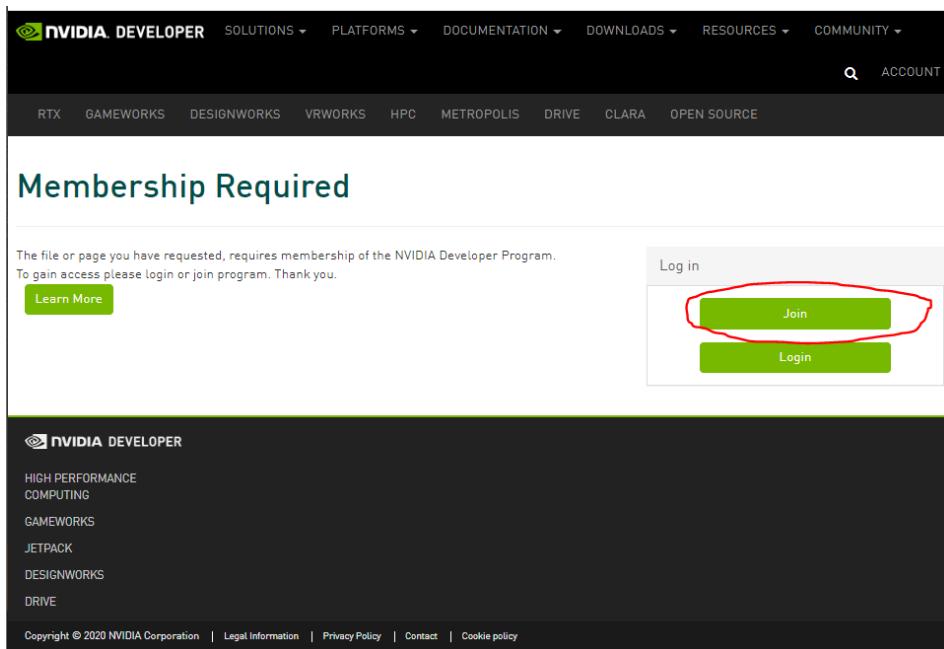
Untuk instalasi cudnn menyesuaikan dengan versi CUDA yang digunakan. Untuk mendapatkan cudnn maka kunjungi link berikut <https://developer.nvidia.com/rdp/cudnn-archive> kemudian sesuaikan versi cudnn yang akan di downlod dengan CUDA yang di install di computer.

1. Berikut tampilan website untuk mendownload cudnn



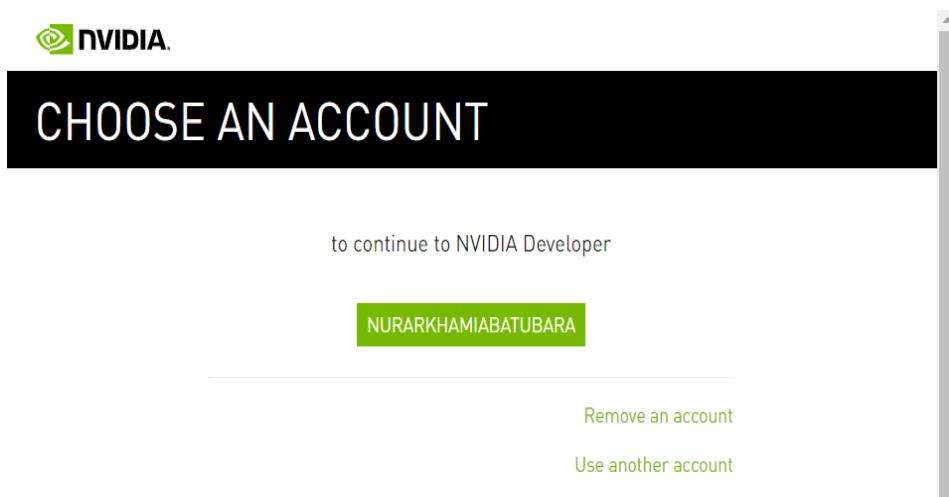
Gambar 5. 12 Downlod Cudnn

2. Saat ingin mendownload cudnn anda harus join dan memiliki akun membership agar bisa login dan mendownload file.



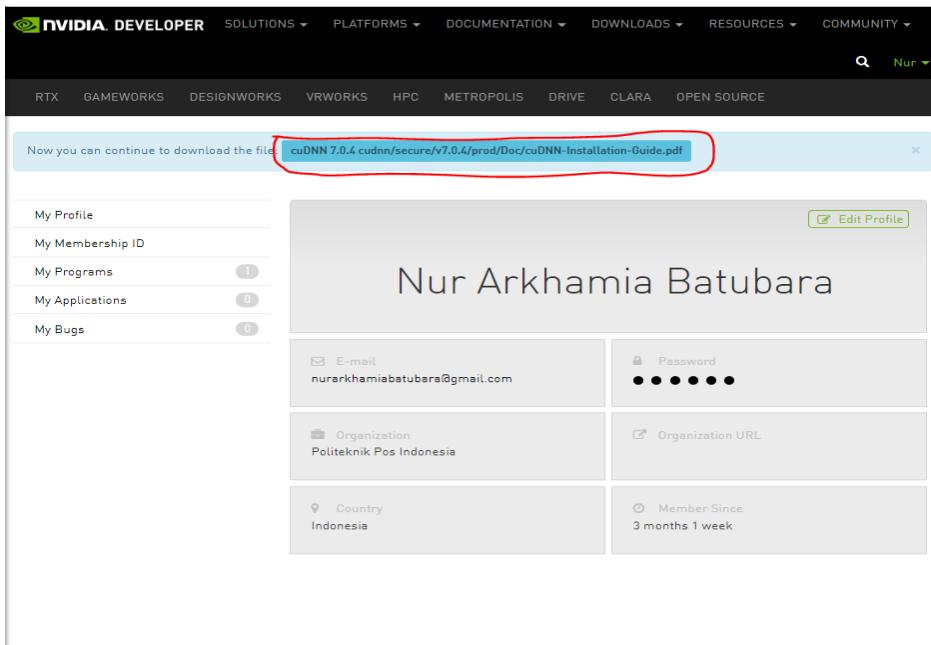
Gambar 5. 13 Daftar Membership

3. Login sebagai email yang didaftarkan.



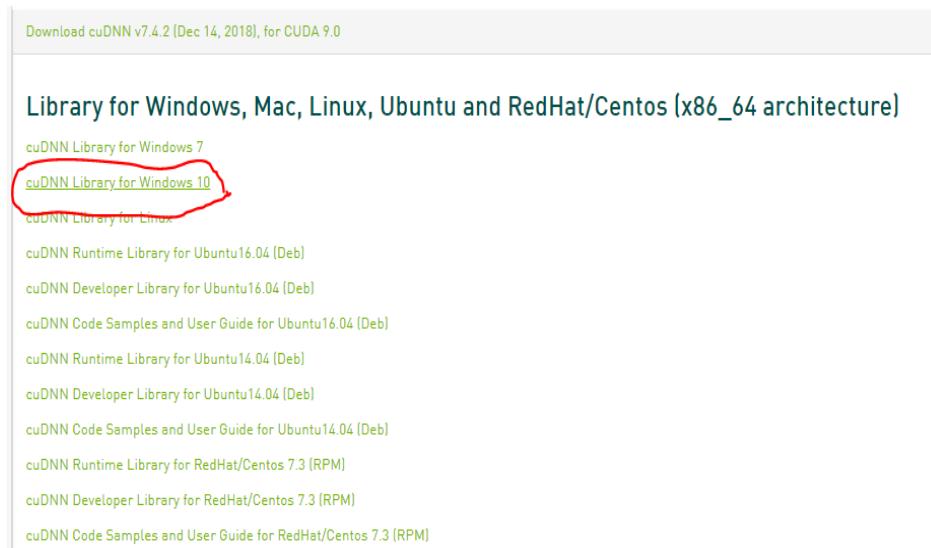
Gambar 5. 14 Login Membership

4. Setelah sudah login maka kembali ke halaman awal untuk memilih cudnn yang sesuai dengan versi CUDA :



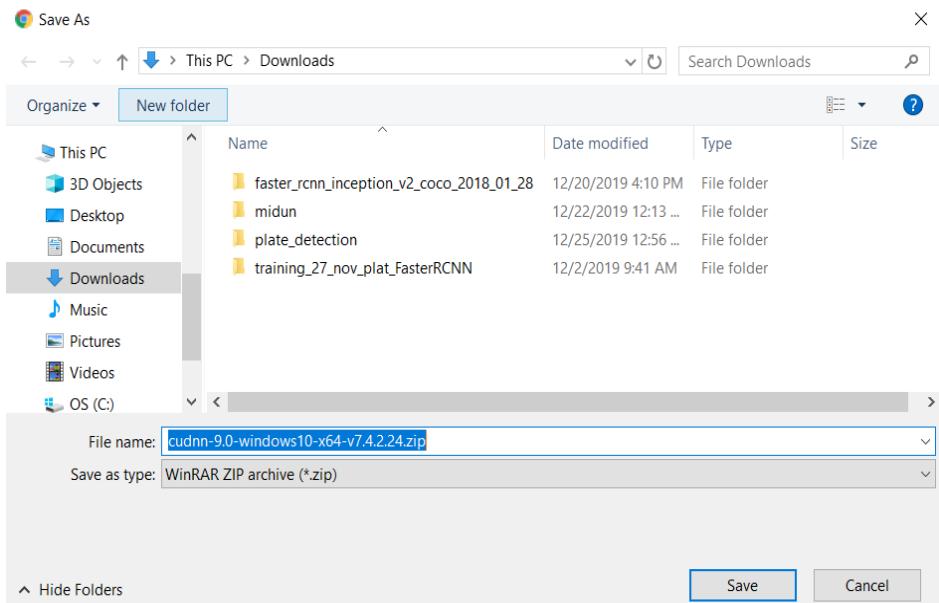
Gambar 5. 15 Selesai Login

5. Saya memilih cudnn dengan versi berikut :



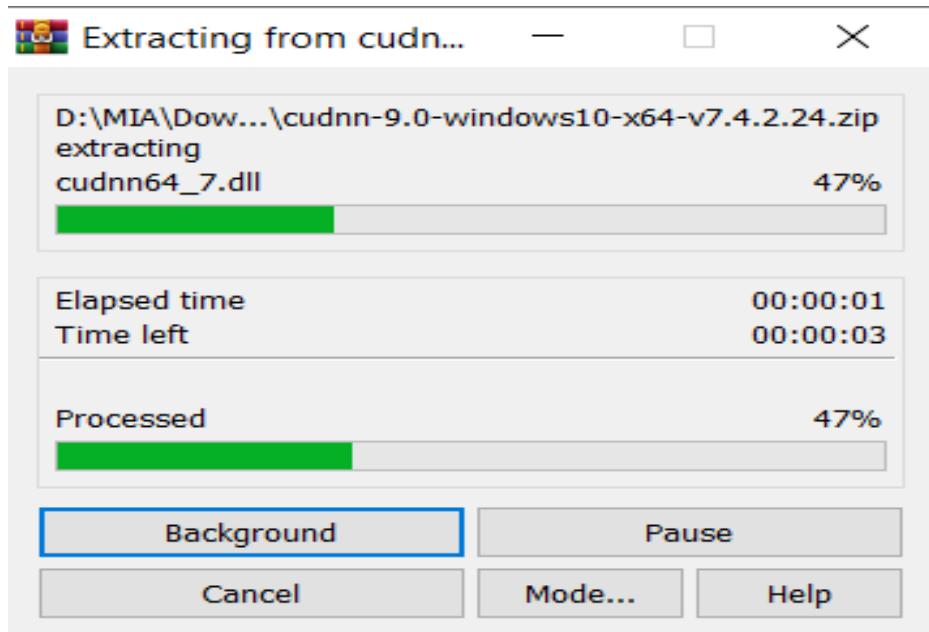
Gambar 5. 16 Memilih versi cudnn

6. Pilih lokasi penyimpanan file :



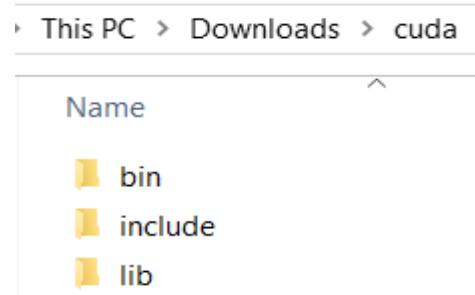
Gambar 5. 17 Pilih Penyimpanan File

7. Melakukan extracting file .zip



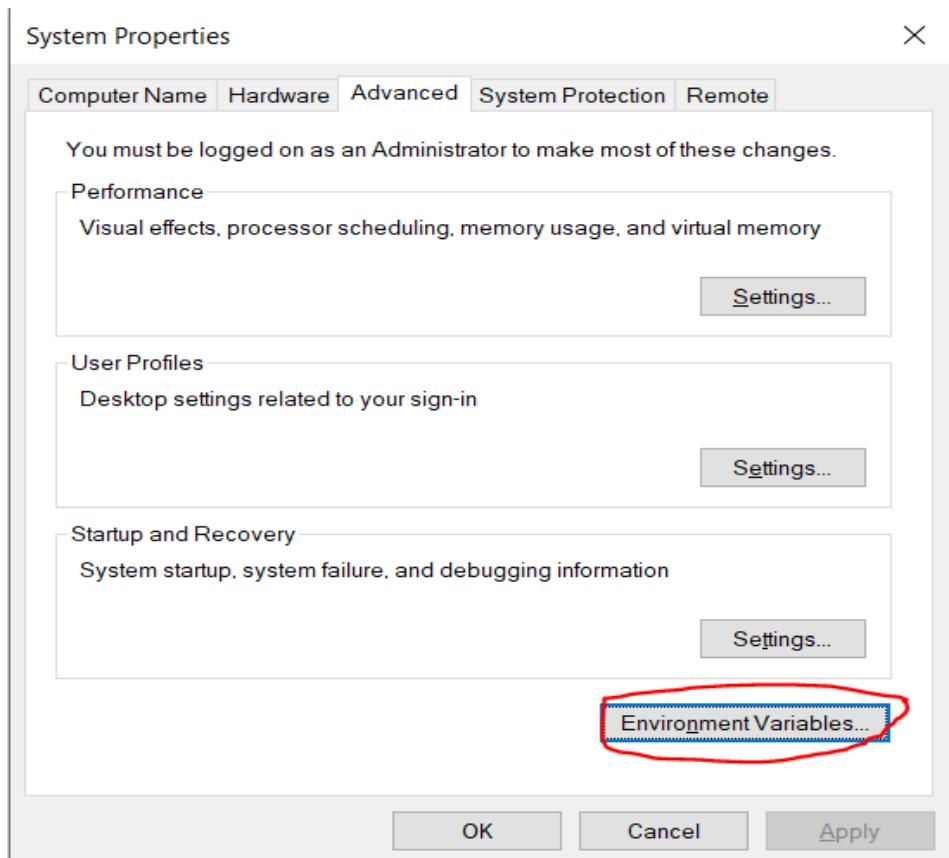
Gambar 5. 18 EXtract File .zip

8. Berikut file yang terdapat pada folder cudnn yang di download :



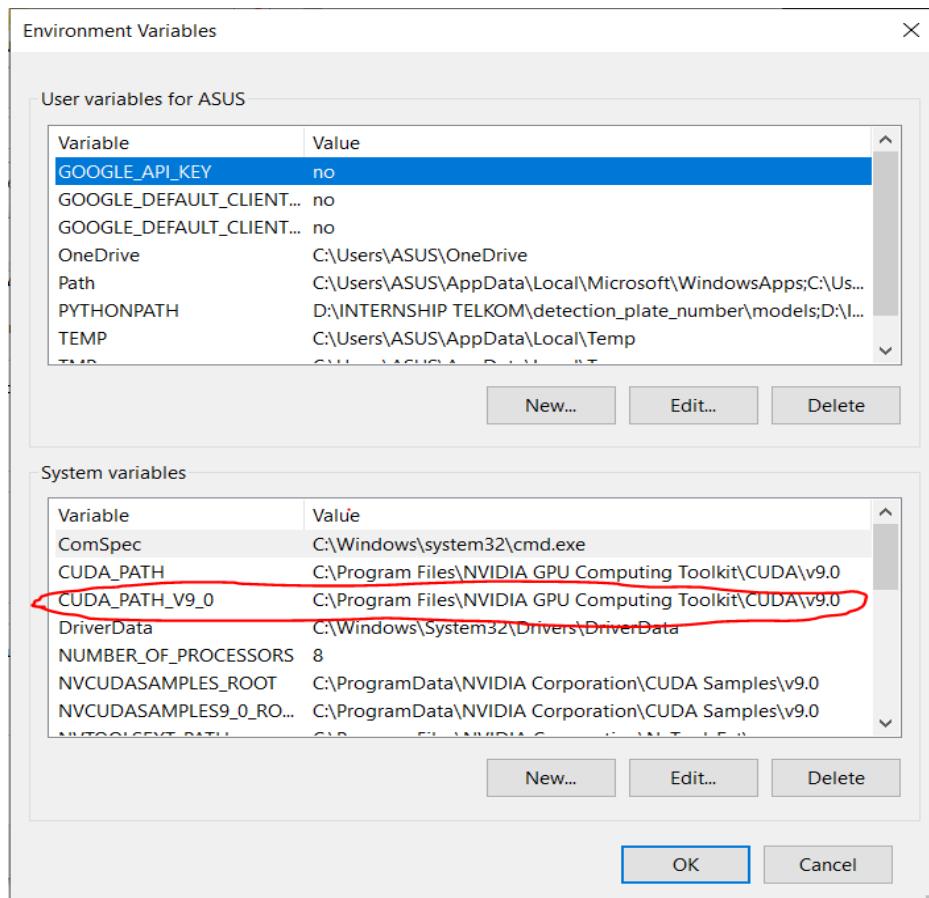
Gambar 5. 19 Fili cudnn

9. Selanjutnya buka path instalasi CUDA dengan melakukan klik environment variabels :



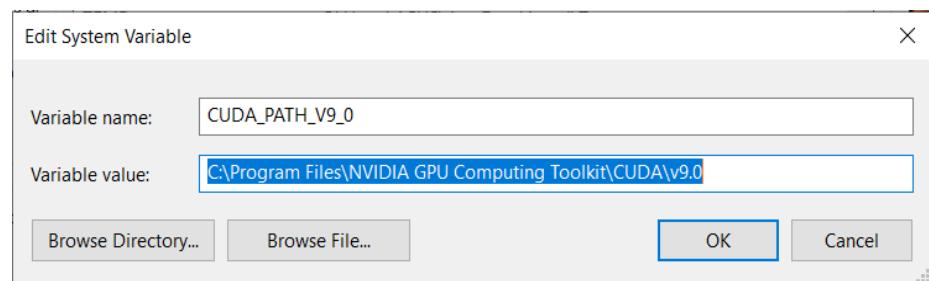
Gambar 5. 20 Membuka Environment

10. Lakukan double klik pada path cuda untuk mendapatkan lokasi penyimpanan file :



Gambar 5. 21 Membuka Path CUDA

11. Copy direktori berikut :



Gambar 5. 22 Copy Link

12. Berikut direktori CUDA yang sesuai dengan path :

| Name                           | Date modified        | Type          | Size  |
|--------------------------------|----------------------|---------------|-------|
| bin                            | 11/7/2019 10:28 A... | File folder   |       |
| doc                            | 11/7/2019 10:25 A... | File folder   |       |
| extras                         | 11/7/2019 10:25 A... | File folder   |       |
| include                        | 11/7/2019 10:28 A... | File folder   |       |
| jre                            | 11/7/2019 10:24 A... | File folder   |       |
| lib                            | 11/7/2019 10:24 A... | File folder   |       |
| libnvvp                        | 11/7/2019 10:25 A... | File folder   |       |
| nvml                           | 11/7/2019 10:25 A... | File folder   |       |
| nvvm                           | 11/7/2019 10:24 A... | File folder   |       |
| src                            | 11/7/2019 10:25 A... | File folder   |       |
| tools                          | 11/7/2019 10:25 A... | File folder   |       |
| CUDA_Toolkit_Release_Notes.txt | 9/2/2017 8:45 PM     | Text Document | 27 KB |
| EULA.txt                       | 9/2/2017 8:45 PM     | Text Document | 82 KB |
| version.txt                    | 9/2/2017 8:46 PM     | Text Document | 1 KB  |

Gambar 5. 23 Replace File

Selanjutnya replace file bin, include, lib dengan file bin, include, lib dari hasil download cudnn tadi, jika sudah maka cudnn sudah terpasang pada CUDA.

### 5.1.6 Library numpy

NumPy merupakan library pada Python yang berfungsi untuk melakukan operasi vektor dan matriks dengan mengolah array dan array multidimensi. Biasanya NumPy digunakan untuk kebutuhan dalam menganalisis data. NumPy adalah paket dasar untuk komputasi ilmiah dengan Python. Ini berisi antara lain:

- objek array N-dimensi yang kuat
- fungsi canggih (penyiaran)
- alat yang berfungsi untuk mengintegrasikan kode C / C ++ dan Fortran
- aljabar linier yang berguna, transformasi Fourier, dan kemampuan angka acak

Selain penggunaan ilmiahnya yang jelas, NumPy juga dapat digunakan sebagai wadah data generik multi dimensi yang efisien. Tipe data sewenang-wenang dapat didefinisikan. Ini memungkinkan NumPy untuk berintegrasi dengan cepat dan cepat dengan berbagai macam basis data. NumPy dilisensikan di bawah lisensi BSD, memungkinkan penggunaan kembali dengan beberapa batasan.

NumPy adalah perpustakaan untuk bahasa pemrograman Python, menambahkan dukungan untuk array dan matriks multi-dimensi yang besar, bersama dengan sejumlah besar fungsi matematika tingkat tinggi untuk beroperasi pada array ini. Nenek moyang NumPy, Numeric, pada awalnya diciptakan oleh Jim Hugunin dengan kontribusi dari beberapa pengembang lainnya. Pada tahun 2005, Travis Oliphant menciptakan NumPy dengan memasukkan fitur-fitur dari Numarray yang bersaing ke Numeric, dengan modifikasi ekstensif. NumPy adalah perangkat lunak sumber terbuka dan memiliki banyak kontributor.

## 1. Sejarah

Bahasa pemrograman Python pada awalnya tidak dirancang untuk komputasi numerik, tetapi menarik perhatian komunitas ilmiah dan teknik sejak awal, sehingga kelompok minat khusus yang disebut matrix-sig didirikan pada tahun 1995 dengan tujuan mendefinisikan paket komputasi array. Di antara anggotanya adalah desainer dan pengelola Python, Guido van Rossum, yang mengimplementasikan ekstensi ke sintaksis Python (khususnya sintaks pengindeksan) untuk membuat komputasi array lebih mudah.

Implementasi paket matriks diselesaikan oleh Jim Fulton, kemudian digeneralisasi oleh Jim Hugunin menjadi Numeric, juga dengan berbagai cara disebut Numerical Python extensions atau NumPy. Hugunin, seorang

mahasiswa pascasarjana di Massachusetts Institute of Technology (MIT), bergabung dengan Corporation for National Research Initiatives (CNRI) untuk bekerja pada JPython pada tahun 1997 meninggalkan Paul Dubois dari Lawrence Livermore Laboratorium Nasional (LLNL) untuk mengambil alih sebagai pengelola. Kontributor awal lainnya termasuk David Ascher, Konrad Hinsen dan Travis Oliphant

Paket baru yang disebut Numarray ditulis sebagai pengganti yang lebih fleksibel untuk Numeric. Seperti Numeric, sekarang sudah usang. Numarray memiliki operasi lebih cepat untuk array besar, tetapi lebih lambat daripada Numeric pada array kecil, jadi untuk sementara waktu kedua paket digunakan untuk kasus penggunaan yang berbeda. Versi terakhir Numeric v24.2 dirilis pada 11 November 2005 dan numarray v1.5.2 dirilis pada 24 Agustus 2006. Ada keinginan untuk memasukkan Numeric ke dalam pustaka standar Python, tetapi Guido van Rossum memutuskan bahwa kode itu tidak dapat dipertahankan dalam keadaan saat itu.

Pada awal 2005, pengembang NumPy Travis Oliphant ingin menyatukan komunitas di sekitar satu paket array dan mem-porting fitur Numarray ke Numeric, merilis hasilnya sebagai NumPy 1.0 pada tahun 2006. Proyek baru ini adalah bagian dari SciPy Untuk menghindari menginstal paket SciPy besar hanya untuk mendapatkan objek array, paket baru ini dipisahkan dan disebut NumPy. Dukungan untuk Python 3 ditambahkan pada 2011 dengan NumPy versi 1.5.0. pada tahun 2011, PyPy memulai pengembangan implementasi API NumPy untuk PyPy. Ini belum sepenuhnya kompatibel dengan NumPy.

## 2. Fitur

NumPy menargetkan implementasi referensi CPython dari Python, yang merupakan penerjemah bytecode yang tidak mengoptimalkan. Algoritma matematika yang ditulis untuk versi Python ini sering berjalan jauh lebih lambat daripada yang dikompilasi setara. NumPy mengatasi masalah kelambatan sebagian dengan menyediakan array multidimensi dan fungsi dan operator yang beroperasi secara efisien pada array, membutuhkan penulisan ulang beberapa kode, sebagian besar loop internal menggunakan NumPy.

Numpy memiliki kegunaan untuk operasi vektor dan matriks. Fiturnya hampir sama dengan MATLAB dalam mengelola array dan array multidimensi. Numpy merupakan salah satu library yang digunakan oleh library lain seperti Scikit-Learn untuk keperluan analisis data.

Menggunakan NumPy dalam Python memberikan fungsionalitas yang sebanding dengan MATLAB karena keduanya diinterpretasikan, dan keduanya memungkinkan pengguna untuk menulis program yang cepat selama sebagian besar operasi bekerja pada array atau matriks, bukannya skalar. Sebagai perbandingan, MATLAB menawarkan sejumlah besar kotak alat tambahan, terutama Simulink, sedangkan NumPy secara intrinsik terintegrasi dengan Python, bahasa pemrograman yang lebih modern dan lengkap. Selain itu, paket Python komplementer tersedia; SciPy adalah pustaka yang menambahkan lebih banyak fungsi seperti MATLAB dan Matplotlib adalah paket merencanakan yang menyediakan fungsi merencanakan seperti MATLAB. Secara internal, baik MATLAB dan NumPy mengandalkan BLAS dan LAPACK untuk perhitungan aljabar linier yang efisien.

Binding Python dari perpustakaan visi komputer yang banyak digunakan OpenCV memanfaatkan array NumPy untuk menyimpan dan beroperasi pada data. Karena gambar dengan banyak saluran hanya direpresentasikan sebagai array tiga dimensi, pengindeksan, pemotongan atau penutupan dengan array lainnya adalah cara yang sangat efisien untuk mengakses piksel tertentu dari suatu gambar. Array NumPy sebagai struktur data universal di OpenCV untuk gambar, poin fitur yang diekstraksi, filter kernel dan banyak lagi yang lebih menyederhanakan alur kerja pemrograman dan debugging.

### 3. Struktur Data Array

Fungsionalitas inti dari NumPy adalah "ndarray" -nya, untuk larik n-dimensi, struktur data. Array ini adalah tampilan langkah pada memori. Berbeda dengan struktur data daftar built-in Python (yang, terlepas dari namanya, adalah array dinamis), array ini diketik secara homogen: semua elemen dari array tunggal harus dari tipe yang sama.

Array tersebut juga dapat dilihat ke buffer memori yang dialokasikan oleh ekstensi C / C ++, Cython, dan Fortran ke juru bahasa CPython tanpa perlu menyalin data, memberikan tingkat kompatibilitas dengan perpustakaan numerik yang ada. Fungsi ini dieksplorasi oleh paket SciPy, yang membungkus sejumlah perpustakaan seperti itu (terutama BLAS dan LAPACK). NumPy memiliki dukungan bawaan untuk ndarrays yang dipetakan di memori. Untuk melakukan instalasi numpy, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install numpy” (versi numpy menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal numpy, dapat dilihat pada gambar 5.24 :

```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install numpy==1.14.2
Collecting numpy==1.14.2
  Downloading https://files.pythonhosted.org/packages/46/eb/846d92fed0ef6dbc1906c198e3e5475f1
d9f7954ace9648c05c0dfddc36b9(numpy-1.14.2-cp35-none-win_amd64.whl (13.4MB)
  100% |#####| 13.4MB 2.1MB/s
tensorboard 1.13.0 has requirement absl-py>=0.4, but you'll have absl-py 0.2.0 which is incompatible.
tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.5.2 which is incompatible.
tensorflow 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.5.2 which is incompatible.
tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have tensorboard 1.13.0 which is incompatible.
Installing collected packages: numpy
  Found existing installation: numpy 1.14.5
    Uninstalling numpy-1.14.5:
      Successfully uninstalled numpy-1.14.5
Successfully installed numpy-1.14.2
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Gambar 5. 24 Instalasi Numpy

### 5.1.7 Library Tensorflow-GPU

TensorFlow dapat dikonfigurasikan untuk dijalankan pada CPU atau GPU. Versi CPU jauh lebih mudah untuk diinstal dan dikonfigurasi sehingga merupakan tempat awal terbaik terutama ketika Anda pertama kali belajar bagaimana menggunakan TensorFlow. Berikut panduan tentang versi CPU vs. GPU dari situs web TensorFlow:

- TensorFlow dengan dukungan CPU saja . Jika sistem Anda tidak memiliki GPU NVIDIA®, Anda harus menginstal versi ini. Perhatikan bahwa versi TensorFlow ini biasanya jauh lebih mudah untuk diinstal (biasanya, dalam 5 atau 10 menit), jadi walaupun Anda memiliki GPU NVIDIA, kami sarankan untuk menginstal versi ini terlebih dahulu.
- TensorFlow dengan dukungan GPU Program TensorFlow biasanya berjalan secara signifikan lebih cepat pada GPU daripada pada CPU. Oleh karena itu, jika sistem Anda memiliki NVIDIA® GPU yang memenuhi prasyarat yang ditunjukkan di bawah ini dan Anda perlu

menjalankan aplikasi yang kritis terhadap kinerja, pada akhirnya Anda harus menginstal versi ini.

Jadi jika Anda baru memulai dengan TensorFlow, Anda mungkin ingin tetap menggunakan versi CPU untuk memulai, kemudian instal versi GPU begitu pelatihan Anda menjadi lebih menuntut komputasi. Untuk melakukan instalasi tensorflow-gpu, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install tensorflow-gpu” (tensorflow-gpu keras menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan apabila berhasil menginstal tensorflow-gpu :



```
Anaconda Prompt - pip install tensorflow-gpu==1.14.0
(tensorflow) C:\Users\ASUS>pip install tensorflow-gpu==1.14.0
Collecting tensorflow-gpu==1.14.0
  Using cached https://files.pythonhosted.org/packages/88/1a/4be03b24b89d75ff6951f14c1a0dd8646d8624ac404ec2adac3d3823314d/tensorflow_gpu-1.14.0-cp35-cp35m-win_amd64.whl
Requirement already satisfied: wheel>=0.26 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (0.31.1)
Requirement already satisfied: gast>=0.2.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (0.2.0)
Collecting numpy<2.0,>=1.14.5
  Downloading https://files.pythonhosted.org/packages/3a/18/7f8ef94683f2a45a786f47d48e8fd11e49cf1ff68b0b87054e5078f2b46/numpy-1.18.1-cp35-cp35m-win_amd64.whl (12.79B)
[##] 12.79B 93kB/s
Processing c:\users\asus\appdata\local\pip\cache\wheels\d7\de\fe\132238792febf6459a96e8597fc3d2ae51590df4fd\wrapt-1.11.2-cp35-none-any.whl
Requirement already satisfied: termcolor>=1.1.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (1.1.0)
Requirement already satisfied: astor>=0.6.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (0.6.2)
Collecting google-pasta>=0.1.6
  Downloading https://files.pythonhosted.org/packages/c3/fd/1e86bc4837cc9a3afaf3dbfb1854aa04ad35b5f381f9648fbe81a6f94e4/google_pasta-0.1.8-py3-none-any.whl (57KB)
[##] 61kB 20kB/s
Requirement already satisfied: grpcio>=1.8.6 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (1.12.1)
Collecting tensorflow-estimator<1.15.0rc0,>=1.14.0rc0
  Using cached https://files.pythonhosted.org/packages/3c/d5/21860a5b1caf0678fb8319341b0ae21a07156911132e0e71bfffed0510d/tensorflow_estimator-1.14.0-py2.py3-none-any.whl
Requirement already satisfied: six>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (1.13.0)
Collecting absl-py>=0.7.0
  Downloading https://files.pythonhosted.org/packages/1a/53/9243c600e047bd43df9e69cfabc1e004a82c2e0c484580a78a94ba2a/absl-py-0.9.0.tar.gz (104kB)
[##] 112kB 69kB/s
Requirement already satisfied: keras-applications>=1.0.6 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (1.0.8)
Collecting protobuf>=3.6.1
  Using cached https://files.pythonhosted.org/packages/9d/3b/53098b2881a32e7a27df921548e767730f6ea94a6d2e86dd88ac80955a3c/protobuf-3.11.2-cp35-cp35m-win_amd64.whl
Requirement already satisfied: keras-preprocessing>=1.0.5 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow-gpu==1.14.0) (1.1.0)
Collecting tensorboard<1.15.0,>=1.14.0
  Using cached https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a9b50ebd50123adf1f8cbea1492f9084169b89d9/tensorboard-1.14.0-py3-none-any.whl
Requirement already satisfied: h5py in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from keras-applications>=1.0.6->tensorflow-gpu==1.14.0) (2.10.0)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf>=3.6.1->tensorflow-gpu==1.14.0) (39.1.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow-gpu==1.14.0) (0.14.1)
Building wheels in directory: c:\users\asus\appdata\local\pip\cache\wheels\8e\28\49\fad4e7f0b9a1227708cbbe4487ac8558a7334849cb81c813d
  Created wheel for absl-py: filename=absl-py-0.9.0-cp35-none-any.whl size=117806 sha256=3738ee6e89233c54b2a1b5f799b5e1cff22775f5dc3a4e145588865bf38047e
  Stored in directory: c:\users\ASUS\appdata\local\pip\cache\wheels\8e\28\49\fad4e7f0b9a1227708cbbe4487ac8558a7334849cb81c813d
Successfully built absl-py
```

Gambar 5. 25 Instalasi Tensorflow-gpu

### 5.1.8 Library matplotlib

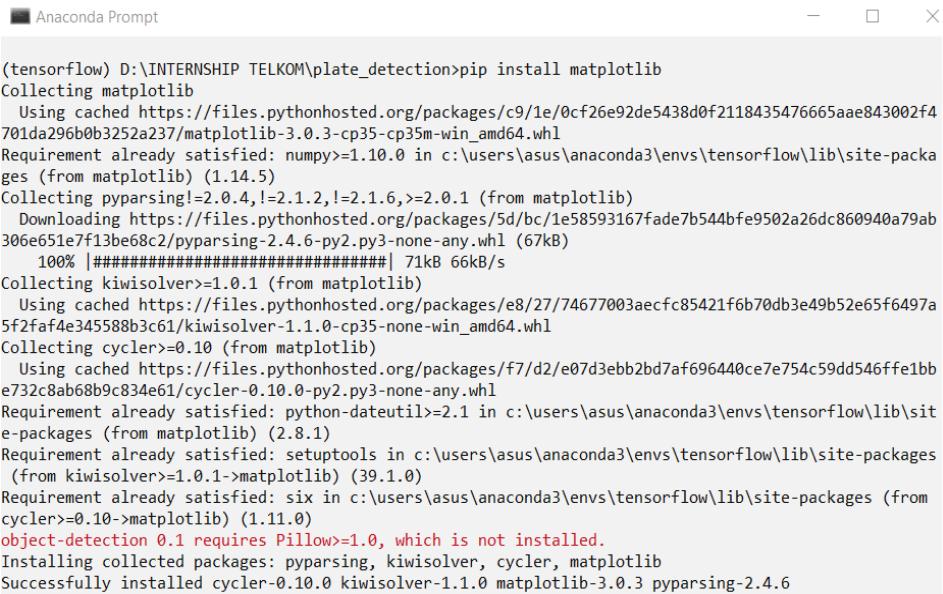
Matplotlib merupakan library plot Python 2D yang dapat menghasilkan angka kualitas publikasi dalam berbagai format hardcopy dan lingkungan interaktif lintas platform. Matplotlib sebagai library Python

2D yang dapat menghasilkan plot dengan kualitas tinggi dalam berbagai format dan dapat digunakan di banyak platform.

Matplotlib berfungsi sebagai pembuat grafik dalam berbagai platform, seperti Python dan Jupyter. Grafik yang dapat dibuat beragam, seperti grafik garis, batang, lingkaran, histogram, dsb. Cara menginstal library matplotlib dapat dilakukan dengan secara manual. Akan tetapi lebih mudah apabila menginstal library matplotlib melalui anaconda cmd, Dimana kita hanya menjalankan pip install matplotlib ke dalamnya. Maka, pip akan mengunduh matplotlib sekaligus numpy, kiwisolver, python-dateutil, cycler, six, serta pyparsing.

Data yang kita olah tentu tidak elok apabila ditampilkan begitu saja dengan tabel hitam saja kepada investor atau manajemen. Bila ditampilkan dengan sejumlah grafik berwarna pasti mereka akan lebih tertarik melihatnya. Matplotlib memiliki fungsi untuk memvisualisasikan data dengan lebih indah dan rapi.

Terdapat sebuah plot untuk menampilkan data secara 2D atau 3D. Sehingga kamu dapat menampilkan data yang telah kamu olah sesuai kebutuhan. Matplotlib terhubung dengan iPython Notebook atau Jupyter dimana kamu dapat membuat sebuah buku interaktif yang dapat diberi penjelasan dan kode yang disisipkan begitupun hasil plottingnya. Matplotlib library sering dan paling banyak digunakan oleh data science untuk menyajikan datanya ke dalam visual yang lebih baik. Untuk melakukan instalasi matplotlib, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install matplotlib” (versi matplotlib menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal matplotlib, dapat dilihat pada gambar 5.26 :



```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install matplotlib
Collecting matplotlib
  Using cached https://files.pythonhosted.org/packages/c9/1e/0cf26e92de5438d0f2118435476665aae843002f4
701da296b0b3252a237/matplotlib-3.0.3-cp35-cp35m-win_amd64.whl
Requirement already satisfied: numpy>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-pac
ges (from matplotlib)
Collecting pyparsing!=2.0.4,!2.1.2,!2.1.6,>=2.0.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/5d/bc/1e58593167fade7b544bfe9502a26dc860940a79ab
306e651e7f13be68c2/pyparsing-2.4.6-py2.py3-none-any.whl (67kB)
    100% [#####] 71kB 66kB/s
Collecting kiwisolver>=1.0.1 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/e8/27/74677003aecfc85421f6b70db3e49b52e65f6497a
5f2faf4e345588b3c61/kiwisolver-1.1.0-cp35-none-win_amd64.whl
Collecting cycler>=0.10 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bb
e732c8ab68b9c834e61/cycler-0.10.0-py2.py3-none-any.whl
Requirement already satisfied: python-dateutil>=2.1 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-pac
ges (from matplotlib) (2.8.1)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages
 (from kiwisolver>=1.0.1>matplotlib) (39.1.0)
Requirement already satisfied: six in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from
cycler>=0.10->matplotlib) (1.11.0)
object-detection 0.1 requires Pillow>=1.0, which is not installed.
Installing collected packages: pyparsing, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.1.0 matplotlib-3.0.3 pyparsing-2.4.6
```

Gambar 5. 26 Instalasi Matplotlib

### 5.1.9 Library pillow

Pustaka Pencitraan Python (PIL) dalam versi yang lebih baru yang dikenal sebagai Pillow adalah pustaka gratis untuk bahasa pemrograman Python yang menambahkan dukungan untuk membuka, memanipulasi, dan menyimpan banyak format file gambar yang berbeda. Ini tersedia untuk Windows, Mac OS X dan Linux. Versi terbaru dari PIL adalah 1.1.7, dirilis pada September 2009 dan mendukung Python 1.5.2-2.7, dengan dukungan Python 3 akan dirilis "nanti".

Pengembangan tampaknya dihentikan dengan komitmen terakhir ke repositori PIL yang akan datang pada 2011. Akibatnya, proyek penerus yang disebut Pillow telah memotong repositori PIL dan menambahkan dukungan Python 3.x. Garpu ini telah diadopsi sebagai pengganti PIL asli dalam distribusi Linux termasuk Debian dan Ubuntu (sejak 13.04). Beberapa format file yang didukung adalah PPM, PNG, JPEG, GIF, TIFF,

dan BMP. Dimungkinkan juga untuk membuat decoder file baru untuk memperluas pustaka format file yang dapat diakses.

Pillow menawarkan beberapa prosedur standar untuk manipulasi gambar. Yaitu :

- manipulasi per-pixel,
- masking dan penanganan transparansi,
- pemfilteran gambar, seperti mengaburkan, membentuk, menghaluskan, atau menemukan tepi,
- peningkatan gambar, seperti mempertajam, menyesuaikan kecerahan, kontras atau warna,
- menambahkan teks ke gambar dan banyak lagi.

Untuk melakukan instalasi pillow, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install pillow” (versi pillow menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal pillow, dapat dilihat pada gambar 5.27 :

```
■ Anaconda Prompt
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install matplotlib
Collecting matplotlib
  Using cached https://files.pythonhosted.org/packages/c9/1e/0cf26e92de5438d0f2118435476665aae843002f4701da296b0b3252a237/matplotlib-3.0.3-cp35-cp35m-win_amd64.whl
Requirement already satisfied: numpy>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from matplotlib) (1.14.5)
Collecting pyparsing!=2.0.4,!2.1.2,!2.1.6,>2.0.1 (from matplotlib)
  Downloading https://files.pythonhosted.org/packages/5d/bc/1e58593167fade7b544bfe9502a26d860940a79ab306e651e7f13be68c2/pyparsing-2.4.6-py2.py3-none-any.whl (67 kB)
  100% |################################| 71KB 66kB/s
Collecting kiwisolver>=1.0.1 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/e8/27/74677003aecfc85421fb670db3e49b52e65f6497a5f2faf4e345588b3c61/kiwisolver-1.1.0-cp35-none-win_amd64.whl
Collecting cycler>0.10 (from matplotlib)
  Using cached https://files.pythonhosted.org/packages/f7/d2/e07d3eb2bd7af696440ce7e754c59dd546ffe1bbe732c8eb68b9c834e61/cycler-0.10.0-py2.py3-none-any.whl
Requirement already satisfied: python-dateutil<2.1 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from matplotlib) (39.1.0)
Requirement already satisfied: six in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from kiwisolver>1.0.1>matplotlib) (1.11.0)
object-detection 0.1 requires Pillow<1.0, which is not installed.
Installing collected packages: pyparsing, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.1.0 matplotlib-3.0.3 pyparsing-2.4.6
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install -upgrade pip' command.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install pillow
Collecting pillow
  Downloading https://files.pythonhosted.org/packages/24/ab/0fffd4690d15eb08039a278b173fac2ede5d4139998195a6de3dd370399f4/Pillow-7.0.0-cp35-cp35m-win_amd64.whl (2.0MB)
  100% |################################| 2.0MB 58kB/s
Installing collected packages: pillow
```

Gambar 5. 27 Instalasi Pillow

### 5.1.10 Library absl-py

Repositori ini adalah kumpulan kode pustaka Python untuk membangun aplikasi Python. Kode dikumpulkan dari basis kode Python Google sendiri, dan telah diuji secara luas dan digunakan dalam produksi.

1. Fiturnya adalah sebagai berikut :

- Startup aplikasi sederhana
- Sistem flag commandline terdistribusi
- Modul logging khusus dengan fitur tambahan
- Utilitas pengujian

2. Instalasi

Untuk melakukan instalasi absl-py, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install absl-py” (versi absl-py menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal absl-py, dapat dilihat pada gambar 5.28:

```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install absl-py==0.2.0
Collecting absl-py==0.2.0
  Downloading https://files.pythonhosted.org/packages/90/6b/ba04a9fe6aefa56ada
fa6b9e0557b959e423c49950527139cb8651b0480b/absl-py-0.2.0.tar.gz (82kB)
    100% |#####| 92kB 207kB/s
Requirement already satisfied: six in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from absl-py==0.2.0) (1.13.0)
Building wheels for collected packages: absl-py
  Running setup.py bdist_wheel for absl-py ... done
  Stored in directory: C:\Users\ASUS\AppData\Local\pip\Cache\wheels\23\35\1d\4
8c0a173ca38690dd8dfccfa47ffc750db48f8989ed898455c
Successfully built absl-py
tensorflow 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.
5.2 which is incompatible.
tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have
tensorboard 1.13.0 which is incompatible.
tensorboard 1.13.0 has requirement absl-py>=0.4, but you'll have absl-py 0.2.0
which is incompatible.
tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.
5.2 which is incompatible.
Installing collected packages: absl-py
  Found existing installation: absl-py 0.4.1
    Uninstalling absl-py-0.4.1:
      Successfully uninstalled absl-py-0.4.1
Successfully installed absl-py-0.2.0
```

Gambar 5. 28 Instalasi absl-py

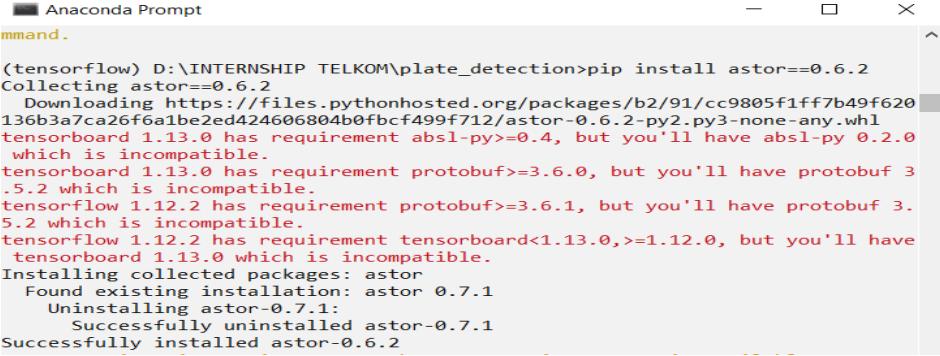
### **5.1.11 Library astor**

Library astor dirancang untuk memungkinkan manipulasi sumber Python melalui AST. Ada beberapa perpustakaan serupa lainnya, tetapi astor berfokus pada bidang-bidang berikut:

- Round-trip ke Python melalui modul codegen.py Armin Ronacher: AST yang dimodifikasi tidak perlu linenumbers, ctx, dll. Atau dapat langsung dikompilasi
- Dump pretty- printing of AST Lebih sulit dibaca daripada kode yang tersandung, tetapi lebih akurat untuk mengetahui apa yang sedang terjadi. - Lebih mudah membaca daripada membuang dari modul AST bawaan
- Kadang-kadang Anda menginginkan treewalk rekursif (dan astor mendukungnya, mulai dari sembarang simpul pada pohon), tetapi kadang-kadang Anda tidak perlu melakukan itu. astor tidak mengharuskan Anda untuk mengunjungi sub-node secara eksplisit kecuali Anda ingin: - Anda dapat menambahkan kode yang dieksekusi sebelum anak-anak node dikunjungi, dan / atau - Anda dapat menambahkan kode yang mengeksekusi setelah anak-anak node dikunjungi, dan / atau - Anda dapat menambahkan kode yang mengeksekusi dan menjaga agar anak-anak node tidak dikunjungi (dan opsional mengunjungi mereka sendiri melalui panggilan rekursif) - Menulis fungsi untuk mengakses pohon berdasarkan nama objek dan / atau nama atribut - Nikmati akses mudah ke simpul orangtua (s) untuk penulisan ulang pohon.

Untuk melakukan instalasi astor, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install astor” (versi astor menyesuaikan dengan kebutuhan

machine learning yang akan dibuat). Berikut tampilan apabila berhasil menginstal astor :



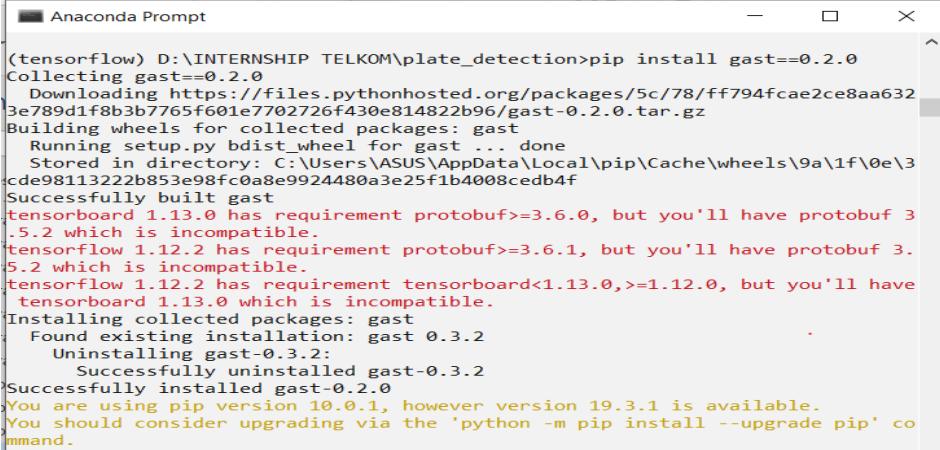
```
Anaconda Prompt
mmand.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install astor==0.6.2
Collecting astor==0.6.2
  Downloading https://files.pythonhosted.org/packages/b2/91/cc9805f1ff7b49f620136b3a7ca26f6a1be2ed424606804b0fbef499f712/astor-0.6.2-py2.py3-none-any.whl
    tensorboard 1.13.0 has requirement absl-py>=0.4, but you'll have absl-py 0.2.0
      which is incompatible.
    tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.5.2 which is incompatible.
    tensorflow 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.5.2 which is incompatible.
    tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have
      tensorboard 1.13.0 which is incompatible.
Installing collected packages: astor
  Found existing installation: astor 0.7.1
    Uninstalling astor-0.7.1:
      Successfully uninstalled astor-0.7.1
Successfully installed astor-0.6.2
```

Gambar 5. 29 Instalasi astor

### 5.1.12 Library gast

GAST adalah toolkit untuk menggunakan kemampuan Sensing Android. Ini berisi contoh, kode pelat ketel, dan algoritma yang Anda butuhkan untuk menggunakan sensor Android dengan benar. GAST menyediakan lapisan kompatibilitas antara AST dari berbagai versi Python, seperti yang diproduksi oleh ast.parse dari modul ast standar. Berikut tampilan cmd apabila berhasil menginstal gast, dapat dilihat pada gambar 5.30 :



```
Anaconda Prompt
mmand.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install gast==0.2.0
Collecting gast==0.2.0
  Downloading https://files.pythonhosted.org/packages/5c/78/ff794fcac2ce8aa6323e789d1f8b3b765f601e7702726f430e814822b96/gast-0.2.0.tar.gz
Building wheels for collected packages: gast
  Running setup.py bdist_wheel for gast ... done
    Stored in directory: C:\Users\ASUS\AppData\Local\pip\Cache\wheels\9a\1f\0e\3cd98113222b853e98fc0a8e9924480a3e25f1b4008cedb4f
Successfully built gast
tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.5.2 which is incompatible.
tensorflow 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.5.2 which is incompatible.
tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have
  tensorboard 1.13.0 which is incompatible.
Installing collected packages: gast
  Found existing installation: gast 0.3.2
    Uninstalling gast-0.3.2:
      Successfully uninstalled gast-0.3.2
Successfully installed gast-0.2.0
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Gambar 5. 30 Instalasi gast

### **5.1.13 Library html5lib**

Library html5lib adalah pustaka python murni untuk parsing HTML. Ini dirancang untuk memenuhi spesifikasi HTML WHATWG, seperti yang diterapkan oleh semua browser web utama.

### **5.1.14 Library Markdown**

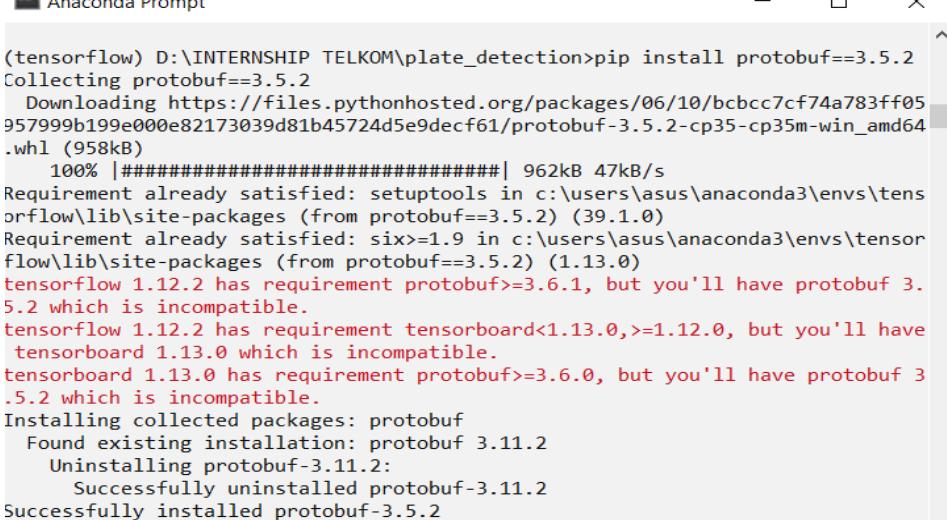
Python-Markdown dimaksudkan untuk menjadi modul perpustakaan python yang digunakan oleh berbagai proyek untuk mengubah sintaks Markdown menjadi HTML. Python-Markdown menyediakan dua fungsi publik (`markdown.markdown` dan `markdown.markdownFromFile`) yang keduanya membungkus kelas publik `markdown.Markdown`. Jika Anda memproses satu dokumen pada satu waktu, fungsi-fungsi ini akan memenuhi kebutuhan Anda. Namun, jika Anda perlu memproses beberapa dokumen, mungkin menguntungkan untuk membuat instance tunggal dari `markdown.Markdown`kelas dan meneruskan banyak dokumen melalui itu. Jika Anda menggunakan satu instance, pastikan untuk memanggil reset metode dengan tepat (lihat di bawah). Untuk melakukan instalasi `markdown`, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “`pip install markdown`” (versi `markdown` menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd, apabila berhasil menginstal `markdown` dapat dilihat pada gambar 5.31 :

```
(tensorflow) C:\Users\ASUS>pip install Markdown==2.6.1
Collecting Markdown==2.6.1
  Downloading https://files.pythonhosted.org/packages/75/01/1e68a8d388d3f9fa741a098546379fb0c1684b9f5a07725e5c69638bd578/Markdown-2.6.1.tar.gz (298kB)
    #| 10kB 26bytes/s eta 3:02:59  |##| 20kB 52bytes/s eta 1:28
    :15 |###| 30kB 78bytes/s eta 0:56:40 |###| 40kB 104bytes/s eta
  0:40:5 |#####| 51kB 131bytes/s eta 0:31:2 |#####| 61kB 157bytes/s eta
ta 0:25:0 |#####| 71kB 183bytes/s eta 0:20:3 |#####| 81kB 209bytes/s eta
s eta 0:17:1 |#####| 92kB 235bytes/s eta 0:14:3 |#####| 102kB 262bytes/s eta
tes/s eta 0:12: |#####| 112kB 262bytes/s eta 0:11: |#####| 122kB 262bytes/s eta
2bytes/s eta 0:11: |#####| 133kB 262bytes/s eta 0:10: |#####| 143kB
  262bytes/s eta 0:09: |#####| 153kB 262bytes/s eta 0:09: |#####| 16
3kB 262bytes/s eta 0:08: |#####| 174kB 262bytes/s eta 0:07: |#####
  184kB 262bytes/s eta 0:07: |#####| 194kB 262bytes/s eta 0:06: |#####
  | 204kB 262bytes/s eta 0:05: |#####| 215kB 262bytes/s eta 0:05: |#####
  | 225kB 262bytes/s eta 0:04: |#####| 235kB 262bytes/s eta 0:03: |#####
## | 245kB 262bytes/s eta 0:03: |#####| 256kB 262bytes/s eta 0:02: |#####
##### | 266kB 262bytes/s eta 0:02: |#####| 276kB 262bytes/s eta 0:01: |#####
##### | 307kB 99kB/s
Building wheels for collected packages: Markdown
  Building wheel for Markdown (setup.py) ... done
  Created wheel for Markdown: filename=Markdown-2.6.1-cp35-none-any.whl size=158137 sha256=7a3b4b28d87dfd0be5f2962ec6f6fd79cf10f1
  03aa4880ee6dfc808380365085
  Stored in directory: C:\Users\ASUS\AppData\Local\pip\Cache\wheels\ae\06\78\b978b04af2ae422cf45ca5e90e5400b0146946720e334bf5
Successfully built Markdown
```

Gambar 5. 31 Instalasi Markdown

### 5.1.15 Library protobuf

Protokol buffer adalah solusi fleksibel, efisien, otomatis untuk menyelesaikan masalah ini dengan tepat. Dengan buffer protokol, Anda menulis .protodeskripsi tentang struktur data yang ingin Anda simpan. Dari itu, kompiler buffer protokol menciptakan kelas yang mengimplementasikan pengkodean otomatis dan penguraian data buffer protokol dengan format biner yang efisien. Kelas yang dihasilkan menyediakan getter dan setter untuk bidang yang membentuk buffer protokol dan mengurus detail membaca dan menulis buffer protokol sebagai satu unit. Yang penting, format buffer protokol mendukung gagasan untuk memperluas format dari waktu ke waktu sedemikian rupa sehingga kode masih dapat membaca data yang dikodekan dengan format lama. Untuk melakukan instalasi protobuf, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install ptotobuf” (versi protobuf menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal protobuf, dapat dilihat pada gambar 5.32:

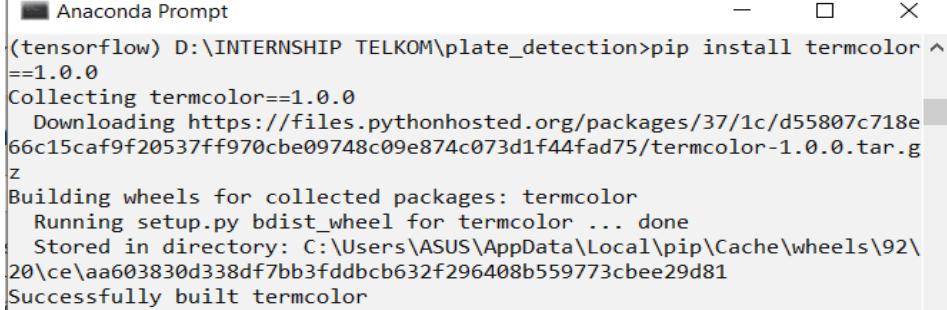


```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install protobuf==3.5.2
Collecting protobuf==3.5.2
  Downloading https://files.pythonhosted.org/packages/06/10/bcbcc7cf74a783ff05957999b199e000e82173039d81b45724d5e9decf61/protobuf-3.5.2-cp35-cp35m-win_amd64.whl (958kB)
    100% |#####| 962kB 47kB/s
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf==3.5.2) (39.1.0)
Requirement already satisfied: six>=1.9 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf==3.5.2) (1.13.0)
tensorforce 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.5.2 which is incompatible.
tensorforce 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have tensorboard 1.13.0 which is incompatible.
tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.5.2 which is incompatible.
Installing collected packages: protobuf
  Found existing installation: protobuf 3.11.2
    Uninstalling protobuf-3.11.2:
      Successfully uninstalled protobuf-3.11.2
Successfully installed protobuf-3.5.2
```

Gambar 5. 32 Instalasi Protobuf

### 5.1.16 Library termcolor

Termcolor adalah pustaka C ++ khusus header untuk mencetak pesan berwarna ke terminal. Ditulis hanya untuk bersenang-senang dengan bantuan the Force. Untuk melakukan instalasi termcolor, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install termcolor” (versi termcolor menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal termcolor, dapat dilihat pada gambar 5.33 :



```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install termcolor ==1.0.0
Collecting termcolor==1.0.0
  Downloading https://files.pythonhosted.org/packages/37/1c/d55807c718e66c15caf9f20537ff970cbe09748c09e874c073d1f44fad75/termcolor-1.0.0.tar.gz
Building wheels for collected packages: termcolor
  Running setup.py bdist_wheel for termcolor ... done
    Stored in directory: C:\Users\ASUS\AppData\Local\pip\Cache\wheels\92\20\ce\aa603830d338df7bb3fddbc632f296408b559773cbee29d81
Successfully built termcolor
```

Gambar 5. 33 Instalasi termcolor

### **5.1.17 Library Werkzeug**

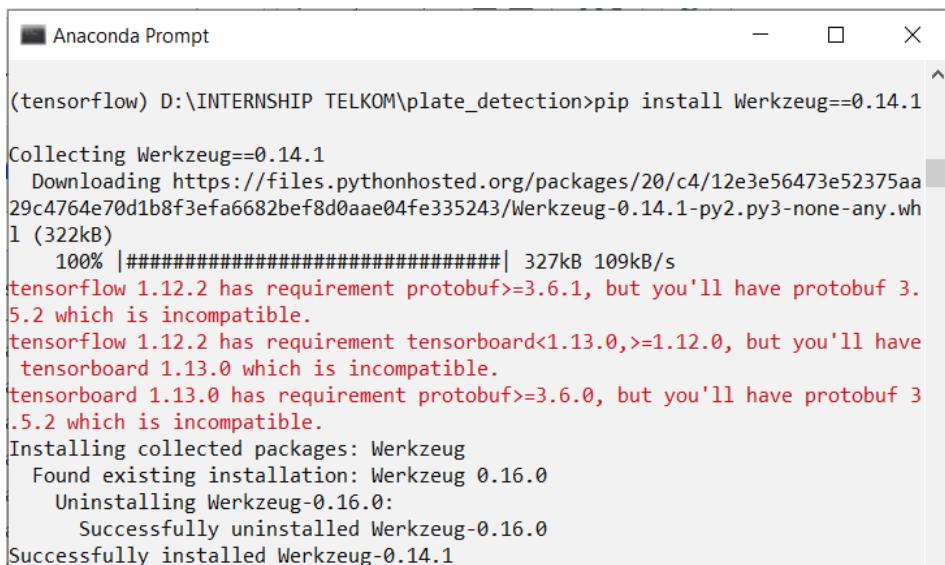
Library Werkzeug adalah perpustakaan aplikasi web WSGI yang komprehensif. Ini dimulai sebagai kumpulan sederhana berbagai utilitas untuk aplikasi WSGI dan telah menjadi salah satu perpustakaan utilitas WSGI paling canggih.

Itu termasuk:

- Debugger interaktif yang memungkinkan memeriksa jejak stack dan kode sumber di browser dengan interpreter interaktif untuk setiap bingkai di stack.
- Objek permintaan berfitur lengkap dengan objek untuk berinteraksi dengan header, argumen permintaan, data formulir, file, dan cookie.
- Objek respons yang dapat membungkus aplikasi WSGI lainnya dan menangani streaming data.
- Sistem perutean untuk mencocokkan URL ke titik akhir dan menghasilkan URL untuk titik akhir, dengan sistem yang dapat diperluas untuk menangkap variabel dari URL.
- Utilitas HTTP untuk menangani tag entitas, kontrol cache, tanggal, agen pengguna, cookie, file, dan banyak lagi.
- Server WSGI berulir untuk digunakan saat mengembangkan aplikasi secara lokal.
- Klien uji untuk mensimulasikan permintaan HTTP selama pengujian tanpa perlu menjalankan server.

Werkzeug sadar akan Unicode dan tidak memberlakukan dependensi apa pun. Terserah pengembang untuk memilih mesin template, adaptor database, dan bahkan bagaimana menangani permintaan. Ini dapat digunakan untuk membangun segala macam aplikasi pengguna akhir seperti blog, wiki, atau papan buletin. Flask membungkus Werkzeug,

menggunakannya untuk menangani detail WSGI sambil memberikan lebih banyak struktur dan pola untuk mendefinisikan aplikasi yang kuat. Instal dan perbarui. Untuk melakukan instalasi werkzeug, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install werkzeug” (versi werkzeug menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal werkzeug, dapat dilihat pada gambar 5.34 :



```
Anaconda Prompt
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install Werkzeug==0.14.1
Collecting Werkzeug==0.14.1
  Downloading https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
    100% |#####| 327kB 109kB/s
tensorflow 1.12.2 has requirement protobuf>=3.6.1, but you'll have protobuf 3.5.2 which is incompatible.
tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'll have tensorboard 1.13.0 which is incompatible.
tensorboard 1.13.0 has requirement protobuf>=3.6.0, but you'll have protobuf 3.5.2 which is incompatible.
Installing collected packages: Werkzeug
  Found existing installation: Werkzeug 0.16.0
    Uninstalling Werkzeug-0.16.0:
      Successfully uninstalled Werkzeug-0.16.0
Successfully installed Werkzeug-0.14.1
```

Gambar 5. 34 Instalasi werlzeug

### 5.1.18 Library contextlib2

contextlib2 adalah backport dari modul contextlib perpustakaan standar ke versi Python sebelumnya. Ini juga berfungsi sebagai tanah pembuktian dunia nyata untuk kemungkinan peningkatan di masa depan ke versi perpustakaan standar. contextlib2 tidak memiliki dependensi runtime, tetapi membutuhkan unittest2 untuk pengujian pada Python 2.x, serta setuptools dan roda untuk menghasilkan arsip roda universal. Pengujian lokal hanya masalah menjalankan python test\_contextlib2.py.

Versi yang saat ini diuji dalam toks dan Travis CI adalah:

- CPython 2.7
- CPython 3.4
- CPython 3.5
- CPython 3.6
- CPython 3.7
- PyPy
- PyPy3

Untuk melakukan instalasi cpython, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install cpython” (versi cpython menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal contextlib2, dapat dilihat pada gambar 5.35 :

```
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install contextlib2==0.6.0
Collecting contextlib2==0.6.0
  Using cached https://files.pythonhosted.org/packages/cf/e5/989798d38831a8505d626f2e25f9add8ed675dc1f/contextlib2-0.6.0-py2.py3-none-any.whl
Installing collected packages: contextlib2
  Found existing installation: contextlib2 0.6.0.post1
    Uninstalling contextlib2-0.6.0.post1:
      Successfully uninstalled contextlib2-0.6.0.post1
Successfully installed contextlib2-0.6.0
```

Gambar 5. 35 Instalasi contextlib2

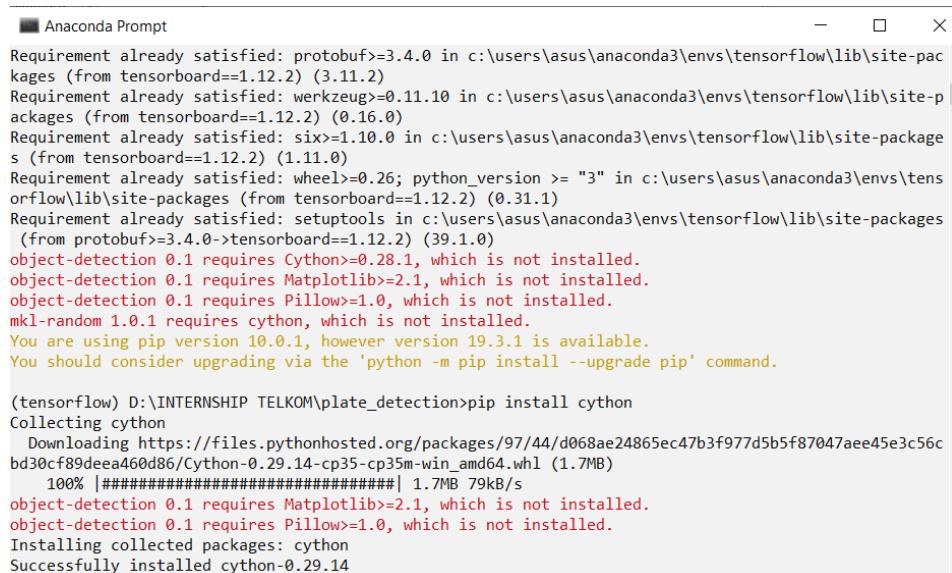
### 5.1.19 Library Cython

Cython adalah kompiler statis yang mengoptimalkan untuk bahasa pemrograman Python dan bahasa pemrograman Cython yang diperluas (berdasarkan Pyrex). Itu membuat menulis ekstensi C untuk Python semudah Python itu sendiri.

Hampir semua bagian dari kode Python juga merupakan kode Cython yang valid. (Ada beberapa Keterbatasan , tetapi perkiraan ini akan

berfungsi untuk saat ini.) Kompiler Cython akan mengubahnya menjadi kode C yang membuat panggilan yang setara ke API Python / C.

Tetapi Cython jauh lebih dari itu, karena parameter dan variabel dapat dinyatakan memiliki tipe data C. Kode yang memanipulasi nilai Python dan nilai C dapat secara bebas dicampur, dengan konversi yang terjadi secara otomatis jika memungkinkan. Pemeliharaan penghitungan referensi dan pengecekan kesalahan operasi Python juga otomatis, dan kekuatan penuh fasilitas penanganan pengecualian Python, termasuk pernyataan coba-kecuali dan coba-akhirnya, tersedia untuk Anda - bahkan di tengah-tengah memanipulasi data C. Untuk melakukan instalasi cython, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install cython” (versi cython menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal cython, dapat dilihat pada gambar 5.36 :



```
Anaconda Prompt
Requirement already satisfied: protobuf>=3.4.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (3.11.2)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.16.0)
Requirement already satisfied: six>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (1.11.0)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.12.2) (0.31.1)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf>=3.4.0->tensorflow==1.12.2) (39.1.0)
object-detection 0.1 requires Cython>=0.28.1, which is not installed.
object-detection 0.1 requires Matplotlib>=2.1, which is not installed.
object-detection 0.1 requires Pillow>=1.0, which is not installed.
mkl-random 1.0.1 requires cython, which is not installed.
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install cython
Collecting cython
  Downloading https://files.pythonhosted.org/packages/97/44/d068ae24865ec47b3f977d5b5f87047aee45e3c56cbd30cf89deea460d86/cython-0.29.14-cp35-cp35m-win_amd64.whl (1.7MB)
    100% [##] 1.7MB 79kB/s
object-detection 0.1 requires Matplotlib>=2.1, which is not installed.
object-detection 0.1 requires Pillow>=1.0, which is not installed.
Installing collected packages: cython
Successfully installed cython-0.29.14
```

Gambar 5. 36 Instalasi cython

### **5.1.20 Library lxml**

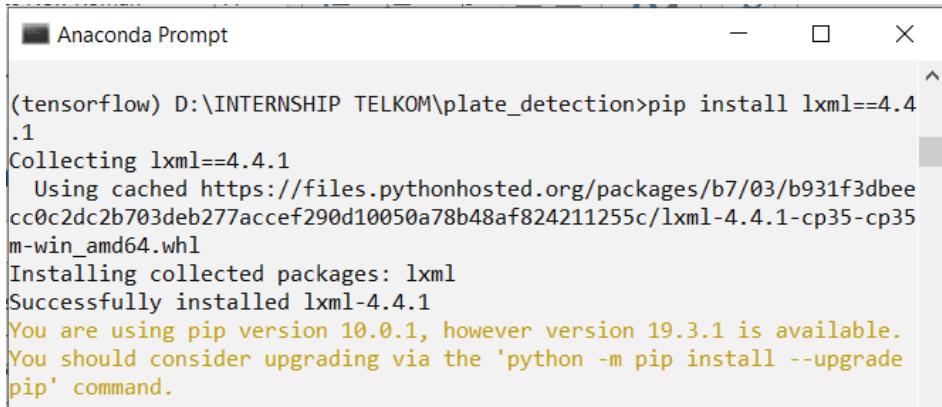
Toolkit lxml XML adalah pengikat Pythonic untuk pustaka C libxml2 dan libxslt. Ini unik karena menggabungkan kecepatan dan kelengkapan fitur XML dari pustaka ini dengan kesederhanaan API Python asli, sebagian besar kompatibel tetapi lebih unggul dari ElementTree API yang terkenal . Rilis terbaru berfungsi dengan semua versi CPython dari 2,7 hingga 3,7. Lihat pengantar untuk informasi lebih lanjut tentang latar belakang dan tujuan proyek lxml. Beberapa pertanyaan umum dijawab dalam FAQ.

lxml adalah Pythonic, pengikatan dewasa untuk pustaka libxml2 dan libxslt. Ini memberikan akses yang aman dan nyaman ke perpustakaan ini menggunakan ElementTree API. Ini memperluas ElementTree API secara signifikan untuk menawarkan dukungan untuk XPath, RelaxNG, XML Schema, XSLT, C14N dan banyak lagi.

lxml telah diunduh dari Indeks Paket Python jutaan kali dan juga tersedia langsung di banyak distribusi paket, misalnya untuk Linux atau MacOS-X. Kebanyakan orang yang menggunakan lxml melakukannya karena mereka suka menggunakannya. Anda dapat menunjukkan kepada kami bahwa Anda menyukainya dengan membuat blog tentang pengalaman Anda dengannya dan menghubungkan ke situs web proyek.

Jika Anda menggunakan lxml untuk pekerjaan Anda dan merasa ingin memberikan sedikit keuntungan Anda sendiri untuk mendukung proyek, pertimbangkan mengirim uang melalui GitHub Sponsors, Tidelift atau PayPal yang dapat kami gunakan untuk memberi kami waktu luang untuk pemeliharaan yang hebat ini. perpustakaan, untuk memperbaiki bug dalam perangkat lunak, meninjau dan mengintegrasikan kontribusi kode, dan meningkatkan fitur dan dokumentasinya. Silakan baca Pemberitahuan

Hukum di bawah ini, di bagian bawah halaman ini. Terima kasih atas dukungan Anda. Untuk melakukan instalasi lxml, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install lxml” (versi lxml menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal lxml, dapat dilihat pada gambar 5.37 :



```
Anaconda Prompt
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install lxml==4.4.1
Collecting lxml==4.4.1
  Using cached https://files.pythonhosted.org/packages/b7/03/b931f3dbec
cc0c2dc2b703deb277accef290d10050a78b48af824211255c/lxml-4.4.1-cp35-cp35
m-win_amd64.whl
Installing collected packages: lxml
Successfully installed lxml-4.4.1
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade
pip' command.
```

Gambar 5. 37 Instalasi lxml

### 5.1.21 Library json-scema

Json-scema adalah library yang memungkinkan Anda untuk membuat anotasi dan memvalidasi dokumen JSON. jsonschema adalah implementasi JSON Schema for Python (mendukung 2.7+ termasuk Python 3). Dokumen JSON yang divalidasi atau dijelaskan kami sebut sebagai instance, dan dokumen yang berisi deskripsi disebut skema. Skema paling dasar adalah objek JSON kosong, yang tidak membatasi apa pun, mengizinkan apa pun, dan tidak menjelaskan apa pun.

Anda dapat menerapkan batasan pada instance dengan menambahkan kata kunci validasi ke skema. Misalnya, kata kunci "type" dapat digunakan untuk membatasi turunan ke objek, array, string, angka, boolean, atau null. Skema JSON adalah hypermedia siap, dan ideal untuk membubuhkan

keterangan API HTTP berbasis JSON yang ada. Dokumen Skema JSON diidentifikasi oleh URI, yang dapat digunakan dalam header HTTP Link, dan di dalam dokumen Skema JSON untuk memungkinkan definisi rekursif.

Proyek json-scema bermaksud untuk menggembalakan keempat seri konsep ke status RFC. Saat ini, kami terus meningkatkan Konsep-Internet yang dipublikasikan sendiri. Langkah selanjutnya adalah membuat draft diadopsi oleh Kelompok Kerja IETF. Kami sedang aktif menyelidiki bagaimana mencapai ini.

Keuntungan json-scema adalah :

- Menjelaskan format data yang ada.
- Memberikan dokumentasi yang bisa dibaca manusia dan mesin.
- Memvalidasi data yang berguna untuk:
- Pengujian otomatis.
- Memastikan kualitas data yang dikirimkan klien.

Skema JSON adalah bahasa deklaratif untuk memvalidasi format dan struktur Obyek JSON. Ini memungkinkan kita untuk menentukan jumlah primitif khusus untuk menggambarkan dengan tepat seperti apa objek JSON yang valid akan terlihat.

The JSON Skema spesifikasi dibagi menjadi tiga bagian:

1. JSON Schema Core : Spesifikasi JSON Schema Core adalah tempat terminologi untuk suatu skema didefinisikan.
2. Validasi Skema JSON : Spesifikasi Validasi Skema JSON adalah dokumen yang mendefinisikan cara-cara yang valid untuk mendefinisikan batasan validasi. Dokumen ini juga mendefinisikan serangkaian kata kunci yang dapat digunakan untuk menentukan

validasi untuk API JSON. Pada contoh berikut, kami akan menggunakan beberapa kata kunci ini.

3. JSON Hyper-Schema : Ini adalah ekstensi lain dari spesifikasi JSON Schema, di mana kata kunci yang terkait dengan hyperlink dan hypermedia didefinisikan.

### **5.1.22 Library Scikit-Learn**

Machine learning ada dua yaitu ada yang berbasis statistika ada juga yang tidak. Salah satunya adalah support vector machine dan regresi linier. Mungkin bagi sebagian orang sudah biasa menulis sendiri library untuk implementasi kedua algoritma tadi. Tapi untuk membuatnya dalam waktu singkat tentu butuh waktu yang tidak sedikit pula.

Scikit-Learn verfungsi memberikan sejumlah fitur untuk keperluan data science seperti:

- Algoritma Regresi
- Algoritma Naive Bayes
- Algoritma Clustering
- Algoritma Decision Tree
- Parameter Tuning
- Data Preprocessing Tool
- Export / Import Model
- Machine learning pipeline dan lainnya

Scikit-Learn sudah teruji dan memiliki dokumentasi yang super lengkap. Bahkan kontributornya pun banyak. Scikit-Learn juga menyediakan ekstensi untuk fuzzy logic dan computer vision. Untuk melakukan instalasi scikit-learn, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install scikit-learn” (versi scikit-learn menyesuaikan dengan

kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal scikit-learn, dapat dilihat pada gambar 5.38 :

```
(tensorflow) C:\Users\ASUS>pip install scikit-learn
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/08/d6/64c94f5f5599ed5283ec876e102a546179b680af78b908c5ec221c474e95/scikit_learn-0.22.1-cp35-cp35m-win_amd64.whl (6.2MB)
    |#####| 6.2MB 2.2MB/s
Collecting joblib<=0.11
  Downloading https://files.pythonhosted.org/packages/28/5c/cf6a2b65a321c4a209efcdf64c2689efae2cb62661f8f6f4bb28547cf1bf/joblib-0.14.1-py2.py3-none-any.whl (294kB)
    |#####| 296kB 819kB/s
Requirement already satisfied: numpy<=1.11.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from scikit-learn) (1.18.1)
Requirement already satisfied: scipy<=0.17.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from scikit-learn) (1.4.1)
Installing collected packages: joblib, scikit-learn
Successfully installed joblib-0.14.1 scikit-learn-0.22.1

(tensorflow) C:\Users\ASUS>
```

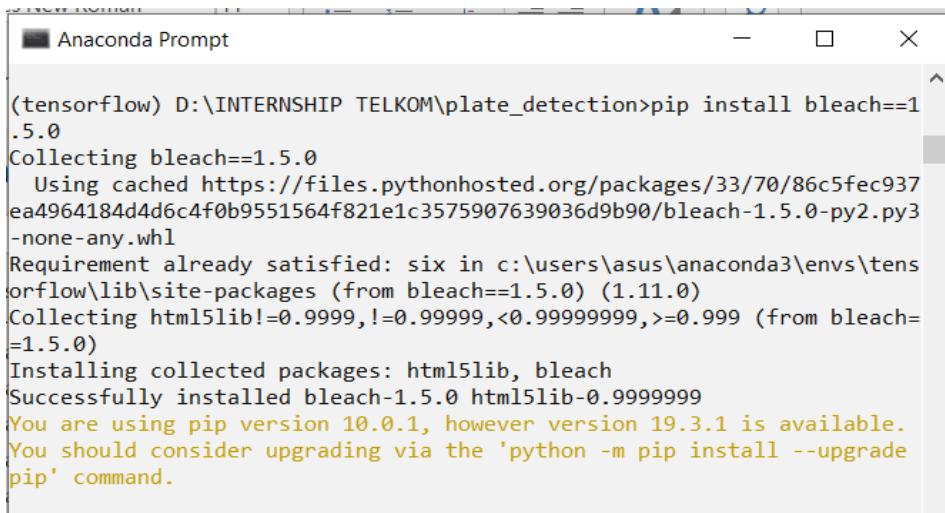
Gambar 5. 38 Instalasi Scikit-learn

### 5.1.23 Library Bleach

Bleach, untuk membersihkan HTML dan secara otomatis menautkan URL dalam teks, yang kami gunakan di addons.mozilla.org dan support.mozilla.com. Ini didasarkan pada html5lib, yang kami pilih karena dasarnya dalam standar web dan komunitas pengembangan aktif. Bleach tersedia di Github dan PyPI. Bleach adalah perpustakaan sanitasi HTML berbasis daftar diizinkan yang lolos atau menghapus markup dan atribut.

Bleach juga dapat menautkan teks dengan aman, menerapkan filter yang tidak dapat disaring oleh filter Django, dan secara opsional mengatur atribut rel , bahkan pada tautan yang sudah ada dalam teks. Bleach dimaksudkan untuk membersihkan teks dari sumber yang tidak terpercaya . Jika Anda menemukan diri Anda melompat melewati lingkaran untuk memungkinkan administrator situs Anda melakukan banyak hal, Anda mungkin berada di luar kasus penggunaan. Entah mempercayai para pengguna itu, atau tidak. Bleach tersedia di PyPI, sehingga Anda dapat

menginstalnya dengan pip : “pip install bleach”. Berikut tampilan cmd apabila berhasil menginstal scikit-learn, dapat dilihat pada gambar 5.39 :



```
Anaconda Prompt
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install bleach==1.5.0
Collecting bleach==1.5.0
  Using cached https://files.pythonhosted.org/packages/33/70/86c5fec937ea4964184d4d6c4f0b9551564f821e1c3575907639036d9b90/bleach-1.5.0-py2.py3-none-any.whl
Requirement already satisfied: six in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from bleach==1.5.0) (1.11.0)
Collecting html5lib!=0.9999,!>=0.99999,<0.99999999,>=0.999 (from bleach==1.5.0)
  Installing collected packages: html5lib, bleach
    Successfully installed bleach-1.5.0 html5lib-0.9999999
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Gambar 5. 39 Instalasi bleach

### 5.1.24 Library Tensorboard

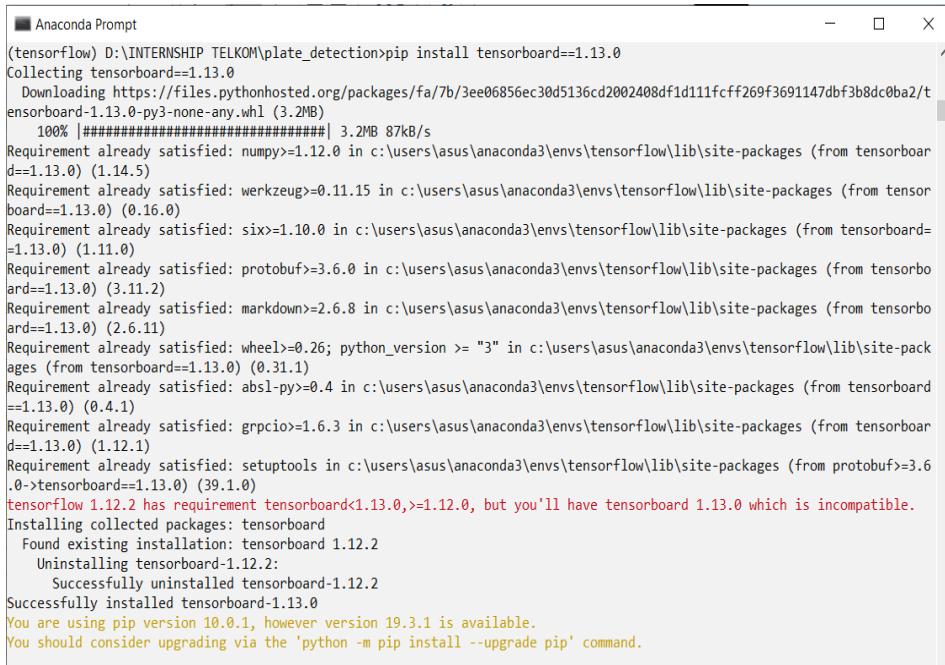
TensorBoard, alat visualisasi TensorFlow, sering digunakan oleh para peneliti dan insinyur untuk memvisualisasikan dan memahami eksperimen ML mereka. Ini memungkinkan pelacakan metrik eksperimen, memvisualisasikan model, membuat profil program ML, memvisualisasikan eksperimen penyetelan hyperparameter, dan banyak lagi.

Sementara TensorBoard memudahkan untuk memvisualisasikan eksperimen Anda sendiri, pembelajaran mesin sering melibatkan kolaborasi. Anda mungkin ingin membagikan penelitian Anda tentang efek hyperparameter, menjelaskan prosedur pelatihan yang rumit, atau mendapatkan bantuan pemecahan masalah perilaku model aneh.

Kami telah melihat orang berbagi tangkapan layar TensorBoards mereka untuk mencapai ini. Namun, tangkapan layar tidak interaktif dan gagal menangkap semua detail. Di Google, peneliti dan insinyur sering

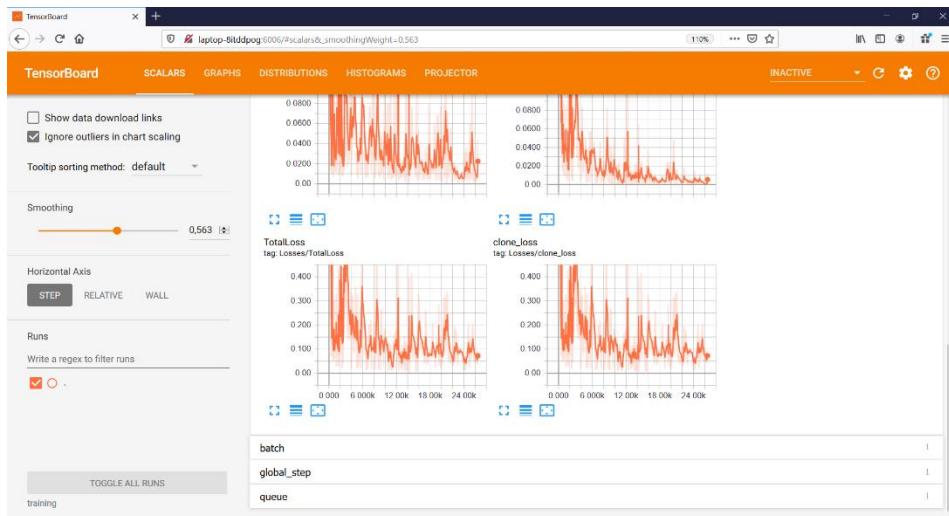
mengomunikasikan wawasan mereka tentang perilaku model dengan mengirimkan visualisasi TensorBoard mereka ke rekan tim. Tujuan kami adalah untuk memberikan kemampuan ini kepada komunitas yang lebih luas.

Itu sebabnya kami meluncurkan TensorBoard.dev: layanan terkelola (saat ini dalam pratinjau) yang memungkinkan Anda untuk meng-host, melacak, dan berbagi eksperimen ML dengan mudah secara gratis. Cukup unggah log TensorBoard Anda dan terima tautan yang dapat dilihat oleh semua orang, tanpa instalasi atau pengaturan. Untuk melakukan instalasi tensorflow, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install tensorflow” (versi tensorflow menyesuaikan dengan kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal tensorflow, dapat dilihat pada gambar 5.40 :



```
[■ Anaconda Prompt
(tensorflow) D:\INTERNSHIP\plate_detection>pip install tensorflow==1.13.0
Collecting tensorflow==1.13.0
  Downloading https://files.pythonhosted.org/packages/fa/7b/3ee06856ec30d5136cd2002408df1d111fcff269f3691147dbf3b8dc0ba2/tensorboard-1.13.0-py3-none-any.whl (3.2MB)
    100% [##] 3.2MB 87kB/s
Requirement already satisfied: numpy>=1.12.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (1.14.5)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (0.16.0)
Requirement already satisfied: six>=1.10.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (1.11.0)
Requirement already satisfied: protobuf>=3.6.0 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (3.11.2)
Requirement already satisfied: markdown>=2.6.8 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (2.6.11)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (0.31.1)
Requirement already satisfied: absl-py>=0.4 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (0.4.1)
Requirement already satisfied: grpcio>=1.6.3 in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from tensorflow==1.13.0) (1.12.1)
Requirement already satisfied: setuptools in c:\users\asus\anaconda3\envs\tensorflow\lib\site-packages (from protobuf>=3.6.0->tensorflow==1.13.0) (39.1.0)
tensorflow 1.12.2 has requirement tensorflow<1.13.0,>=1.12.0, but you'll have tensorflow 1.13.0 which is incompatible.
Installing collected packages: tensorflow
  Found existing installation: tensorflow 1.12.2
  Uninstalling tensorflow-1.12.2:
    Successfully uninstalled tensorflow-1.12.2
Successfully installed tensorflow-1.13.0
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Gambar 5. 40 Instalasi Tensorboard



Gambar 5. 41 Visualisasi Tensorboard

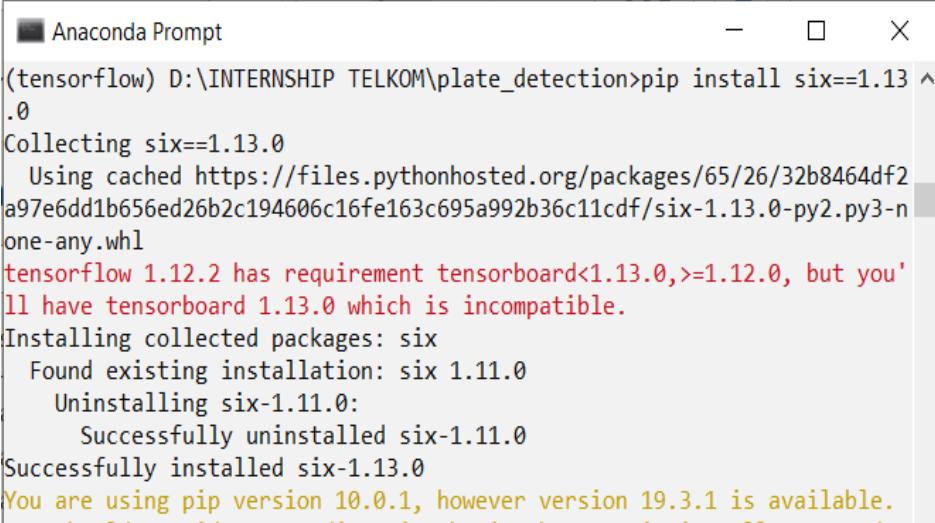
### 5.1.25 Library six

Six adalah library kompatibilitas Python 2 dan 3. Ini menyediakan fungsi utilitas untuk memperlancar perbedaan antara versi Python dengan tujuan menulis kode Python yang kompatibel pada kedua versi Python. Lihat dokumentasi untuk informasi lebih lanjut tentang apa yang disediakan. Enam mendukung setiap versi Python sejak 2.6.

Six menyediakan utilitas sederhana untuk membungkus perbedaan antara Python 2 dan Python 3. Hal ini dimaksudkan untuk mendukung basis kode yang berfungsi pada kedua Python 2 dan 3 tanpa modifikasi. Six hanya terdiri dari satu file Python, jadi tidak sulit untuk menyalin ke suatu proyek.

Ini terkandung hanya dalam satu file Python, sehingga dapat dengan mudah disalin ke proyek Anda. (Pemberitahuan hak cipta dan lisensi harus dipertahankan). Untuk melakukan instalasi six, buka anaconda command prompt kemudian aktivasi environment yang telah dibuat, dan jalankan perintah berikut “pip install six” (versi six menyesuaikan dengan

kebutuhan machine learning yang akan dibuat). Berikut tampilan cmd apabila berhasil menginstal six, dapat dilihat pada gambar 5.42 :



```
Anaconda Prompt
(tensorflow) D:\INTERNSHIP TELKOM\plate_detection>pip install six==1.13.^
.0
Collecting six==1.13.0
  Using cached https://files.pythonhosted.org/packages/65/26/32b8464df2
a97e6dd1b656ed26b2c194606c16fe163c695a992b36c11cdf/six-1.13.0-py2.py3-n
one-any.whl
tensorflow 1.12.2 has requirement tensorboard<1.13.0,>=1.12.0, but you'
ll have tensorboard 1.13.0 which is incompatible.
Installing collected packages: six
  Found existing installation: six 1.11.0
    Uninstalling six-1.11.0:
      Successfully uninstalled six-1.11.0
Successfully installed six-1.13.0
You are using pip version 10.0.1, however version 19.3.1 is available.

```

Gambar 5. 42 Instalasi six

## **BAB VI**

---

### **TUTORIAL OBJECT DETECTION PLATE NUMBER**

Sebelum melakukan tutorial penulis akan menjelaskan sedikit pemaparan latar belakang mengapa system objek deteksi plat nomor ini dibuat dan hasil yang diperoleh dari sistem yang sudah berhasil dibuat dan diuji. Salah satu permasalahan pada computer vision yang masih berkembang adalah object detection sebagai teknologi yang berguna untuk mengenali objek pada gambar selayaknya manusia dengan pembelajaran pada sebuah komputer dengan menggunakan jaringan saraf tiruan. Salah satu sub tipe jaringan saraf tiruan yang menangani permasalahan computer vision adalah Convolutional Neural Network (CNN).

CNN menjadi salah satu metode yang sedang banyak diterapkan dalam deteksi objek, dengan beberapa pembaharuan CNN berevolusi menjadi arsitektur Faster-RCNN yang mampu mendeteksi objek dengan sangat baik. Penelitian ini memanfaatkan kemampuan Faster-RCNN dalam mendeteksi plat nomor kendaraan. Kemudian untuk menangani masalah object detection maka digunakanlah framework tensorflow, tensorflow digunakan sebagai support agar object detection dapat melakukan training data dan testing data.

Berdasarkan uraian diatas, maka peneliti membuat sebuah sistem pengolahan data gambar untuk deteksi plat nomor kendaraan menggunakan metode CNN. Dengan menggunakan Faster Regional-Convolution Neural Network (R-CNN) sebagai model arsitektur CNN. Bahasa pemograman yang digunakan adalah Python. Hasil penelitian ini sistem objek deteksi yang dibuat dapat mendeteksi plat nomor kendaraan dengan inputan gambar, video dan secara real time di webcam. Hasil

penelitian menunjukkan confidence rate dari 50 file gambar, 1 file video dan secara live yang uji pada sistem objek deteksi rata-rata mendapat persentase 82-100%.

## 6.1 Mempersiapkan Tools

TensorFlow-GPU memungkinkan PC Anda menggunakan kartu video untuk memberikan daya pemrosesan ekstra saat pelatihan, sehingga akan digunakan untuk tutorial ini. Dalam pengalaman saya, menggunakan TensorFlow-GPU sebagai ganti TensorFlow biasa mengurangi waktu pelatihan dengan faktor sekitar 8 (3 jam untuk melatih, bukan 24 jam). Versi CPU-only dari TensorFlow juga dapat digunakan untuk tutorial ini, tetapi akan memakan waktu lebih lama. Jika Anda menggunakan TensorFlow khusus CPU, Anda tidak perlu menginstal CUDA dan cuDNN

Untuk membuat system objek deteksi maka diperlukan beberapa tools yang dapat mendukung dalam pembuatannya dan tools ini sudah dibahas pada bab sebelumnya, berikut tools yang harus dipersiapkan dalam pembuatan system object deteksi :

- Anaconda
- CUDA
- cuDNN
- LabelImg

Jika Anda menggunakan versi TensorFlow yang lebih lama, pastikan Anda menggunakan versi CUDA dan cuDNN yang kompatibel dengan versi TensorFlow yang Anda gunakan. Versi Anaconda saat ini menggunakan Python 3.7, yang tidak secara resmi didukung oleh TensorFlow.

## **6.2 Mengatur Direktori TensorFlow dan Lingkungan Virtual Anaconda**

API Deteksi Objek TensorFlow mengharuskan menggunakan struktur direktori spesifik yang disediakan dalam repositori GitHub-nya. Ini juga memerlukan beberapa paket Python tambahan, tambahan spesifik untuk variabel PATH dan PYTHONPATH, dan beberapa perintah pengaturan tambahan untuk mengatur semuanya agar berjalan atau melatih model deteksi objek. Bagian tutorial ini membahas pengaturan lengkap yang diperlukan. Ini cukup teliti, tetapi ikuti instruksi dengan seksama, karena pengaturan yang tidak tepat dapat menyebabkan kesalahan berat di jalan.

### **6.2.1 Unduh repositori API Deteksi Objek TensorFlow dari GitHub**

Buat folder langsung di D: dan beri nama "detection\_plate\_number". Direktori kerja ini nantinya akan berisi kerangka kerja pendeksi objek TensorFlow lengkap, serta gambar pelatihan, data pelatihan, classifier terlatih, file konfigurasi, dan segala sesuatu yang diperlukan untuk classifier deteksi objek. Unduh repositori deteksi objek TensorFlow lengkap yang terdapat di [https://github.com/miabb999/OBJECT\\_DETECTION\\_PLATE](https://github.com/miabb999/OBJECT_DETECTION_PLATE) dengan mengeklik tombol “Kloning atau Unduh” dan mengunduh file zip. Buka file zip yang diunduh dan ekstrak folder "model-master" langsung ke direktori D:\ detection\_plate\_number yang baru saja Anda buat.

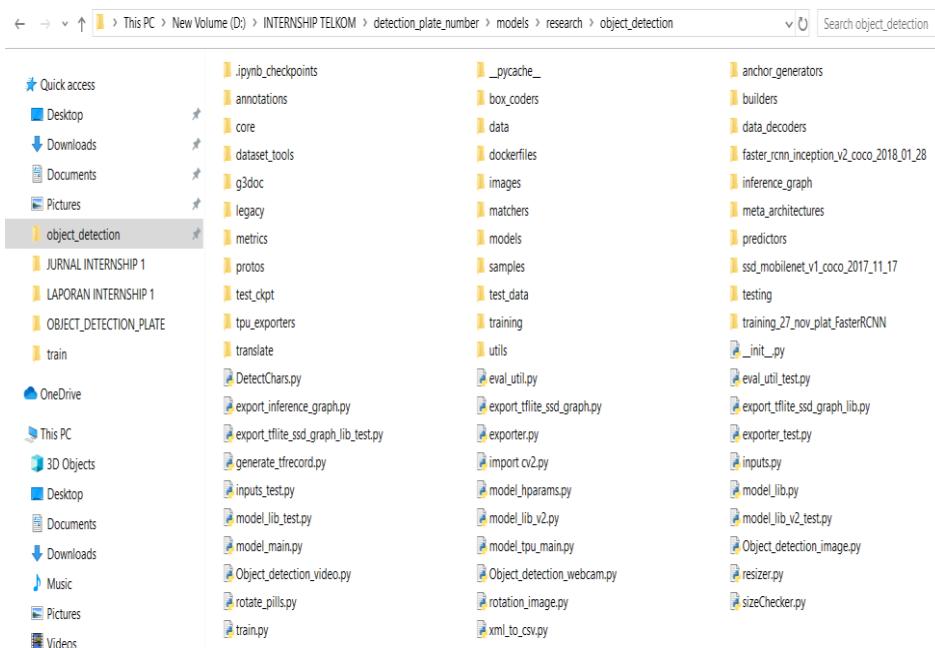
### **6.2.2 Unduh model Faster-RCNN-Inception**

TensorFlow menyediakan beberapa model deteksi objek (pengklasifikasi pra-terlatih dengan arsitektur jaringan saraf. Tutorial ini akan menggunakan model Faster-RCNN-Inception-V2. Unduh modelnya di sini. Buka file fast\_rcnn\_inception\_v2\_coco\_2018\_01\_28.tar.gz yang diunduh dengan pengarsipan file seperti WinZip atau 7-Zip dan ekstrak

folder fast\_rcnn\_inception\_v2\_coco\_2018\_01\_28 ke folder D:\ detection\_plate\_number \ models \ research \ object\_detection. (Catatan: Tanggal dan versi model kemungkinan akan berubah di masa mendatang, tetapi masih harus bekerja dengan tutorial ini.).

### 6.2.3 Unduh repositori tutorial di r GitHub

Unduh repositori lengkap yang terdapat di halaman ini (gulir ke atas dan klik Klon atau Unduh) dan ekstrak semua konten langsung ke direktori D:\ detection\_plate\_number \ models \ research \ object\_detection. Pada titik ini, inilah tampilan folder \ object\_detection Anda:



Gambar 6. 1 Folder Object Detection

Repositori ini berisi gambar, data anotasi, file .csv, dan TFRecords yang diperlukan untuk melatih detektor plate. Anda dapat menggunakan gambar dan data ini untuk berlatih membuat detector plate Anda sendiri. Folder ini juga berisi skrip Python yang digunakan untuk menghasilkan

data pelatihan. Terdapat skrip untuk menguji klasifikasi deteksi objek pada gambar, video, atau umpan webcam.

Jika Anda ingin melatih detektor objek Anda sendiri, hapus file-file berikut (jangan hapus folder):

- Semua file dalam \object\_detection\images\train and \object\_detection\images\test
- File "test\_labels.csv" dan "train\_labels.csv" di \object\_detection\images
- Semua file dalam \object\_detection\training
- Semua file di \object\_detection\inference\_graph

Sekarang, Anda siap untuk memulai dari awal dalam melatih detektor objek Anda sendiri. Tutorial ini akan mengasumsikan bahwa semua file yang tercantum di atas telah dihapus, dan selanjutnya akan menjelaskan cara membuat file untuk dataset pelatihan Anda sendiri.

#### **6.2.4 Pengaturan Anaconda Virtual Environment**

Selanjutnya melakukan pengaturan path di Anaconda untuk tensorflow-gpu. Dari menu Start di Windows, cari utilitas Anaconda Prompt, klik kanan padanya, dan klik "Run as Administrator". Jika Windows bertanya apakah Anda ingin mengizinkannya untuk melakukan perubahan pada komputer Anda, klik Ya.

- Pada anaconda command prompt, buat environment dengan nama "tensorflow1" dengan menggunakan perintah berikut :  
“conda create -n tensorflow1 pip python=3.5”
- Kemudian, aktifkan environment dan perbarui pip dengan menjalankan perintah berikut:  
“activate tensorflow1”  
“python -m pip install --upgrade pip”

- Instal tensorflow-gpu di lingkungan ini dengan perintah:  
“pip install --ignore-installed --upgrade tensorflow-gpu”
- Instal paket-paket lain yang diperlukan dengan mengeluarkan perintah berikut:
  - conda install -c anaconda protobuf
  - pip install pillow
  - pip install lxml
  - pip install Cython
  - pip install contextlib2
  - pip install jupyter
  - pip install matplotlib
  - pip install pandas
  - pip install opencv-python

Catatan: Package 'pandas' dan 'opencv-python' tidak diperlukan oleh TensorFlow, tetapi mereka digunakan dalam skrip Python untuk menghasilkan TFRecords dan untuk bekerja dengan gambar, video, dan umpan webcam.

### **6.2.5 Konfigurasi8u PYTHONPATH environment variable**

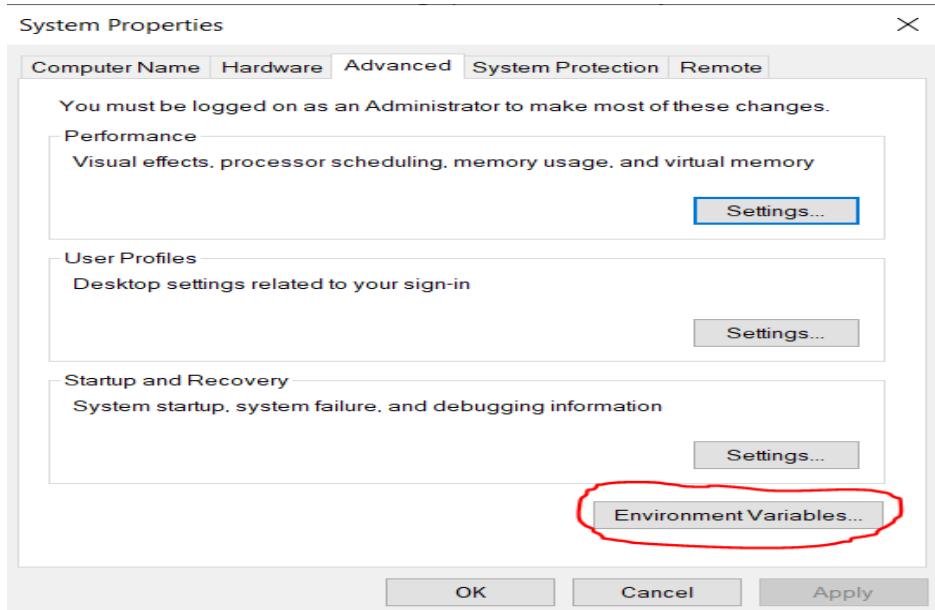
Variabel PYTHONPATH harus dibuat yang menunjuk ke direktori \ models, \ models \ research, dan \ models \ research \ slim. Lakukan ini dengan mengeluarkan perintah berikut (dari direktori mana saja) :

```
(tensorflow1) D:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim
```

Catatan : Setiap kali lingkungan virtual "tensorflow1" keluar, variabel PYTHONPATH diatur ulang dan perlu disetel lagi. Anda dapat menggunakan "echo% PYTHONPATH%" untuk melihat apakah telah

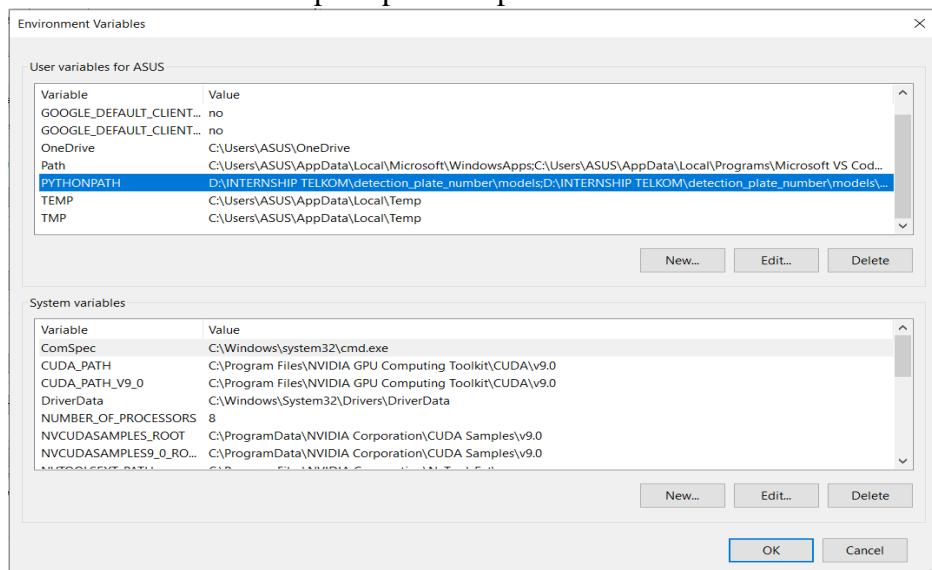
disetel atau tidak. Berikut contoh pengaturan pythonpath untuk objek deteksi :

- Buka Environment Variable lalu lakukan double klik



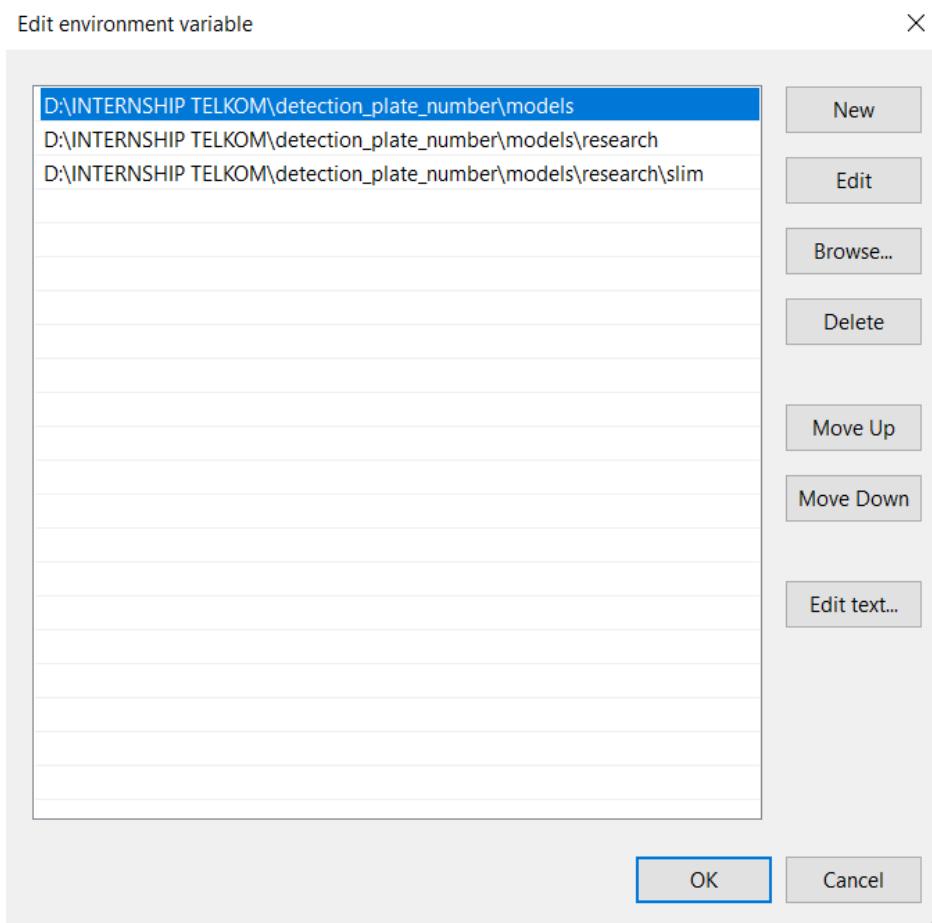
Gambar 6. 2 Environment Variabel

- Kemudian akan tampil seperti tampilan berikut



Gambar 6. 3 Python Path

- Atur path anaconda kalian seperti berikut kemudian beri nama pythonpath



Gambar 6. 4 Python Path Anaconda

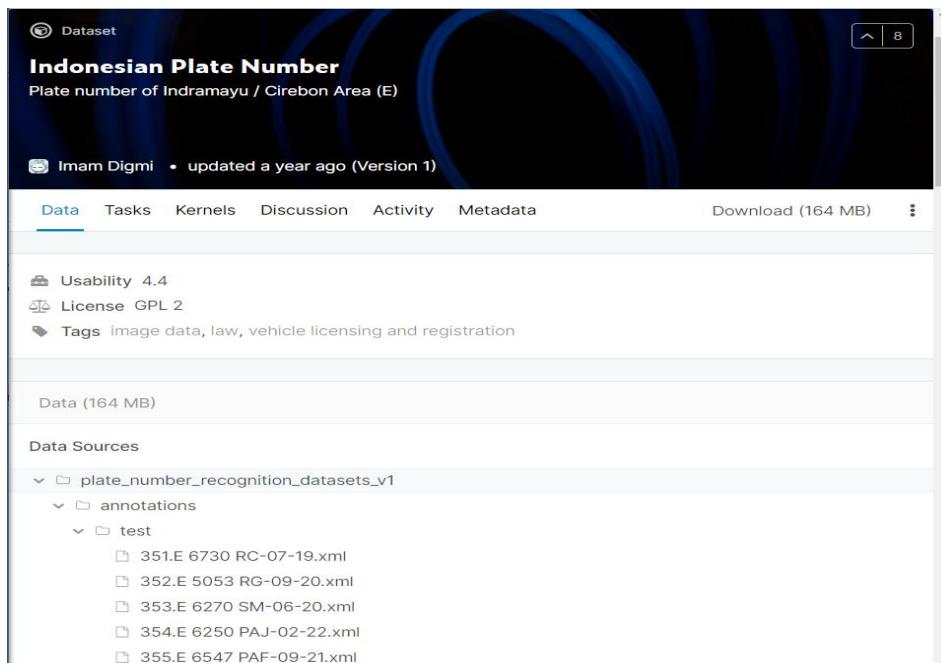
### 6.3 Mempersiapkan Dataset

TensorFlow membutuhkan ratusan gambar suatu objek untuk melatih pengelompokan deteksi yang baik. Untuk melatih pengklasifikasi yang kuat, gambar pelatihan harus memiliki objek acak dalam gambar bersama dengan objek yang diinginkan, dan harus memiliki berbagai latar belakang dan kondisi pencahayaan. Harus ada beberapa gambar di mana objek yang diinginkan sebagian dikaburkan, tumpang tindih dengan sesuatu yang lain,

atau hanya setengah di dalam gambar. Kemudian resolusi dari gambar terdapat yang bagus dan buruk, kemudian jarak dari camera ke objek ada yang jauh dan ada yang dekat.

### 6.3.1 Pengumpulan Data

Jika kita ingin mengenali objek “khusus” yang ingin kita deteksi, tentu kita memerlukan dataset untuk proses training, sehingga Neural Network yang akan kita latih untuk mengenali objek tersebut dapat mengenali objek yang kita maksud. Anda dapat menggunakan telepon Anda untuk mengambil gambar objek atau mengunduh gambar objek dari Google Image Search. Apabila anda tidak ingin membuat data set maka anda dapat menggunakan dataset penulis pada link github yang sudah dipaparkan sebelumnya atau mendapatkan dataser dari repository online yaitu di situs Kaggle dengan link berikut “<https://www.kaggle.com/imamdigmi/indonesian-plate-number>“.



Gambar 6. 5 Dataset Kaggle

Pengumpulan data yang digunakan merupakan data berupa gambar. Data tersebut meliputi data gambar plat motor dan mobil yang di ambil langsung dari repository Kaggle dan dibuat atau diambil sendiri oleh penulis menggunakan smartphone. Data gambar berisi plat nomor kendaraan yang berada di area parkir dan jalanan yang diambil dengan berbagai angle foto mulai dari lurus hingga miring, kemudian warna dari plat nomor bermacam-macam yaitu hitam, merah dan kuning. Data tersebut dapat dilihat pada tabel 5.1 sebagai berikut:

Tabel 6. 1 Pengumpulan Data

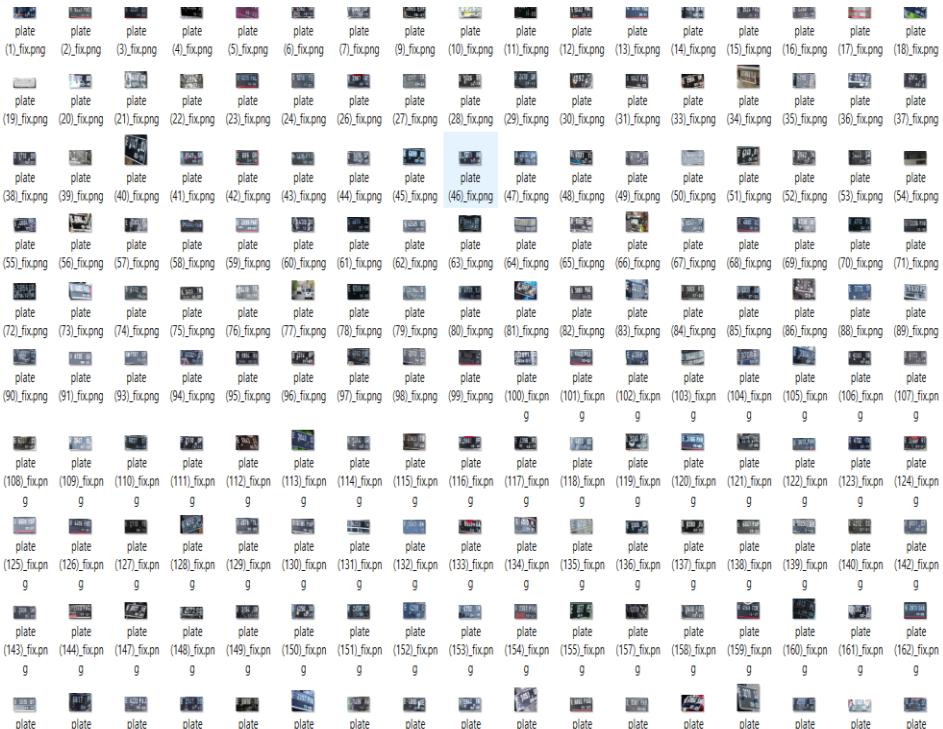
| NO | Sumber Data | Jenis Data   | Lokasi Data        | Jumlah Data | Format Data |
|----|-------------|--------------|--------------------|-------------|-------------|
| 1. | Kaggle      | Data Train   | Tempat kejadian    | 472 Gambar  | Png         |
| 2. | Smartphone  | Data Train   | Parkiran & Jalanan | 453 Gambar  | Png         |
| 3. | Kaggle      | Data Test    | Tempat kejadian    | 30 Gambar   | Png         |
| 4. | Google      | Data Test    | Tempat kejadian    | 76 Gambar   | Png         |
| 5. | Google      | Data Testing | Tempat kejadian    | 49 Gambar   | Jpg         |

Berdasarkan data pada table 6.1, data yang digunakan berasal dari kaggle dengan jumlah 472 gambar untuk data train & 30 Gambar untuk data test. Kemudian data yang dibuat langsung menggunakan smartphone berjumlah 453 untuk data train. Data yang di dapat dari google berjumlah 76 Gambar untuk data test dan 49 gambar untuk data pengujian. Dari data tersebut didapatkan total jumlah data yang digunakan dalam penelitian ini

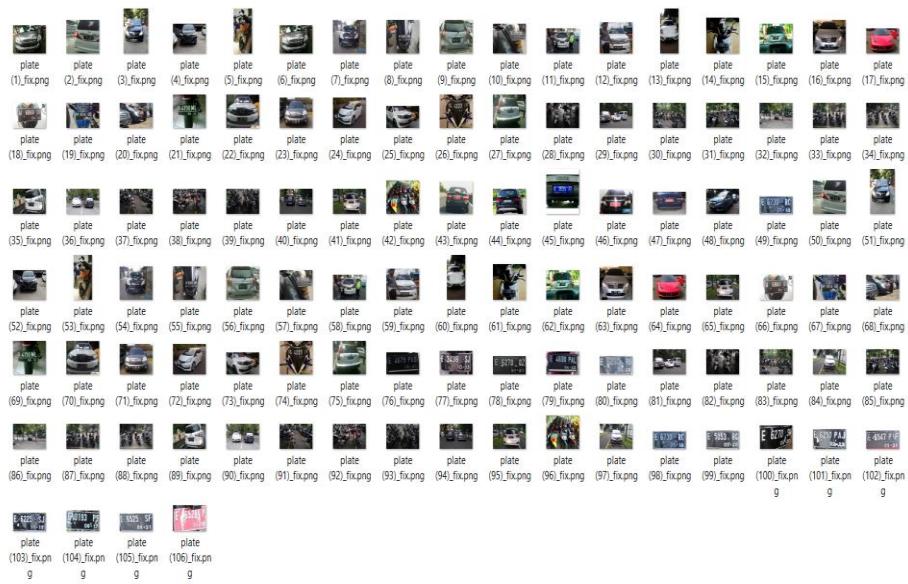
yaitu 1.080 gambar, dengan demikian data tersebut dapat diolah untuk dijadikan dataset yaitu data training dan data test.

Pastikan gambarnya tidak terlalu besar. Masing-masing harus kurang dari 200KB, dan resolusi mereka tidak boleh lebih dari 720x1280. Semakin besar gambar, semakin lama waktu yang dibutuhkan untuk melatih penggolong. Anda dapat menggunakan skrip resizer.py dalam repositori ini untuk mengurangi ukuran gambar. Setelah Anda memiliki semua gambar yang Anda butuhkan, pindahkan 20% dari mereka ke direktori tes \ object\_detection \ images \ , dan 80% dari mereka ke direktori \ object\_detection \ images \ train. Pastikan ada berbagai gambar di direktori \ test dan \ train.

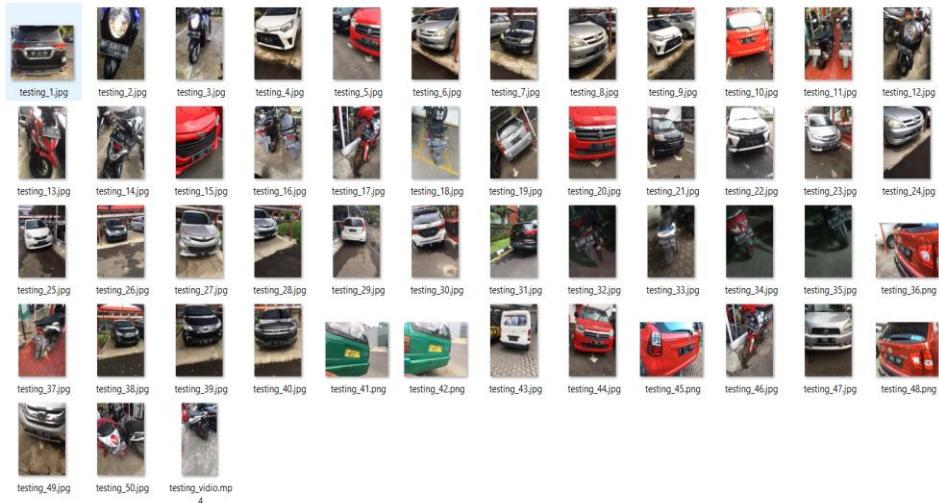
Berikut contoh dataset gambar yang dimiliki oleh penulis :



Gambar 5. 43 Data Train



Gambar 5. 44 Data Test



Gambar 5. 45 Data Pengujian

### 6.3.2 Pengolahan Data

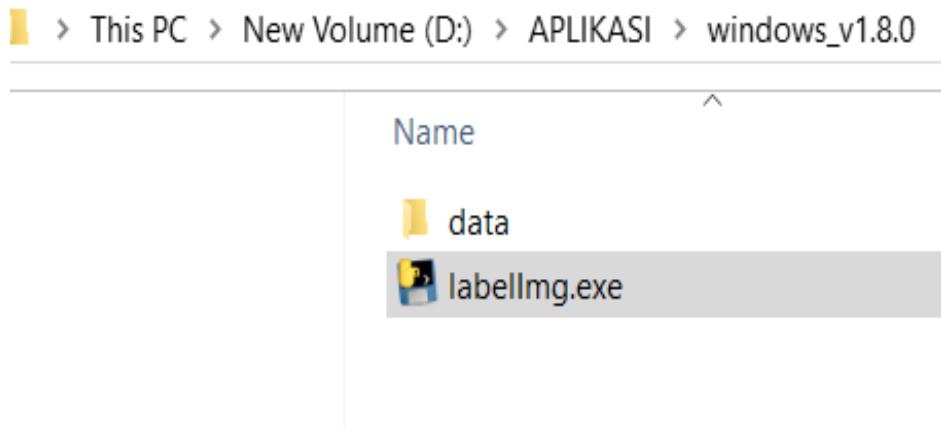
Selanjutnya adalah pengolahan data atau memberikan label pada gambar, Inilah bagian yang menyenangkan. Dengan semua gambar yang telah didapat dan dikumpulkan, inilah saatnya untuk memberi label objek

yang diinginkan pada setiap gambar. Pada proses ini kita menggunakan tools LabelImg, LabelImg adalah alat yang hebat untuk memberi label pada gambar, hasil dari labeling gambar disini berupa anotasi gambar yang nantinya akan digunakan untuk proses training.

Pada proses ini, penulis melakukan proses labeling image yaitu memberikan label pada objek yang diinginkan di setiap gambar dengan menggunakan aplikasi LabelImg, LabelImg adalah alat anotasi gambar grafis dan label kotak pembatas objek dalam sebuah gambar. LabelImg menyimpan file .xml yang berisi data label untuk setiap gambar. File .xml ini akan digunakan untuk menghasilkan TFRecords, yang merupakan salah satu input untuk pelatih TensorFlow. Setelah memberi label dan menyimpan setiap gambar, akan ada satu file .xml untuk setiap gambar di direktori.

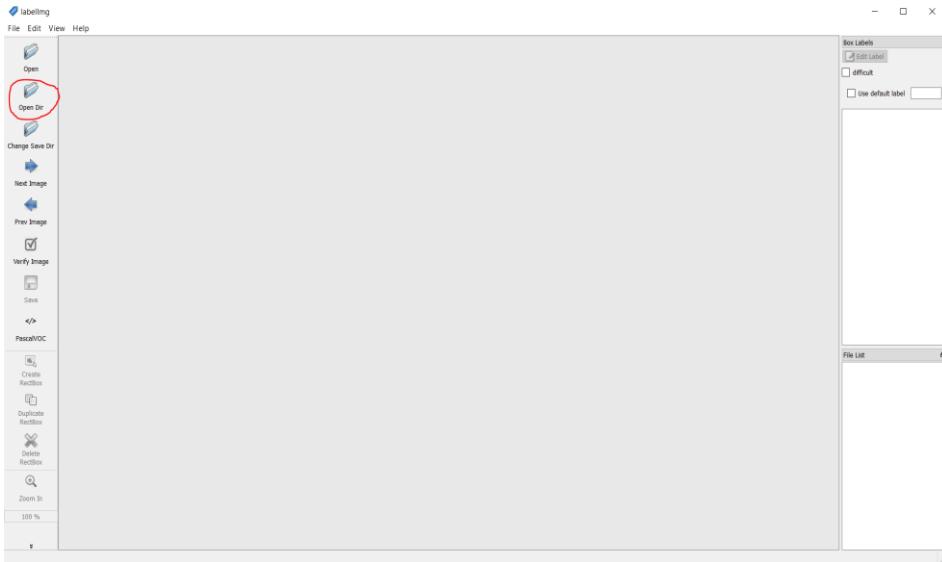
Untuk memuat annotation atau memberikan label pada gambar kita akan menggunakan aplikasi LabelImg yang nantinya akan disimpan kedalam file XML dengan format PASCAL VOC Cara membuat anotasinya :

- Buka aplikasi labelImg dengan melakukan double klik file .exe



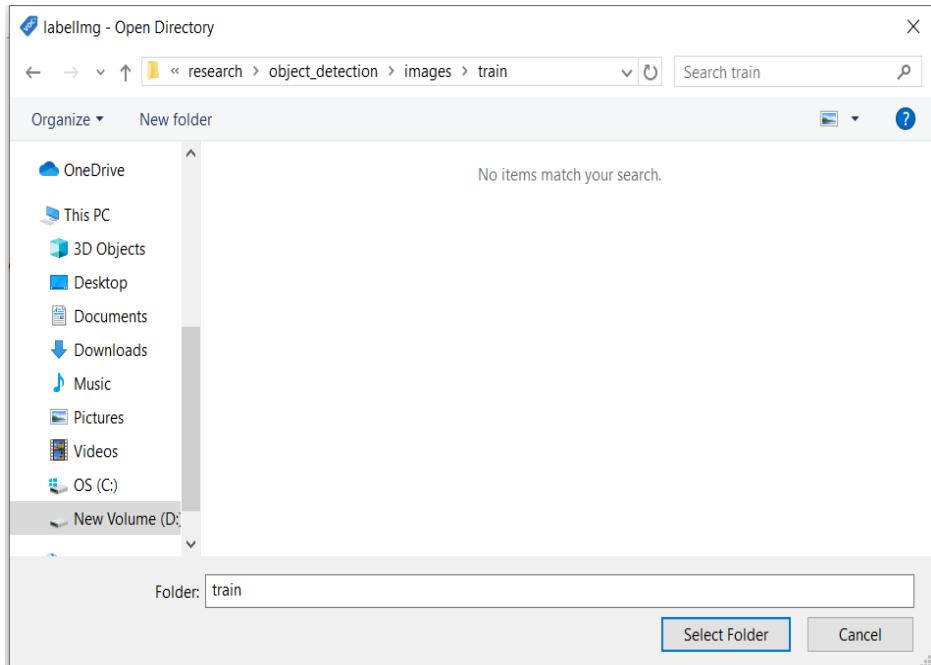
Gambar 6. 6 Buka aplikasi labelImg

- Klik tombol “OpenDir”



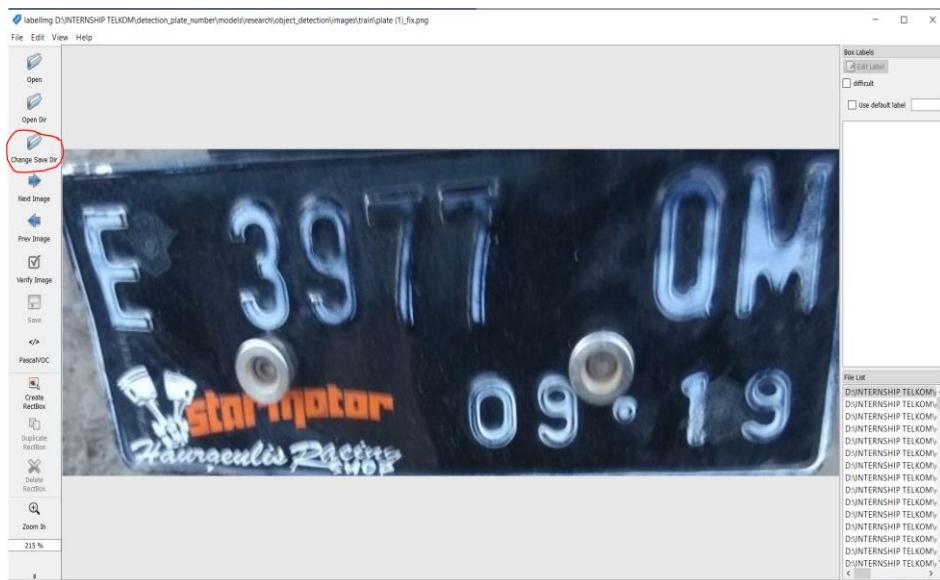
Gambar 6. 7 Klik tombol “OpenDir”

- Pilih direktori dataset gambar kemudian klik select folder



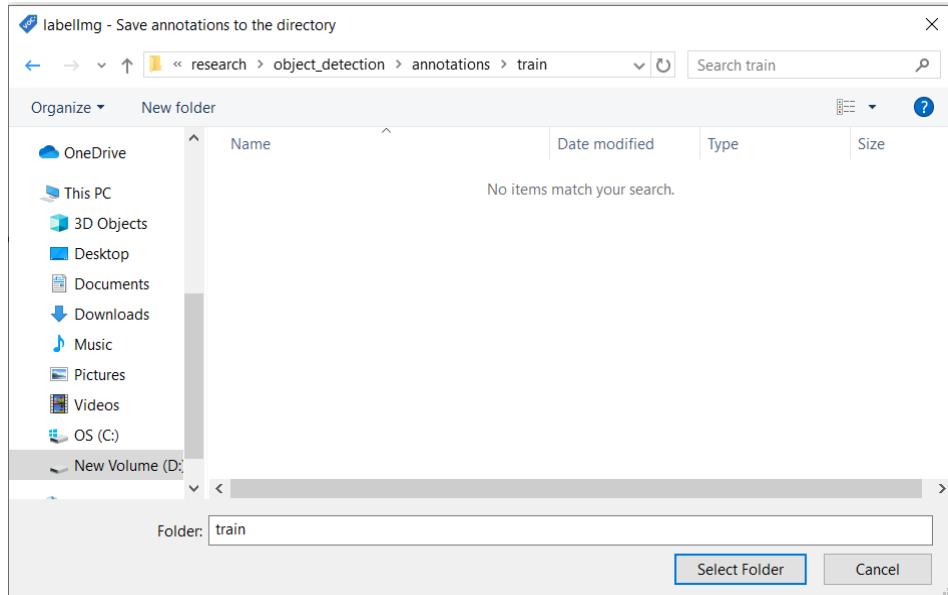
Gambar 6. 8 Pilih direktori dataset

- Klik Open Save Dir untuk mengatur direktori penyimpanan file anotasi .xml



Gambar 6. 9 Klik Open Save Dir

- Pilih direktori penyimpanan anotasi



Gambar 6. 10 Pilih direktori penyimpanan anotasi

- Klik tombol “Create RectBox” untuk membuat kotak area objek yang akan dikenali



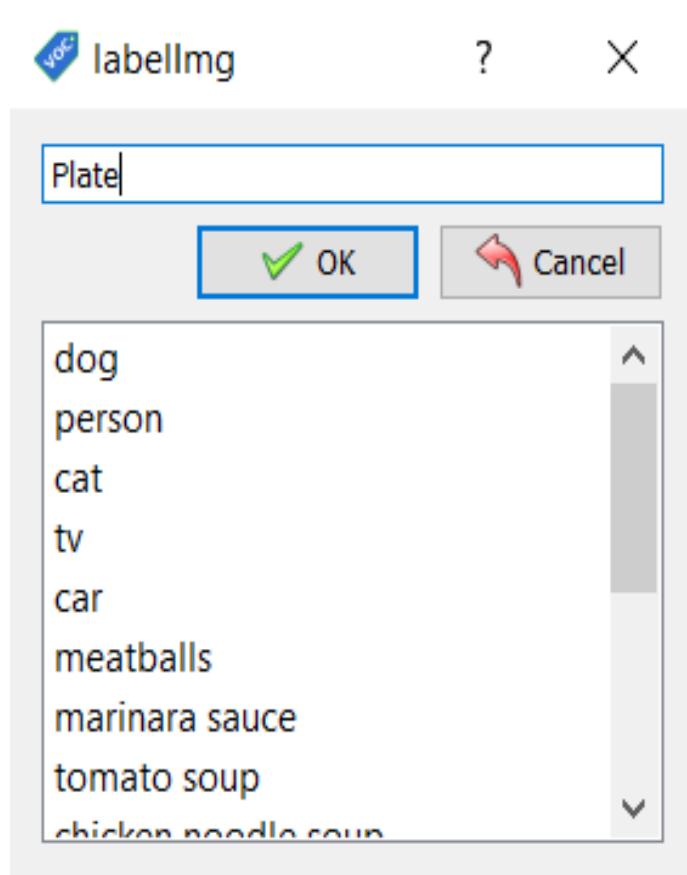
Gambar 6. 11 Klik tombol “Create RectBox”

- Arahkan kursos dan tarik area kotak disekitar objek



Gambar 6. 12 Arahkan kursos

- Lalu akan muncul kotak dialog untuk memberikan nama label dari objek yang akan kita kenali



Gambar 6. 13 Barikan Label

- Simpan hasil pelabelan dengan menekan tombol CTRL+S

Note: Nama label objek yang dikenali harus sama Case Sensitive. Simpan hasilnya kedalam direktori yang sama dengan data gambar.

LabelImg menyimpan file .xml yang berisi data label untuk setiap gambar. File .xml ini akan digunakan untuk menghasilkan TFRecords, yang merupakan salah satu input untuk pelatih TensorFlow. Setelah Anda memberi label dan menyimpan setiap gambar, akan ada satu file .xml

untuk setiap gambar di direktori \ test and \ train. Contoh dari file xml yang di dapat dilihat pada gambar 6.14.



```
<?xml version="1.0" ?>
<annotation>
<folder>None</folder>
<filename>plate (346)_fix.png</filename>
<path>fix/images/train/plate (346)_fix.png</path>
<source>
<database>Unknown</database>
</source>
<size>
<width>1478</width>
<height>1108</height>
<depth>3</depth>
</size>
<segmented/>
<object>
<name>plate</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>372</xmin>
<ymin>392</ymin>
<xmax>1005</xmax>
<ymax>652</ymax>
</bndbox>
</object>
</annotation>
```

Gambar 6. 14 Code Pada File xml

File xml ini berisi informasi yang dibawa dari gambar, yaitu ukuran gambar, kemudian class dari gambar yang diberi label, kemudian titik koordinat box yang ada di gambar

### 6.3.3 Convert XML ke CSV

Dengan gambar yang berlabel, saatnya untuk menghasilkan TFRecords yang berfungsi sebagai input data ke model pelatihan

TensorFlow. Tutorial ini menggunakan skrip `xml_to_csv.py` dan `generate_tfrecord.py` dari dataset Dat Tran's Raccoon Detector, dengan sedikit modifikasi untuk bekerja dengan struktur direktori penulis.

Dataset annotation yang kita buat diatas menggunakan aplikasi labelImg perlu dikonversi dari format .xml ke .csv yang nantinya akan digunakan untuk mengenerate berkas TFRecord, Pada proses ini dataset anotasi yang dibuat menggunakan aplikasi labelimg perlu dikonversi dari format .xml ke .csv yang nantinya akan digunakan untuk menggenerate berkas TFRecord. Untuk mengkonversi file .xml ke file .csv maka dibuat code program yang disimpan dengan nama file `xml_to_csv.py` yang berisi code untuk mengkonversi format berkas yang kita butuhkan. Berikut adalah kode program untuk mengkonversi format .xml ke .csv yang kita butuhkan :

Tabel 6. 2 Kode Program XML ke CSV

```
xml_to_csv.py
#
# untuk mengatur lokasi path
import os
# dalam script ini untuk membaca file xml dalam bentuk glob
import glob
# untuk mengolah data csv
import pandas as pd
# untuk mengolah data xml
import xml.etree.ElementTree as ET
def xml_to_csv(path):
    # inisialisasi list xml
    xml_list = []
```

```

# untuk setiap file xml dalam folder annotations lakukan langkah
berikut

for xml_file in glob.glob(path + '*.xml'):

    # baca file xml dari bagian terluar
    tree = ET.parse(xml_file)
    root = tree.getroot()

    # -----
    print(path)
    print(root.find('filename').text)

    # untuk setiap objek yg diinformasikan pada file xml lakukan
    langkah berikut

    for member in root.findall('object'):

        # dapatkan informasi nama kelas, ukuran image, dan lokasi box
        value = (root.find('filename').text,
                 int(root.find('size')[0].text),
                 int(root.find('size')[1].text),
                 member[0].text,
                 int(member[4][0].text),
                 int(member[4][1].text),
                 int(member[4][2].text),
                 int(member[4][3].text)
                 )

        # -----
        # masukan informasi objek kedalam list xml
        xml_list.append(value)
        # -----
# -----

```

```

# -----
# mendefinisikan column untuk file csv yg akan di generate
column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin',
'xmax', 'ymax']

# konversi list xml kedalam format table menggunakan pandas
xml_df = pd.DataFrame(xml_list, columns=column_name)

return xml_df

def main():

    for directory in ['train', 'test']:

        # mendapatkan list lokasi image (train/test)
        image_path = os.path.join(os.getcwd(),
'annotations/{}'.format(directory))

        # konversi informasi objek dari file xml & image,
        # untuk dicatat kedalam bentuk tabel yg nantinya akan disimpan
        dalam bentuk csv

        xml_df = xml_to_csv(image_path)

        # lakukan konversi kedalam csv lalu simpan
        xml_df.to_csv('data/{}_labels.csv'.format(directory),
index=None)

        # notif proses konversi xml to csv berhasil
        print('Successfully converted xml to csv.')

#untuk menjalankan program
main()

```

Berikut merupakan penjelasan dari kode pada tabel 6.2 :

1. Baris 1 import os berfungsi untuk mengatur lokasi path.
2. Baris 2 import glob dalam script ini untuk membaca file xml dalam bentuk glob.

3. Baris 3 import pandas as untuk mengolah data csv.
4. Baris 4 sampai 14 import xml.etree.ElementTree as ET berfungsi untuk mengolah data xml.
5. Baris 15 berfungsi untuk mendefinisikan column untuk file csv yg akan di generate.
6. Baris 16 berfungsi untuk konversi list xml kedalam format table menggunakan pandas.
7. Baris 20 berfungsi untuk mendapatkan list lokasi image (train/test).
8. Baris 21 berfungsi untuk konversi informasi objek dari file xml & image, untuk dicatat kedalam bentuk tabel yg nantinya akan disimpan dalam bentuk csv.
9. Baris 22 berfungsi untuk melakukan konversi kedalam csv lalu simpan.
10. Baris 23 berfungsi untuk notif proses konversi xml to csv berhasil.

Simpan kode program diatas dengan nama `xml_to_csv.py`. Lalu jalankan perintah dibawah ini untuk mengkonversi file XML ke CSV.

### Code

```
1 $ python3 xml_to_csv.py
```

Gambar 6. 15 Perintah xml to csv

```
D:\SEMESTER 7\object_detection\annotations\test
plate (87)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (88)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (89)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (90)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (91)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (92)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (93)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (94)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (95)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (96)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (97)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (98)_fix.png
D:\SEMESTER 7\object_detection\annotations\test
plate (99)_fix.png
Successfully converted xml to csv.
```

Gambar 6. 16 Proses xml ke csv

Convert file xml ke csv merupakan proses awal untuk melakukan proses training, data file csv yang didapat nantinya digunakan untuk proses selanjutnya yaitu generate TFRecord. Berikut adalah hasil dari konversi file xml ke csv pada data train dan data test dapat dilihat pada gambar 6.17 dan 6.18 :

test\_labels.csv - Excel

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	plate (1)_fix.png	490	310	plate	198	263	318	310
3	plate (10)_fix.png	700	465	plate	355	108	672	269
4	plate (100)_fix.png	629	328	plate	4	24	615	319
5	plate (101)_fix.png	502	223	plate	3	1	496	218
6	plate (102)_fix.png	453	182	plate	3	10	448	174
7	plate (103)_fix.png	304	123	plate	2	2	298	120
8	plate (104)_fix.png	489	219	plate	4	2	483	214
9	plate (105)_fix.png	537	211	plate	7	6	526	200
10	plate (106)_fix.png	121	69	plate	2	4	120	66
11	plate (11)_fix.png	512	288	plate	216	147	282	171
12	plate (111)_fix.png	512	288	plate	488	89	510	102
13	plate (12)_fix.png	435	300	plate	187	226	303	266
14	plate (13)_fix.png	168	300	plate	53	191	117	209
15	plate (14)_fix.png	251	200	plate	71	141	174	184
16	plate (15)_fix.png	275	183	plate	109	79	161	98
17	plate (16)_fix.png	300	225	plate	107	165	187	192
18	plate (17)_fix.png	300	168	plate	122	125	177	144
19	plate (18)_fix.png	1280	720	plate	182	155	1101	564
20	plate (19)_fix.png	1280	720	plate	350	253	964	505
21	plate (20)_fix.png	679	510	plate	361	240	571	306
22	plate (200)_fix.png	460	259	plate	45	172	130	237
23	plate (201)_fix.png	460	259	plate	299	139	317	160
24	plate (202)_fix.png	480	360	plate	69	110	361	226
25	plate (203)_fix.png	640	480	plate	207	325	408	396

Gambar 6. 17 File CSV Data Test

train\_labels.csv - Excel

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	plate (1)_fix.png	738	273	plate	2	3	718	266
3	plate (10)_fix.png	331	145	plate	4	2	326	137
4	plate (100)_fix.png	343	139	plate	6	3	336	134
5	plate (101)_fix.png	534	239	plate	7	5	524	239
6	plate (102)_fix.png	447	193	plate	10	4	440	189
7	plate (103)_fix.png	383	161	plate	3	7	381	155
8	plate (104)_fix.png	330	147	plate	6	3	321	143
9	plate (105)_fix.png	688	361	plate	13	4	678	358
10	plate (106)_fix.png	357	139	plate	7	2	352	136
11	plate (107)_fix.png	374	149	plate	3	2	369	146
12	plate (108)_fix.png	665	270	plate	5	4	650	264
13	plate (109)_fix.png	267	103	plate	3	1	264	101
14	plate (11)_fix.png	153	83	plate	2	3	152	80
15	plate (110)_fix.png	314	136	plate	1	3	310	133
16	plate (111)_fix.png	376	154	plate	2	1	370	150
17	plate (112)_fix.png	179	75	plate	2	1	177	73
18	plate (113)_fix.png	729	410	plate	10	5	718	402
19	plate (114)_fix.png	581	219	plate	7	3	570	213
20	plate (115)_fix.png	709	323	plate	10	6	698	315
21	plate (116)_fix.png	369	151	plate	5	6	366	143
22	plate (117)_fix.png	423	169	plate	4	3	421	163
23	plate (118)_fix.png	397	145	plate	9	1	391	143
24	plate (119)_fix.png	229	108	plate	3	1	225	106
25	plate (12)_fix.png	806	302	plate	12	5	798	295

Gambar 6. 18 File Csv Data Train

#### **6.3.4 Convert TFRecord**

Pada saat proses training, pertama-tama TensorFlow akan membaca data input dan proses ini dinamakan feeding data. Proses ini secara langsung mengambil informasi dari dataset yang telah disiapkan dalam format TFRecord, maka dari itu perlu dilakukan proses generate data annotation yang telah dikonversi ke .csv kedalam format TFRecord. Untuk meng-generate TFRecord dibuat code program yang disimpan dengan nama file generate\_tfrecord.py. Berikut ini adalah kode program untuk men-generate file TFRecord :

Tabel 6. 3 Kode Program Generate TFRecord

```
generate_tfrecord.py.

# Import Library
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import
# untuk mengatur lokasi path
import os
# dalam script ini digunakan untuk konversi image kedalam format byte
import io
# untuk mengolah csv
import pandas as pd
# untuk generate tfrecord
import tensorflow as tf
# untuk mengolah/read data image
from PIL import Image
# dalam script ini digunakan untuk transformasi setiap value yg akan
dijadikan tfrecord
```

```

from object_detection.utils import dataset_util
# untuk keperluan mengolah array, list, dict
from collections import namedtuple, OrderedDict
# parsing parameter dari command python (train/test, data csv, path
output untuk tfrecord)
flags = tf.app.flags
flags.DEFINE_string('type', "", 'Type of CSV input (train/test)')
flags.DEFINE_string('csv_input', "", 'Path to the CSV input')
flags.DEFINE_string('output_path', "", 'Path to output TFRecord')
FLAGS = flags.FLAGS
# -----
# Translasi nama kelas dalam bentuk text ke bentuk indeks integer
def class_text_to_int(row_label):
    if row_label == 'plate':
        return 1
    else:
        None
# -----
# untuk memisahkan data csv yg terbaca kedalam beberapa kolom
(delimiter)
def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]
# -----
# Fungsi untuk menghasilkan tf record

```

```

def create_tf_example(group, path):
    # membaca file image
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)),
    'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        #
        # mendapatkan ukuran image
        width, height = image.size
        # mendapatkan nama file dari image
        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        # inisialisasi list
        xmins = []
        xmaxs = []
        ymins = []
        ymaxs = []
        classes_text = []
        classes = []
        #
        # untuk setiap box pada image lakukan berikut,..
        for index, row in group.object.iterrows():
            # konversi setiap koordinat dari box dari pixel ke domain 0 - 1
            xmins.append(row['xmin'] / width)
            xmaxs.append(row['xmax'] / width)
            ymins.append(row['ymin'] / height)

```

```

ymins.append(row['ymin'] / height)
# -----
# Menyimpan nama kelas kedalam list dalam bentuk text
classes_text.append(row['class'].encode('utf8'))

# Menyimpan nama kelas kedalam list dalam bentuk index integer
classes.append(class_text_to_int(row['class']))

# -----
# konversi ke tfrecord menggunakan fungsi tensorflow
tf_example = tf.train.Example(features=tf.train.Features(feature={
    'image/height': dataset_util.int64_feature(height),
    'image/width': dataset_util.int64_feature(width),
    'image/filename': dataset_util.bytes_feature(filename),
    'image/source_id': dataset_util.bytes_feature(filename),
    'image/encoded': dataset_util.bytes_feature(encoded_jpg),
    'image/format': dataset_util.bytes_feature(image_format),
    'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
    'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
    'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
    'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
    'image/object/class/text':
        dataset_util.bytes_list_feature(classes_text),
    'image/object/class/label': dataset_util.int64_list_feature(classes),
}))
```

# -----

```

return tf_example
```

# -----

```

def main(_):
```

```

# menyiapkan variable untuk menyimpan tf record kedalam bentuk
file

writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
# inisialisasi path, apakah untuk path data train atau data test
path = os.path.join(os.getcwd(), 'images/{}'.format(FLAGS.type))
# membaca file csv
examples = pd.read_csv(FLAGS.csv_input)
# memisahkan setiap kolom yg terbaca dari data csv
grouped = split(examples, 'filename')

# mengolah tfrecord lalu menyimpannya kedalam bentuk file
.tfrecord
for group in grouped:
    tf_example = create_tf_example(group, path)
    writer.write(tf_example.SerializeToString())
writer.close()
# -----
# menampilkan lokasi file .tfrecord
output_path = os.path.join(os.getcwd(), FLAGS.output_path)
print('Successfully created the TFRecords: {}'.format(output_path))
# -----
if __name__ == '__main__':
    tf.app.run()

```

Berikut merupakan penjelasan dari kode pada tabel 6.3 :

1. Baris 1-3 berfungsi untuk import library.
2. Baris 4 berfungsi untuk mengatur lokasi path.

3. Baris 5 berfungsi untuk dalam script ini digunakan untuk konversi image kedalam format byte.
4. Baris 6 berfungsi untuk mengolah csv.
5. Baris 7 berfungsi untuk generate tfrecord.
6. Baris 8 berfungsi untuk mengolah/read data image.
7. Baris 9 berfungsi dalam script ini digunakan untuk transformasi setiap value yg akan dijadikan tfrecord.
8. Baris 10 berfungsi untuk keperluan mengolah array, list, dict.
9. Baris 11-15 berfungsi untuk parsing parameter dari command python (train/test, data csv, path output untuk tfrecord).
10. Baris 16-20 berfungsi untuk translasi nama kelas dalam bentuk text ke bentuk indeks integer.
11. Baris 21-24 berfungsi untuk memisahkan data csv yg terbaca kedalam beberapa kolom (delimiter).
12. Baris 25 berfungsi untuk menghasilkan tf record.
13. Baris 26-29 berfungsi untuk membaca file image.
14. Baris 30 berfungsi untuk mendapatkan ukuran image.
15. Baris 31-32 berfungsi untuk mendapatkan nama file dari image.
16. Baris 33-38 berfungsi untuk inisialisasi list.
17. Baris 39 berfungsi untuk setiap box pada image lakukan
18. Baris 40-43 berfungsi untuk konversi setiap koordinat dari box dari pixel ke domain 0 – 1.
19. Baris 44 berfungsi untuk menyimpan nama kelas kedalam list dalam bentuk text.
20. Baris 45 berfungsi untuk menyimpan nama kelas kedalam list dalam bentuk index integer.

21. Baris 46-59 berfungsi untuk konversi ke tfrecord menggunakan fungsi tensorflow.
22. Baris 62 berfungsi untuk menyiapkan variable untuk menyimpan tf record kedalam bentuk file.
23. Baris 63 berfungsi untuk inisialisasi path, apakah untuk path data train atau data test.
24. Baris 64 berfungsi untuk membaca file csv.
25. Baris 65 berfungsi untuk memisahkan setiap kolom yg terbaca dari data csv.
26. Baris 66-69 berfungsi untuk mengolah tfrecord lalu menyimpannya kedalam bentuk file .tfrecord.
27. Baris 70-71 berfungsi untuk menampilkan lokasi file .tfrecord.

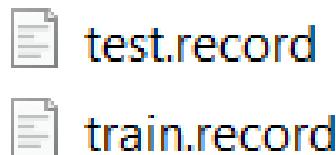
Pada code program TFRecord sesuaikan label map dengan class yang kita buat di mana objek dari class memiliki ID. Tugas nomor ID akan digunakan ketika mengkonfirmasi file labelmap.pbtxt. Kemudian hasilkan file TFRecord dengan mengeluarkan perintah-perintah ini dari folder \ object\_detection:

```
(tensorflow1) D:\SEMESTER 7\object_detection>
(tensorflow1) D:\SEMESTER 7\object_detection>python generate_tfrecord.py --type=train --csv_input=data/train_labels.csv --output_path=data/train.record
Successfully created the TFRecords: D:\SEMESTER 7\object_detection\data\train.record

(tensorflow1) D:\SEMESTER 7\object_detection>python generate_tfrecord.py --type=test --csv_input=data/test_labels.csv --output_path=data/test.record
Successfully created the TFRecords: D:\SEMESTER 7\object_detection\data\test.record
```

Gambar 6. 19 Proses Generate TFRecord

Generate file TFRecord dilakukan agar tensorflow dapat membaca data inputan dan mengambil informasi dari dataset yang telah disiapkan dalam format file TFRecord. Berikut adalah hasil dari generate TFRecord pada data train dan data test dapat dilihat pada gambar 6.20 :



Gambar 6. 20 Hasil Generate TFRecord

### 6.3.5 Membuat Label Map

Label Map adalah konfigurasi untuk memetakan label yang akan kita gunakan untuk memberikan penamaan pada objek yang akan kita deteksi. Jika objek yang akan kita deteksi ada lebih dari satu objek maka buat item sebanyak objek yang akan kita deteksi dan id maupun name harus menyesuaikan dokumentasi, namun pada kasus ini kita akan mendeteksi hanya untuk satu objek saja. Nomor ID peta label harus sama dengan apa yang didefinisikan dalam file generate\_tfrecord.py.

Label map memberi tahu saat proses training dari masing-masing objek dengan mendefenisikan pemetaan nama kelas ke nomor ID kelas. Jika objek yang akan dideteksi ada lebih dari satu objek maka buat item sebanyak objek yang akan dideteksi dan ID maupun name harus menyesuaikan dokumentasi. Pada penelitian ini penulis akan mendeteksi hanya untuk satu objek saja yaitu objek plate. Berikut kode program dari konfigurasi label map yang akan digunakan:

Tabel 6. 4 Kode Program Konfigurasi LabelMap

Labelmap.pbtxt
item { id: 1 name: 'plate' }

Berikut merupakan penjelasan dari kode pada tabel 6.4 :

1. Disini kita akan mengatur terlebih dahulu labelnya, pada table di atas terdapat 1 class yang diberi nama plate dan memiliki 1 ID. Peta label memberi tahu pelatih apa masing-masing objek dengan mendefinisikan pemetaan nama kelas ke nomor ID kelas.

### **6.3.6 Konfigurasi Pipeline**

Karena tensorflow menggunakan ProtoBuf maka kita perlu membuat konfigurasi pipeline dan kita akan menggunakan konfigurasi dari Faster Inception V2 yang digunakan oleh Oxford-IIIT untuk Dataset hewan peliharaan. Contoh konfigurasi dapat dilihat disini.

Konfigurasi pipeline adalah langkah terakhir sebelum menjalankan proses training untuk mendefinisikan model mana dan parameter apa yang akan digunakan untuk pelatihan. Pada penelitian ini penulis menggunakan metode CNN dengan model arsitektur Faster R-CNN dari hasil training coco dataset sebagai model training-nya, yang kemudian penulis mengubah code sesuai dengan kebutuhan penelitian.

Pada proses training objek deteksi pipeline harus dikonfigurasikan. Pipeline berfungsi untuk mendefinisikan model arsitektur apa dan parameter apa yang akan digunakan untuk proses training. Konfigurasi pipeline adalah langkah terakhir sebelum menjalankan proses training. Pada penelitian ini penulis menggunakan metode CNN dengan arsitektur Faster R-CNN dari pengujian data coco tahun 2018 sebagai model training-nya. Code program konfigurasi pipeline dapat dimanipulasi sesuai kebutuhan penulis dan disimpan dengan nama file plate\_number\_v3.config dengan mengatur resizer\_image, batch\_size = 1 dan step atau epoch = 100.000. Berikut adalah kode program dari konfigurasi pipeline yang digunakan :

Tabel 6. 5 Kode Konfigurasi Pipeline

plate_number_v3.config
#arsitektur / topologi objek deteksi yang dipakai model { faster_rcnn { #jumlah class num_classes: 1 #ukuran gambar yang akan di inputkan saat training image_resizer { keep_aspect_ratio_resizer { min_dimension: 600 max_dimension: 1024 } } #tahapan feature extraction feature_extractor { type: "faster_rcnn_inception_v2" #perpindahan stride first_stage_features_stride: 16 } first_stage_anchor_generator { grid_anchor_generator { height_stride: 16 width_stride: 16 scales: 0.25 scales: 0.5 scales: 1.0 } } } }

```

    scales: 2.0
    aspect_ratios: 0.5
    aspect_ratios: 1.0
    aspect_ratios: 2.0
}
}

first_stage_box_predictor_conv_hyperparams {
op: CONV #proses convolusi
regularizer {
l2_regularizer {
    weight: 0.0
}
}
initializer {
truncated_normal_initializer {
    stddev: 0.00999999977648 #perhitungan matematika stanser
deviasi untuk memanipulasi data (normalisasi)
}
}
}

}#untuk prediksi RPN I

first_stage_nms_score_threshold: 0.0
first_stage_nms_iou_threshold: 0.699999988079
first_stage_max_proposals: 100 #max proposal yang diajukan
sebagai candidat bounding box
first_stage_localization_loss_weight: 2.0
first_stage_objectness_loss_weight: 1.0
initial_crop_size: 14

```

```
maxpool_kernel_size: 2 #melakukan max pooling
maxpool_stride: 2
second_stage_box_predictor { #step 2 memprediksi box
mask_rcnn_box_predictor {
    fc_hyperparams {
        op: FC #fully connecter layer
        regularizer {
            l2_regularizer {
                weight: 0.0
            }
        }
    }
    initializer {
        variance_scaling_initializer {
            factor: 1.0
            uniform: true
            mode: FAN_AVG
        }
    }
}
use_dropout: false #untuk mengaktifkan dan nonaktifkan dropout
dropout_keep_probability: 1.0 #probability dropout
}
}
second_stage_post_processing {
batch_non_max_suppression {
score_threshold: 0.300000011921
iou_threshold: 0.600000023842
```

```
    max_detections_per_class: 100
    max_total_detections: 100
} #menggunakan activation softmax
score_converter: SOFTMAX
}
second_stage_localization_loss_weight: 2.0
second_stage_classification_loss_weight: 1.0
}
}

train_config {
batch_size: 1 #computer belajar 1 gambar itu satu kali
data_augmentation_options {
    random_horizontal_flip {
    }
}
optimizer {
    momentum_optimizer {
        learning_rate {
            manual_step_learning_rate { #konstanta training
                initial_learning_rate: 0.00019999994948
            schedule {
                step: 50000
                learning_rate: 0.00019999994948
            }
            schedule {
                step: 900000
                learning_rate: 1.9999994948e-05
            }
        }
    }
}
```

```
        }
      schedule {
        step: 1200000
        learning_rate: 1.9999999495e-06
      }
    }
  }

  momentum_optimizer_value: 0.899999976158
}

use_moving_average: false
}

gradient_clipping_by_norm: 10.0
fine_tune_checkpoint:
"faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt"
from_detection_checkpoint: true
num_steps: 100000 #banyaknya step yang kita lakukan
}

train_input_reader {
  label_map_path: "data/plate_number_map.pbtxt"
  tf_record_input_reader {
    input_path: "data/train.record"
  }
}

eval_config {
  num_examples: 8000
  max_evals: 10
  use_moving_averages: false
}
```

```
}
```

```
eval_input_reader {
```

```
    label_map_path: "data/plate_number_map.pbtxt"
```

```
    shuffle: false
```

```
    num_readers: 1
```

```
    tf_record_input_reader {
```

```
        input_path: "data/test.record"
```

```
    }
```

```
}
```

Berikut merupakan penjelasan dari kode pada tabel 6.5 :

1. Baris 2 - 3 arsitektur / topologi objek deteksi yang dipakai yaitu faster rcnn.
2. Baris 5 jumlah class
3. Baris 7 - 10 untuk resize ukuran gambar yang akan di inputkan saat training
4. Baris 14 - 15 tahapan feature extraction menggunakan faster rcnn
5. Baris 17 perpindahan stride 16
6. Baris 33 proses convolusi
7. Baris 39 - 41 perhitungan matematika stanser deviasi untuk memanipulasi data (normalisasi)
8. Baris 45 dan seterusnya untuk prediksi RPN
9. Baris 47 max proposal yang diajukan sebagai candidat bounding box
10. Baris 51 – 52 melakukan max pooling
11. Baris 53 step 2 memprediksi box
12. Baris 56 fully connecter layer
13. Baris 70 untuk mengaktifkan dan nonaktifkan dropout
14. Baris 81 menggunakan activation softmax

15. Baris 88 computer belajar 1 gambar itu satu kali
16. Baris 96 untuk konstanta training
17. Baris 119 banyaknya step yang kita lakukan

Simpan file konfigurasi diatas dengan nama plate\_number\_v3.config kedalam folder training. Konfigurasi diatas menggunakan batch\_size=1 dan num\_steps=100000 pada komputer penulis dengan spesifikasi 5 Core CPU, 2 GPU, dan RAM 4GB, nilai tersebut cukup berhasil untuk mendeteksi objek Plat Nomor, tapi sebaiknya nilai tersebut disesuaikan dengan spesifikasi komputer yang anda gunakan. Jika komputer yang anda gunakan mempunyai spesifikasi dibawah komputer yang penulis gunakan maka direkomendasikan untuk menurunkan nilai batch\_size setidaknya batch\_size=2 dan num\_steps=140000. Sampai ditahap ini kita sudah selesai menyiapkan semua file yang dibutuhkan pada saat training model object detection kita.

Note: Pastikan struktur direktori dan file dengan benar.

Berikut susunan akhir direktori yang penulis gunakan :

```

Code
1
2   annotations
3     test
4       IMG-20180108-WA0115.xml
5       ...
6       IMG-20180108-WA0120.xml
7       IMG-20180108-WA0121.xml
8     train
9       100.E 6467 QW-09-20.xml
10      ...
11      255.E 6527 OD-02-20.xml
12      256.E 4572 SS-01-16.xml
13
14   data
15     plate_number_map.pbtxt
16     test.record
17     test_labels.csv
18     train.record
19     train_labels.csv
20
21   images
22     test
23       IMG-20180108-WA0115.jpeg
24       ...
25       IMG-20180108-WA0120.jpeg
26       IMG-20180108-WA0121.jpeg
27     train
28       100.E 6467 QW-09-20.jpeg
29       ...
30       255.E 6527 OD-02-20.jpeg
31       256.E 4572 SS-01-16.jpeg
32
33   training
34     plate_number_v1.config
35     generate_tfrecord.py
36     xml_to_csv.py
37
38 8 directories, 1016 files

```

Gambar 6. 21 Susunan Direktori

### 6.3.7 Training Model

Setelah semua proses dilakukan dan semua file yang dibutuhkan sudah selesai dipersiapkan maka selanjutnya memasuki tahap training Neural Network untuk mengenali pola objek deteksi plat nomor kendaraan. Untuk melakukan proses training dibuat code program dengan nama file train.py yang akan dipanggil ketika proses training dijalankan. Kode program dari train.py yang digunakan pada saat training data dapat dilihat pada table 5.6:

Tabel 6. 6 Kode Program Train.py

train.py
#melakukan import import functools import json import os import tensorflow as tf from object_detection.builders import dataset_builder from object_detection.builders import graph_rewriter_builder from object_detection.builders import model_builder from object_detection.legacy import trainer from object_detection.utils import config_util tf.logging.set_verbosity(tf.logging.INFO) # parsing parameter command flags = tf.app.flags flags.DEFINE_string('master', "", 'Name of the TensorFlow master to use.') flags.DEFINE_integer('task', 0, 'task id')

```

flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per
worker.')

flags.DEFINE_boolean('clone_on_cpu', False,
    'Force clones to be deployed on CPU. Note that even if '
    'set to False (allowing ops to run on gpu), some ops may '
    'still be run on the CPU if they have no GPU kernel.')

flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer
'

'replicas.')

flags.DEFINE_integer('ps_tasks', 0,
    'Number of parameter server tasks. If None, does not use '
    'a parameter server.')

flags.DEFINE_string('train_dir', '',
    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
    'Path to a train_pb2.TrainConfig config file.')

flags.DEFINE_string('input_config_path', '',
    'Path to an input_reader_pb2.InputReader config file.')

flags.DEFINE_string('model_config_path', '',
    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS

# -----

```

```

@tf.contrib.framework.deprecated(None,                                     'Use
object_detection/model_main.py.')
def main(_):
    # mengecek apakah direktory training ada atau tidak
    assert FLAGS.train_dir, "train_dir` is missing."
    # jika flag task adalah 0, buat direktory training
    if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
    # Jika path .config di sebutkan di command parameter maka,
    # akan membaca file .config kedalam tensorflow
    if FLAGS.pipeline_config_path:
        configs = config_util.get_configs_from_pipeline_file(
            FLAGS.pipeline_config_path)
        if FLAGS.task == 0:
            tf.gfile.Copy(FLAGS.pipeline_config_path,
                          os.path.join(FLAGS.train_dir, 'pipeline.config'),
                          overwrite=True)

    # -----
    # Jika path .config tidak di command parameter maka,
    # gunakan file .config yg sudah disediakan tensorflow
    else:
        configs = config_util.get_configs_from_multiple_files(
            model_config_path=FLAGS.model_config_path,
            train_config_path=FLAGS.train_config_path,
            train_input_config_path=FLAGS.input_config_path)
    if FLAGS.task == 0:
        for name, config in [('model.config', FLAGS.model_config_path),
                             ('train.config', FLAGS.train_config_path)],

```

```

        ('input.config', FLAGS.input_config_path)]:
    tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                  overwrite=True)

# -----
# mendapatkan informasi dari file config yg telah di read
model_config = configs['model']
train_config = configs['train_config']
input_config = configs['train_input_config']
# -----
# inisialisasi model training
model_fn = functools.partial(
    model_builder.build,
    model_config=model_config,
    is_training=True)
# -----
# inisialisasi iterator data training
def get_next(config):
    return dataset_builder.make_initializable_iterator(
        dataset_builder.build(config)).get_next()
create_input_dict_fn = functools.partial(get_next, input_config)
# -----
# membaca file .config dalam bentuk json
env = json.loads(os.environ.get('TF_CONFIG', '{}'))
# mendapatkan data cluster dari konfigurasi training
cluster_data = env.get('cluster', None)
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else None
# -----

```

```

# mendapatkan task dari konfigurasi training
task_data = env.get('task', None) or {'type': 'master', 'index': 0}
task_info = type('TaskSpec', (object,), task_data)
# -----
# Menyiapkan variabel untuk keperluan training
# Parameters for a single worker.

ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ""

if cluster_data and 'worker' in cluster_data:
    # Number of total worker replicas include "worker"s and the "master".
    worker_replicas = len(cluster_data['worker']) + 1
if cluster_data and 'ps' in cluster_data:
    ps_tasks = len(cluster_data['ps'])
if worker_replicas > 1 and ps_tasks < 1:
    raise ValueError('At least 1 ps task is needed for distributed training.')
if worker_replicas >= 1 and ps_tasks > 0:
    # Set up distributed training.
    server = tf.train.Server(tf.train.ClusterSpec(cluster), protocol='grpc',
                            job_name=task_info.type,
                            task_index=task_info.index)
    if task_info.type == 'ps':
        server.join()
return

```

```

worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
task = task_info.index
is_chief = (task_info.type == 'master')
master = server.target
graph_rewriter_fn = None
if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=True)

# -----
# Lakukan proses training
trainer.train(
    create_input_dict_fn,
    model_fn,
    train_config,
    master,
    task,
    FLAGS.num_clones,
    worker_replicas,
    FLAGS.clone_on_cpu,
    ps_tasks,
    worker_job_name,
    is_chief,
    FLAGS.train_dir,
    graph_hook_fn=graph_rewriter_fn)

# -----
# Jalankan script ini melalui fungsi main berikut
if __name__ == '__main__':

```

```
tf.app.run()  
# -----
```

Berikut merupakan penjelasan dari kode pada tabel 5.6 :

1. Baris 1 dan seterusnya berfungsi untuk melakukan import.
2. Baris 11 dan seterusnya berfungsi untuk melakukan parsing parameter command.
3. Baris 26 berfungsi untuk mengecek apakah direktori training ada atau tidak.
4. Baris 27 berarti jika flag task adalah 0, buat direktori training.
5. Baris 28 dan seterusnya Jika path .config di sebutkan di command parameter maka, akan membaca file .config kedalam tensorflow.
6. Baris 35 dan seterusnya berfungsi untuk mendapatkan informasi dari file config yg telah di read.
7. Baris 38 dan seterusnya berfungsi untuk inisialisasi model training.
8. Baris 43 dan seterusnya berfungsi untuk inisialisasi iterator data training.
9. Baris 47 berfungsi untuk membaca file .config dalam bentuk json.
10. Baris 48 berfungsi untuk mendapatkan data cluster dari konfigurasi training.
11. Baris 50 berfungsi untuk mendapatkan task dari konfigurasi training.
12. Baris 52 dan seterusnya berfungsi untuk Menyiapkan variabel untuk keperluan training, Parameters for a single worker.
13. Baris 59 dan seterusnya berfungsi untuk jumlah dari total worker, replicas yang termasuk “workers” dan “master”.
14. Baris 65 dan selanjutnya berfungsi untuk Set up distributed training.
15. Baris 78 berfungsi untuk Lakukan proses training.

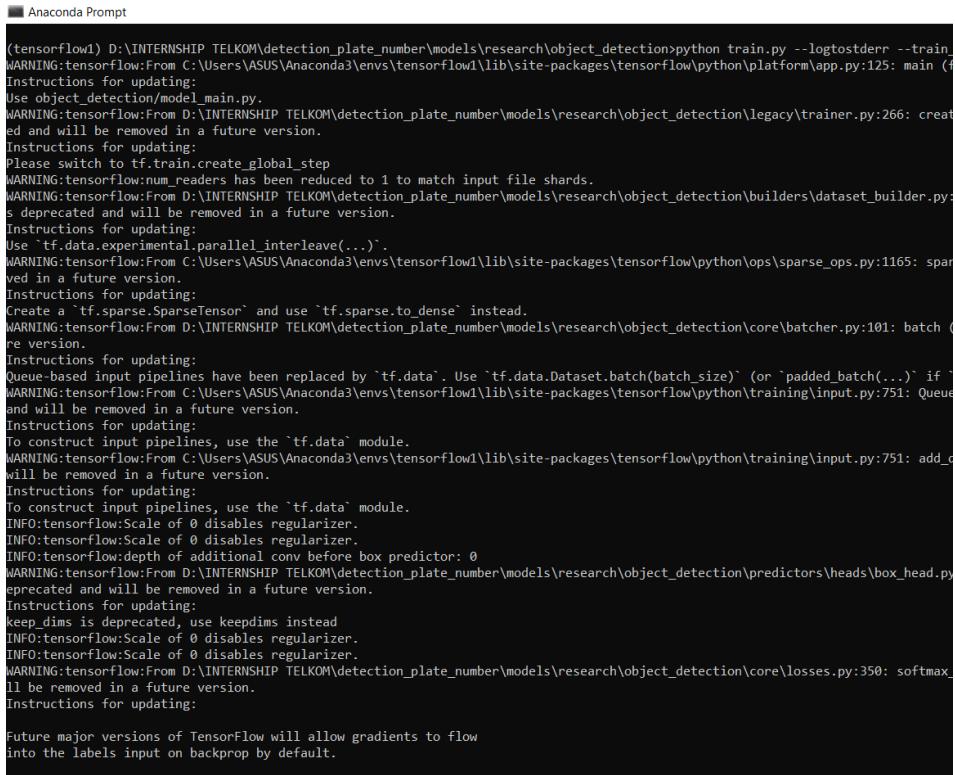
16. Baris 93 berfungsi untuk Jalankan script ini melalui fungsi main berikut if `__name__ == '__main__'`: `tf.app.run()`.

Mari kita mulai training Neural Network dengan menjalankan perintah dibawah ini pada anaconda command prompt untuk memulai proses training :

```
(tensorflow1) D:\SEMESTER 7\object_detection>python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/plate_number_v3.config
```

Gambar 6. 22 Perintah Training Data

Jika semuanya telah diatur dengan benar, Tensorflow akan menginisialisasi pelatihan. Inisialisasi dapat memakan waktu hingga 30 detik sebelum pelatihan sebenarnya dimulai. Ketika pelatihan dimulai akan terlihat seperti gambar berikut:



The screenshot shows the Anaconda Prompt window with the title 'Anaconda Prompt'. The terminal output displays the command 'python train.py' being run within a TensorFlow environment. The logs include several 'WARNING' and 'INFO' messages related to deprecated TensorFlow code, such as 'tf.train.create\_global\_step', 'tf.data.experimental.parallel\_interleave', and various 'tf.sparse' operations. These messages indicate that the code is being executed in an older TensorFlow version and will be removed in future updates. The logs also mention 'tf.data.Dataset.batch' and 'tf.data.Dataset.padded\_batch' as replacements for older queue-based input pipelines.

```
(tensorflow1) D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection>python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/plate_number_v3.config
WARNING:tensorflow:From C:\Users\ASUS\Anaconda3\envs\tensorflow1\lib\site-packages\tensorflow\python\platform\app.py:125: main (f
Instructions for updating:
Use object_detection/model_main.py.
WARNING:tensorflow:From D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\legacy\trainer.py:266: creat
ed and will be removed in a future version.
Instructions for updating:
Please switch to tf.train.create_global_step
WARNING:tensorflow:num readers has been reduced to 1 to match input file shards.
WARNING:tensorflow:From D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\builders\dataset_builder.py:
s deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.data.experimental.parallel_interleave(...)`.
WARNING:tensorflow:From C:\Users\ASUS\Anaconda3\envs\tensorflow1\lib\site-packages\tensorflow\python\ops\sparse_ops.py:1165: spar
ved in a future version.
Instructions for updating:
Create a `tf.sparse.SparseTensor` and use `tf.sparse.to_dense` instead.
WARNING:tensorflow:From D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\core\batcher.py:101: batch (
re version.
Instructions for updating:
Queue-based input pipelines have been replaced by `tf.data`. Use `tf.data.Dataset.batch(batch_size)` (or `padded_batch(...)` if
WARNING:tensorflow:From C:\Users\ASUS\Anaconda3\envs\tensorflow1\lib\site-packages\tensorflow\python\training\input.py:751: Queue
and will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
WARNING:tensorflow:From C:\Users\ASUS\Anaconda3\envs\tensorflow1\lib\site-packages\tensorflow\python\training\input.py:751: add_o
will be removed in a future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:depth of additional conv before box predictor: 0
WARNING:tensorflow:From D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\predictors\heads\box_head.py:
deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
INFO:tensorflow:Scale of 0 disables regularizer.
INFO:tensorflow:Scale of 0 disables regularizer.
WARNING:tensorflow:From D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\core\losses.py:350: softmax_
ll be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.
```

Gambar 6. 23 Proses Sebelum Training Dimulai

```
[■ Anaconda Prompt]
I and will be removed in a future version.
INFO:tensorflow:Recording summary at step 0
INFO:tensorflow:Recording summary at step 0
INFO:tensorflow:global step 1: loss = 1.712 (57.643 sec/step)
INFO:tensorflow:global step 2: loss = 1.0447 (1.280 sec/step)
INFO:tensorflow:global step 3: loss = 1.0447 (1.280 sec/step)
INFO:tensorflow:global step 4: loss = 1.0223 (1.280 sec/step)
INFO:tensorflow:global step 5: loss = 1.0223 (1.280 sec/step)
INFO:tensorflow:global step 6: loss = 0.9713 (6.513 sec/step)
INFO:tensorflow:global step 7: loss = 0.9517 (5.022 sec/step)
INFO:tensorflow:global step 8: loss = 0.9517 (5.022 sec/step)
INFO:tensorflow:global step 9: loss = 1.1797 (1.180 sec/step)
INFO:tensorflow:global step 10: loss = 0.8856 (4.581 sec/step)
INFO:tensorflow:global step 11: loss = 0.8856 (4.581 sec/step)
INFO:tensorflow:global step 12: loss = 0.8703 (1.135 sec/step)
INFO:tensorflow:global step 13: loss = 0.8703 (1.135 sec/step)
INFO:tensorflow:global step 14: loss = 0.8444 (1.135 sec/step)
INFO:tensorflow:global step 15: loss = 0.7528 (1.113 sec/step)
INFO:tensorflow:global step 16: loss = 0.7311 (4.714 sec/step)
INFO:tensorflow:global step 17: loss = 0.7308 (1.120 sec/step)
INFO:tensorflow:global step 18: loss = 0.7311 (4.714 sec/step)
INFO:tensorflow:global step 19: loss = 0.7308 (1.120 sec/step)
INFO:tensorflow:global step 20: loss = 0.7364 (1.142 sec/step)
INFO:tensorflow:global step 21: loss = 0.7205 (3.469 sec/step)
INFO:tensorflow:global step/acc: 0.233829
INFO:tensorflow:Recording summary at step 20
INFO:tensorflow:Recording summary at step 21
INFO:tensorflow:global step 22: loss = 0.7092 (3.910 sec/step)
```

Gambar 6. 24 Proses Sebelum Training

Dapat dilihat proses training menggunakan GPU Geforce MX230, setelah proses sebelum training sebenarnya dimulai maka selanjutnya proses training dimulai berikut tampilan dari training yang dilakukan :

```
[■ Anaconda Prompt]
I and there could be performance gains if more memory were available.
INFO:tensorflow:Recording summary at step 0
INFO:tensorflow:Recording summary at step 0
INFO:tensorflow:global step 1: loss = 1.712 (57.643 sec/step)
INFO:tensorflow:global step 2: loss = 1.0447 (1.280 sec/step)
INFO:tensorflow:global step 3: loss = 1.0447 (1.280 sec/step)
INFO:tensorflow:global step 4: loss = 1.0223 (1.280 sec/step)
INFO:tensorflow:global step 5: loss = 1.0223 (1.280 sec/step)
INFO:tensorflow:global step 6: loss = 0.9713 (6.513 sec/step)
INFO:tensorflow:global step 7: loss = 0.9517 (5.022 sec/step)
INFO:tensorflow:global step 8: loss = 0.9517 (5.022 sec/step)
INFO:tensorflow:global step 9: loss = 1.1797 (1.180 sec/step)
INFO:tensorflow:global step 10: loss = 0.8856 (4.581 sec/step)
INFO:tensorflow:global step 11: loss = 0.8856 (4.581 sec/step)
INFO:tensorflow:global step 12: loss = 0.8703 (1.135 sec/step)
INFO:tensorflow:global step 13: loss = 0.8703 (1.135 sec/step)
INFO:tensorflow:global step 14: loss = 0.8444 (1.135 sec/step)
INFO:tensorflow:global step 15: loss = 0.7528 (1.113 sec/step)
INFO:tensorflow:global step 16: loss = 0.7311 (4.714 sec/step)
INFO:tensorflow:global step 17: loss = 0.7308 (1.120 sec/step)
INFO:tensorflow:global step 18: loss = 0.7311 (4.714 sec/step)
INFO:tensorflow:global step 19: loss = 0.7308 (1.120 sec/step)
INFO:tensorflow:global step 20: loss = 0.7364 (1.142 sec/step)
INFO:tensorflow:global step 21: loss = 0.7205 (3.469 sec/step)
INFO:tensorflow:global step/acc: 0.233829
INFO:tensorflow:Recording summary at step 20
INFO:tensorflow:Recording summary at step 21
INFO:tensorflow:global step 22: loss = 0.7092 (3.910 sec/step)
```

Gambar 6. 25 Proses Training Data

Proses di atas akan memakan waktu cukup lama. Setiap langkah melaporkan loss. Loss akan dimulai dari tinggi kemudian akan semakin rendah dan semakin rendah saat training berlangsung. Untuk training pada

penelitian ini loss dimulai sekitar 1.1 dengan cepat turun di bawah 0.9. Biarkan proses training berlatih sampai loss turun di bawah 0.05 yang akan memakan waktu sesuai step atau epoch yang telah diatur pada konfigurasi pipeline yaitu 100.000 epoch. Kemudian setiap step yang sudah di training akan di recording dan tersimpan dalam folder training, sehingga apabila proses training terjeda dan ingin melanjutkan maka tidak akan training ulang dari awal melainkan melanjutkan proses training yang sebelumnya.

Proses training yang sudah selesai dapat dilihat pada gambar 6.23:

```
■ Anaconda Prompt
INFO:tensorflow:global step 99980: loss = 0.0274 (0.766 sec/step)
INFO:tensorflow:global step 99980: loss = 0.0274 (0.766 sec/step)
INFO:tensorflow:global step 99981: loss = 0.0097 (0.682 sec/step)
INFO:tensorflow:global step 99981: loss = 0.0097 (0.682 sec/step)
INFO:tensorflow:global step 99982: loss = 0.0795 (0.774 sec/step)
INFO:tensorflow:global step 99982: loss = 0.0795 (0.774 sec/step)
INFO:tensorflow:global step 99983: loss = 0.0311 (1.089 sec/step)
INFO:tensorflow:global step 99983: loss = 0.0311 (1.089 sec/step)
INFO:tensorflow:Recording summary at step 99983.
INFO:tensorflow:Recording summary at step 99983.
INFO:tensorflow:global step 99984: loss = 0.0677 (0.814 sec/step)
INFO:tensorflow:global step 99984: loss = 0.0677 (0.814 sec/step)
INFO:tensorflow:global step 99985: loss = 0.0465 (0.812 sec/step)
INFO:tensorflow:global step 99985: loss = 0.0465 (0.812 sec/step)
INFO:tensorflow:global step 99986: loss = 0.0377 (0.763 sec/step)
INFO:tensorflow:global step 99986: loss = 0.0377 (0.763 sec/step)
INFO:tensorflow:global step 99987: loss = 0.0161 (0.763 sec/step)
INFO:tensorflow:global step 99987: loss = 0.0161 (0.763 sec/step)
INFO:tensorflow:global step 99988: loss = 0.0232 (0.853 sec/step)
INFO:tensorflow:global step 99988: loss = 0.0232 (0.853 sec/step)
INFO:tensorflow:global step 99989: loss = 0.0248 (0.632 sec/step)
INFO:tensorflow:global step 99989: loss = 0.0248 (0.632 sec/step)
INFO:tensorflow:global step 99990: loss = 0.0201 (0.704 sec/step)
INFO:tensorflow:global step 99990: loss = 0.0201 (0.704 sec/step)
INFO:tensorflow:global step 99991: loss = 0.0130 (0.774 sec/step)
INFO:tensorflow:global step 99991: loss = 0.0130 (0.774 sec/step)
INFO:tensorflow:global step 99992: loss = 0.0574 (0.774 sec/step)
INFO:tensorflow:global step 99992: loss = 0.0574 (0.774 sec/step)
INFO:tensorflow:global step 99993: loss = 0.0265 (0.767 sec/step)
INFO:tensorflow:global step 99993: loss = 0.0265 (0.767 sec/step)
INFO:tensorflow:global step 99994: loss = 0.0788 (0.773 sec/step)
INFO:tensorflow:global step 99994: loss = 0.0788 (0.773 sec/step)
INFO:tensorflow:global step 99995: loss = 0.0535 (0.843 sec/step)
INFO:tensorflow:global step 99995: loss = 0.0535 (0.843 sec/step)
INFO:tensorflow:global step 99996: loss = 0.0328 (0.801 sec/step)
INFO:tensorflow:global step 99996: loss = 0.0328 (0.801 sec/step)
INFO:tensorflow:global step 99997: loss = 0.0340 (0.692 sec/step)
INFO:tensorflow:global step 99997: loss = 0.0340 (0.692 sec/step)
INFO:tensorflow:global step 99998: loss = 0.0549 (0.684 sec/step)
INFO:tensorflow:global step 99998: loss = 0.0549 (0.684 sec/step)
INFO:tensorflow:global step 99999: loss = 0.0390 (0.723 sec/step)
INFO:tensorflow:global step 99999: loss = 0.0390 (0.723 sec/step)
INFO:tensorflow:global step 100000: loss = 0.0198 (0.764 sec/step)
INFO:tensorflow:global step 100000: loss = 0.0198 (0.764 sec/step)
INFO:tensorflow:Stopping Training.
INFO:tensorflow:Stopping Training.
INFO:tensorflow:Finished training! Saving model to disk.
INFO:tensorflow:Finished training! Saving model to disk.
```

Gambar 6. 26 Proses Training Selesai

Training neural network dilakukan untuk mengenali pola plat nomor kendaraan. Pada penelitian ini penulis melakukan training dengan 100000 epoch atau step yaitu komputer akan belajar 100000x untuk mengenali

objek plat nomor pada proses training. Setiap step hasil training akan tersimpan pada folder training dalam bentuk model checkpoint meta, data dan index yang di dalamnya berisi informasi saat melakukan training. Berikut adalah checkpoint hasil training yang sudah selesai dengan 100000 epoch pada gambar 6.27:

- ❑ model.ckpt-100000.meta
- ❑ model.ckpt-100000.index
- ❑ model.ckpt-100000.data-00000-of-00001

Gambar 6. 27 Checkpoint Training

### 6.3.8 Grafik TensorBoard

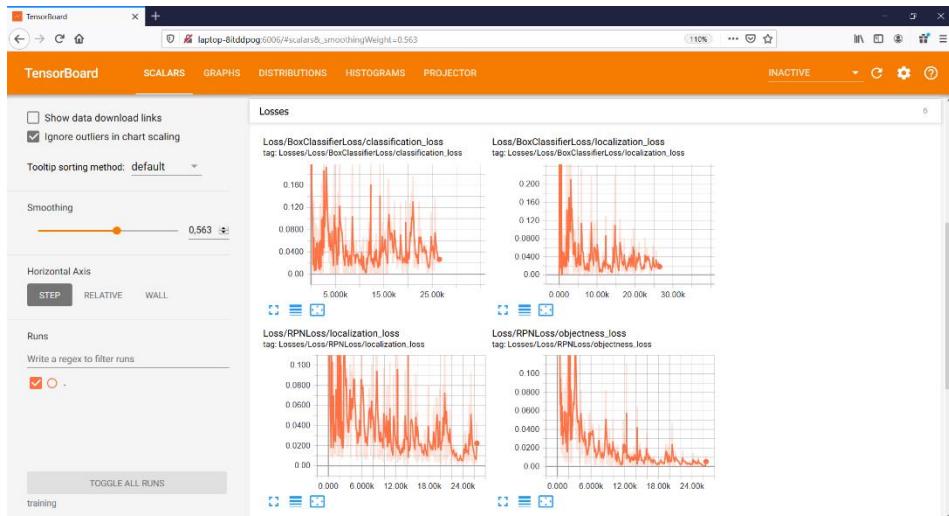
Setelah melakukan proses training data proses kemajuan training dapat dilihat dengan TensorBoard. Peneliti akan menggunakan TensorBoard untuk memantau kemajuan training data dan melakukan visualisasi pada saat training. Untuk melihat TensorBoard jalankan perintah seperti berikut:

```
(tensorflow1) D:\SEMESTER 7\object_detection>tensorboard --logdir=training
```

Gambar 6. 28 Perintah Tensorboard

Printah ini akan membuat halaman web pada mesin lokal di YourPCName: 6006, yang dapat dilihat melalui browser web. Tensorboard adalah sebuah library pada tensorflow yang dapat digunakan untuk melihat grafik dari proses training yang dilakukan Halaman TensorBoard memberikan informasi dan grafik yang menunjukkan kemajuan dari proses training. Satu grafik penting adalah grafik Loss, yang menunjukkan Loss keseluruhan dari pengklasifikasi dari waktu ke waktu. Berikut adalah grafik loss pada proses training yang sudah selesai dengan 100.000 step:

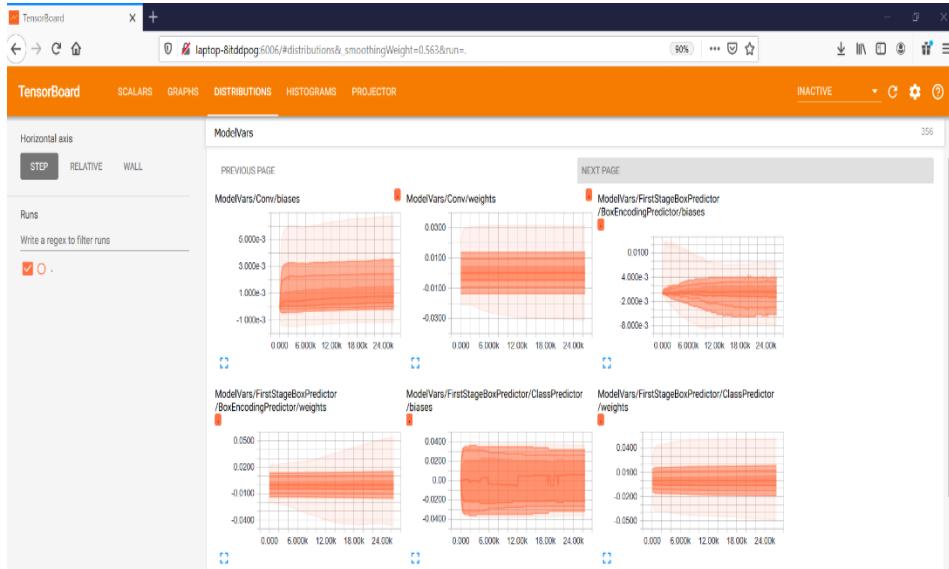
## 1. Grafik Loss



Gambar 6. 29 Grafik Loss

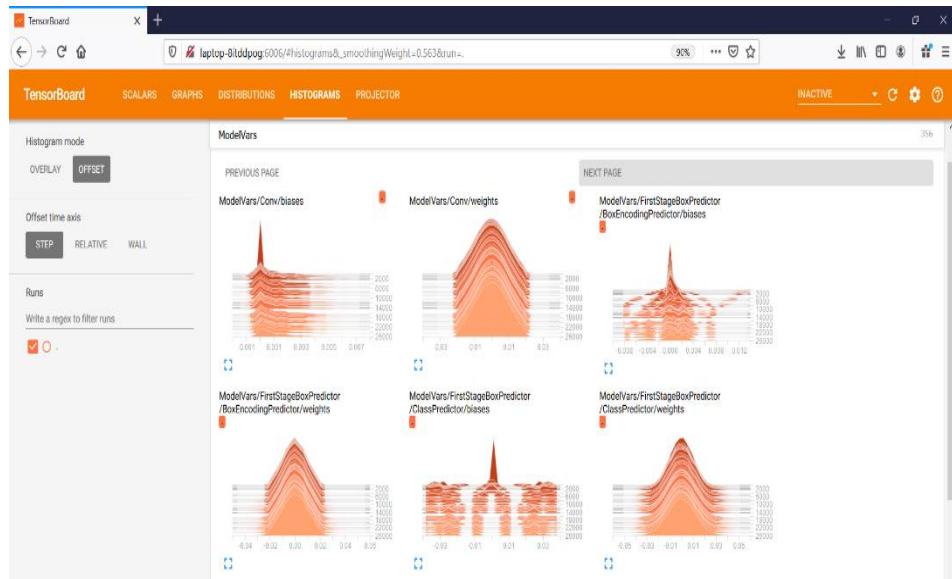
Dapat dilihat pada grafik loss gambar 6.7 bahwa loss sudah dibawah 0.03, angka ini sangat bagus yang berarti komputer melakukan pembelajaran dengan benar sehingga mencapai loss yang rendah.

## 2. Grafik Distributions



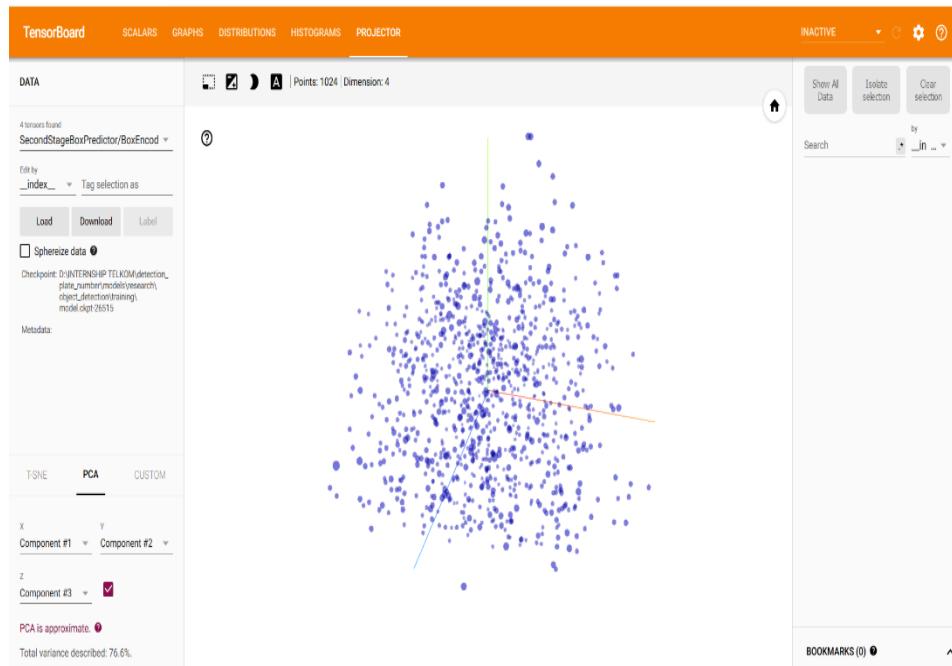
Gambar 6. 30 Grafik Distributions

### 3. Grafik Histograms



Gambar 6. 31 Grafik Histogram

### 4. Projector



Gambar 6. 32 Grafik Histogram

### 6.3.9 Mendapatkan Model

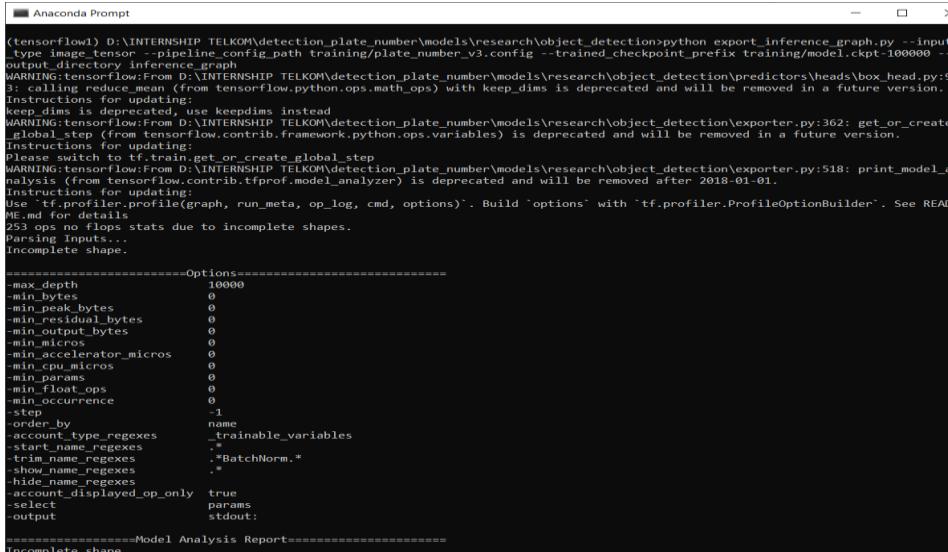
Proses training belum sepenuhnya selesai karena tujuan utama dari proses training Neural Network adalah mendapatkan sebuah model yang akan digunakan untuk mendeteksi objek-objek yang telah kita masukkan sebelumnya, maka dari itu perlu dilakukan proses generate model yang nantinya akan tersimpan ke dalam bentuk file frozen\_inference\_graph.pb. Jalankan perintah berikut, di mana chekpoint diisi dengan file .ckpt bernomor tertinggi di folder training yang tersimpan yaitu 100.000:

```
(tensorflow1) D:\SEMESTER 7\object_detection>python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/plate_number_v3.config --trained_checkpoint_prefix training/model.ckpt-100000 --output_directory inference_graph
```

Gambar 6. 33 Generate Model

Perintah ini akan menciptakan file frozen\_inference\_graph.pb pada folder object\_detection \ inference\_graph. File.pb berisi classifier deteksi objek yang digunakan untuk proses pengujian data pada model.

Berikut adalah proses generate untuk mendapatkan model Faster R-CNN yang telah di training:



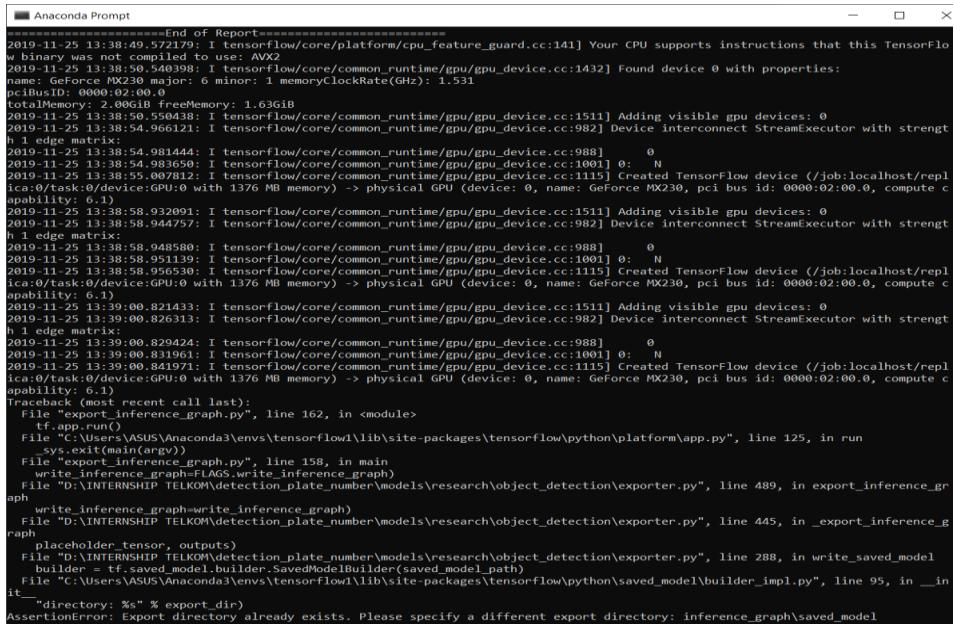
```
[Anaconda Prompt]
(tensorflow1) D:\INTERNSHIP_TELKOM\detection_plate_number\models\research\object_detection>python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/plate_number_v3.config --trained_checkpoint_prefix training/model.ckpt-100000 --output_directory inference_graph
WARNING:tensorflow:From D:\INTERNSHIP_TELKOM\detection_plate_number\models\research\object_detection\predictors\heads\box_head.py:93: calling reduce_mean (from tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
Keep dims is deprecated, use keepdims instead
WARNING:tensorflow:From D:\INTERNSHIP_TELKOM\detection_plate_number\models\research\object_detection\exporter.py:362: get_or_create_global_step (from tensorflow.contrib.framework.python.ops.variables) is deprecated and will be removed in a future version.
Instructions for updating:
Please switch to tf.train.get_or_create_global_step
WARNING:tensorflow:From D:\INTERNSHIP_TELKOM\detection_plate_number\models\research\object_detection\exporter.py:518: print_model_analysis (from tensorflow.contrib.tfprof.model_analyzer) is deprecated and will be removed after 2018-01-01.
Instructions for updating:
Use 'tf.profiler.profile(graph, run_meta, op_log, cmd, options)' . Build 'options' with 'tf.profiler.ProfileOptionBuilder'. See README.md for details
253 ops no flops stats due to incomplete shapes.
Parsing Inputs...
Incomplete shape.

=====
Options=====
--max_depth          10000
--min_bytes           0
--min_per_node_bytes   0
--min_residual_bytes   0
--min_output_bytes     0
--min_micros           0
--min_accelerator_micros 0
--min_cpu_micros       0
--min_params            0
--min_float_ops         0
--min_occurrence        0
--step                  -1
--order_by              name
--account_type_regexes _trainable_variables
--start_name_regexes    "."
--tune_name_regexes     ".BatchNorm.*"
--show_name_regexes      "."
--hide_name_regexes     "."
--account_displayed_op_only true
--select                 params
--output                stdout

=====
Model Analysis Report=====
Incomplete shape.
```

Gambar 6. 34 Proses Generate Model

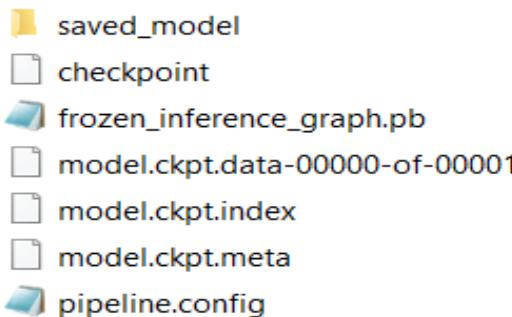
Proses ini membutuhkan waktu beberapa detik sampai generate berhasil dan akan menampilkan seperti gambar berikut:



```
■ Anaconda Prompt
=====
2019-11-25 13:38:49.572179: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2019-11-25 13:38:50.540398: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: GeForce MX230
pciBusID: 0000:02:00.0
totalMemory: 2.006GB freeMemory: 1.63GB
2019-11-25 13:38:50.550438: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-11-25 13:38:54.966121: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength h 1 edge matrix:
2019-11-25 13:38:54.981444: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-11-25 13:38:54.983650: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-11-25 13:38:55.007812: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/relica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pci bus id: 0000:02:00.0, compute capability: 6.1)
2019-11-25 13:38:58.932091: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-11-25 13:38:58.944757: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength h 1 edge matrix:
2019-11-25 13:38:58.948589: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-11-25 13:38:58.951130: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-11-25 13:38:58.956530: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/relica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pci bus id: 0000:02:00.0, compute capability: 6.1)
2019-11-25 13:39:00.821433: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-11-25 13:39:00.826313: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength h 1 edge matrix:
2019-11-25 13:39:00.829424: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-11-25 13:39:00.831961: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-11-25 13:39:00.841971: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/relica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pci bus id: 0000:02:00.0, compute capability: 6.1)
Traceback (most recent call last):
  File "export_inference_graph.py", line 162, in <module>
    t = tf.Session()
  File "C:\Users\ASUS\Anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\platform\app.py", line 125, in run
    sys.exit(main(argv))
  File "export_inference_graph.py", line 158, in main
    write_inference_graph(FLAGS.write_inference_graph)
  File "D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\exporter.py", line 489, in export_inference_graph
    write_inference_graph=write_inference_graph)
  File "D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\exporter.py", line 445, in _export_inference_graph
    placeholder_tensor, outputs)
  File "D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection\exporter.py", line 288, in write_saved_model
    builder = tf.saved_model.builder.SavedModelBuilder(saved_model_path)
  File "C:\Users\ASUS\Anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\saved_model\builder_impl.py", line 95, in __init__
    "directory: %s" % export_dir)
AssertionError: Export directory already exists. Please specify a different export directory: inference_graph\saved_model
```

Gambar 6. 35 Proses Generate Model NN

Generate model akan menciptakan file frozen\_inference\_graph.pb pada folder object\_detection\ inference\_graph. File .pb berisi classifier deteksi objek yang digunakan untuk proses pengujian data pada model. Berikut adalah file frozen\_inference\_graph.pb hasil generate model yang sudah selesai di training:



Gambar 6. 36 File Frozen Inference Graph

Kita sudah mendapatkan model Neural Network yang kita latih, selanjutnya adalah melakukan pengujian untuk mengetahui confidence rate yang dicapai dari model yang kita buat.

#### **6.4 Proses Pengujian Model**

Pengujian model dapat dilakukan dengan inputan berbentuk file image, file video dan secara live dengan webcam. Untuk melakukan pengujian maka dibuat kode program dengan nama file Object\_detection\_image.py, Object\_detection\_vidio.py dan Object\_detection\_webcam.py. Kode program berisi perintah untuk memanggil model hasil training yang telah dibuat dan file inputan yang akan diuji. Berikut adalah kode program yang akan digunakan pada saat pengujian model:

##### **6.4.1 Pengujian Image**

Tabel 6. 7 Kode Program Pengujian Image

Object_detection_image.py
#import library import numpy as np import cv2 from detector import detector from imutils import paths import os import time  #import plot_cv dari folder utils from utils import plot_cv #variabel det untuk memanggil class detector det = detector.detector()

```

#mengatur path lokasi image yang akan di testing
image_folder = 'test_image'
#menyimpan file test image
imageNames = os.listdir(image_folder)
#nama class
name_class = ["plate"]
# mengulang pada setiap image yang ada pada folder
for imageName in imageNames:
    #membaca file gambar
    image = cv2.imread(image_folder + '/' + imageName)
    #untuk ngedetek lokasi kotak pada gambar
    (boxes, scores, classes, num, dump ) = det.detect_plate(image)
    #jika dalam gambar tidak ditemukan box maka next
    if len(boxes) == 0:
        print('skip detection')
        continue

    #get ukuran gambar original
    h, w = image.shape[:2]
    boxes = boxes[0]
    scores = scores[0]
    classes = classes[0]
    #untuk filter box mana yang akan ditampilkan
    for box, score, class_idx in zip(boxes, scores, classes):
        (startY, startX, endY, endX) = box

        #koordinat box

```

```

startX = int(startX * w)
startY = int(startY * h)
endX = int(endX * w)
endY = int(endY * h)

#koordinat box yang di dapat
box = (startX, startY, endX, endY)

#untuk menampilkan persentase confidence rate dalam %
text = name_class[int(class_idx-1)] + " " + str(int(score * 100)) +
"%"

#if score lebih besar dari 0,7 maka box akan ditampilkan
if score > 0.7:
    image = plot_cv.plot_object(image, box, text, int(class_idx-1))

# Semua hasil telah diambil pada gambar. Sekarang tampilkan
gambar.

cv2.namedWindow("detector", cv2.WINDOW_KEEPRATIO)
cv2.setWindowProperty("detector",
cv2.WND_PROP_ASPECT_RATIO, cv2.WINDOW_KEEPRATIO)
cv2.imshow('detector', image)

# tekan apapun untuk menutup gambar
cv2.waitKey(0)

# Menutup window
cv2.destroyAllWindows()

```

Berikut merupakan penjelasan dari kode pada tabel 6.7 :

1. Baris 1 dan seterusnya melakukan import library.
2. Baris 9 import plot\_cv dari folder utils
3. Baris 10 variabel det untuk memanggil class detector.

4. Baris 16 mengatur path lokasi image yang akan di testing
5. Baris 19 menyimpan file test image
6. Baris 22 nama class
7. Baris 25 mengulang pada setiap image yang ada pada folder
8. Baris 27 membaca file gambar
9. Baris 31 untuk ngedetek lokasi kotak pada gambar
10. Baris 34 - 36 jika dalam gambar tidak ditemukan box maka next
11. Baris 39 - 42 get ukuran gambar original
12. Baris 45 - 46 untuk filter box mana yang akan ditampilkan
13. Baris 48 - 51 koordinat box
14. Baris 53 koordinat box yang di dapat
15. Baris 55 untuk menampilkan persentase confidence rate dalam %
16. Baris 57 - 58 jika score lebih besar dari 0,7 maka box akan ditampilkan
17. Baris 61- 63 Semua hasil telah diambil pada gambar. Sekarang tampilkan gambar.
18. Baris 66 tekan apapun untuk menutup gambar
19. Baris 69 Menutup window.

Untuk melakukan pengujian pada inputan image maka masukkan nama file image yang akan dipanggil ke dalam file object\_detection\_image.py. yaitu pada code program berikut IMAGE\_NAME = 'testing/testing\_1.JPG'. Selanjutnya jalankan perintah berikut pada anaconda command prompt untuk memulai proses training gambar:

```
Anaconda Prompt - python Object_detection_image.py

(tensorflow1) D:\[INTERNSHIP TELKOM]\detection_plate_number\models\research\object_detection\python Object_detection_image.py
2019-11-29 09:36:14.756498: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
2019-11-29 09:36:15.671842: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: GeForce MX230 major: 6 minor: 1 memoryClockRate(GHz): 1.531
pciBusID: 0000:02:00.0
totalMemory: 2.006GiB freeMemory: 1.636GiB
2019-11-29 09:36:15.677018: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-11-29 09:36:18.849438: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-11-29 09:36:18.855596: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-11-29 09:36:18.856547: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-11-29 09:36:18.879451: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pci bus id: 0000:02:00.0, compute capability: 6.1)
2019-11-29 09:36:29.562137: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.05GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:29.881526: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.55GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:30.599282: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.06GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:30.655468: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.58GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:30.930796: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.76GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:30.947217: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.34GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:31.140418: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.59GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-11-29 09:36:31.174386: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.92GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
plate: 100%
```

Gambar 6. 37 Proses Pengujian Image

Tensorflow1 adalah environment yang digunakan untuk me-manage keseluruhan library yang digunakan. Kemudian path diarahkan pada direktori penyimpanan file object\_detection\_image.py. Dapat dilihat dalam proses pengujian ini menggunakan tensorflow-gpu yaitu Geforce MX230. Hasil persentase plate adalah 100% hal ini berarti bahwa confidence rate dari objek yang dideteksi sudah tinggi dan akurat.

Selanjutnya setelah code program berjalan maka, akan ditampilkan window object detection. Tahap ini merupakan tahap terakhir dalam pembuatan sistem objek deteksi, dimana kode program cv2.imshow('Plate Detector', image) akan menampilkan window objek deteksi plat. Window tersebut akan menampilkan gambar, video dan live webcam hasil inputan. Terdapat kotak (box) yang sudah mendetek objek platnya, kemudian confidence rate dari objek tersebut ditampilkan dalam bentuk persentase

(%) untuk mengetahui berapa % keakuratan pada objek yang terdeteksi, sehingga dapat diketahui model yang di buat dan di training dengan Faster Region-Convolution Neural Network (R-CNN) apakah berhasil melakukan testing dengan benar. Berikut adalah gambar hasil pengujian data image :



Gambar 6. 38 Menampilkan Objek Deteksi Image

Dari gambar 6.38 dapat dilihat penulis menguji sistem objek deteksi dengan inputan gambar kendaraan mobil, dan hasil pengujian menyatakan bahwah objek class plate yang terdeteksi hanya satu yang ditandai dengan box berwarna hijau. persentase dari objek yang terdeteksi adalah 100% plate. Hal ini berarti model objek deteksi yang telah dibuat sudah bisa mengenali objek dengan benar dan akurat.

#### **6.4.2 Pengujian Vidio**

Tabel 6. 8 Kode Program Pengujian Vidio

Object_detection_video.py
#import library import numpy as np import cv2 from detector import detector from imutils import paths import os import time #import plot_cv dari folder utils from utils import plot_cv #untuk notifikasi print("[INFO] starting video stream...") #untuk mengatur path pada file yang akan di test vs = cv2.VideoCapture('test_image/testing_vidio.mp4') #melakukan jeda time.sleep(2.0) #variabel det untuk memanggil class detector det = detector.detector() #nama class name_class = ["plate"] # Mengulang setiap gambar while True: #print(imageName) ret, image = vs.read() #untuk mendetek

```

(boxes, scores, classes, num, category_index) =
det.detect_plate(image)

#jika box tidak ada maka next
if len(boxes) == 0:
    print('skip detection')
    continue

#ukuran gambar original
h, w = image.shape[:2]
boxes = boxes[0]
scores = scores[0]
classes = classes[0]

#untuk filter box mana yang akan ditampilkan
for box, score, class_idx in zip(boxes, scores, classes):
    (startY, startX, endY, endX) = box
    #koordinat box dikali dengan lebar dan tinggi gambar
    startX = int(startX * w)
    startY = int(startY * h)
    endX = int(endX * w)
    endY = int(endY * h)
    #koordinat box yang di dapat
    box = (startX, startY, endX, endY)
    #untuk menampilkan persentase confidence rate dalam %
    text = name_class[int(class_idx-1)] + " " + str(int(score * 100)) +
    "%"
    #jika score lebih besar dari 0,7 maka box akan ditampilkan
    if score > 0.7:
        image = plot_cv.plot_object(image, box, text, int(class_idx-1))

```

```

# Semua hasil telah diambil pada gambar. Sekarang tampilkan
gambar.

cv2.namedWindow("Plant detector", cv2.WINDOW_KEEPRATIO)
cv2.setWindowProperty("Plant detector",
cv2.WND_PROP_ASPECT_RATIO, cv2.WINDOW_KEEPRATIO)
cv2.imshow('Plant detector', image)

# tekan apapun untuk menutup gambar
key = cv2.waitKey(1) & 0xFF

# jika menekan q maka proses loop berhenti
if key == ord("q"):

    break

# Menutup window
cv2.destroyAllWindows()

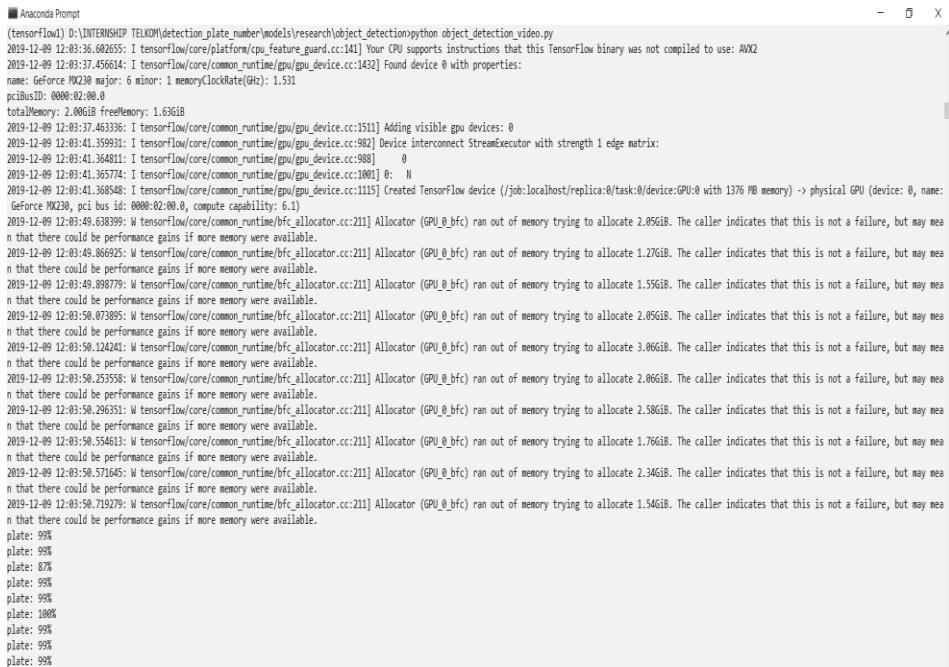
```

Berikut merupakan penjelasan dari kode pada tabel 6.8 :

1. Baris 1 - 7 dan seterusnya melakukan import library.
2. Baris 10 import plot\_cv dari folder utils
3. Baris 13 untuk notifikasi.
4. Baris 16 untuk mengatur path pada file yang akan di test
5. Baris 18 melakukan jeda
6. Baris 21 variabel det untuk memanggil class detector
7. Baris 24 nama class
8. Baris 27 mengulang setiap gambar
9. Baris 30 print(imageName)
10. Baris 32 untuk mendetek
11. Baris 34 - 36 jika box tidak ada maka next
12. Baris 38 - 41 ukuran gambar original
13. Baris 44 - 44 untuk filter box mana yang akan ditampilkan

14. Baris 47 - 50 koordinat box dikali dengan lebar dan tinggi gambar
15. Baris 52 koordinat box yang di dapat
16. Baris 54 untuk menampilkan persentase confidence rate dalam %
17. Baris 56 - 57 jika score lebih besar dari 0,7 maka box akan ditampilkan
18. Baris 60 – 62 Semua hasil telah diambil pada gambar. Sekarang tampilkan gambar.
19. Baris 65 tekan apapun untuk menutup gambar
20. Barid 68 jika menekan q maka proses loop berhenti
21. Baris 71 Menutup window.

Untuk melakukan pengujian pada inputan vidio maka masukkan nama file vidio yang akan dipanggil ke dalam file object\_detection\_video.py. yaitu pada code program berikut IMAGE\_NAME = 'testing/testing\_vidio'. Selanjutnya jalankan perintah berikut pada anaconda command prompt untuk memulai proses training pada vidio:



```

■ Anaconda Prompt
(tensorflow1) D:\INTERNSHIP TELKOM\detection_plate_number\models\research\object_detection>python object_detection video.py
2019-12-09 12:03:36.602655: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this Tensorflow binary was not compiled to use: AVX2
2019-12-09 12:03:37.456614: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properties:
name: GeForce MX230 major: 6 minor: 1 memoryClockRate(GHz): 1.531
pciBusID: 0000:02:00.0
totalMemory: 2.048GiB
2019-12-09 12:03:37.463336: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-12-09 12:03:41.359931: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-12-09 12:03:41.364813: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-12-09 12:03:41.365774: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-12-09 12:03:41.368540: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pci bus id: 0000:02:00.0, compute capability: 6.1)
2019-12-09 12:03:49.636399: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.056iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:49.866925: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.276iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:49.898779: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.556iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.073095: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.056iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.124241: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.066iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.253558: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.066iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.286351: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.586iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.554613: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.766iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.571645: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.346iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 12:03:50.719279: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.546iB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
plate: 99%
plate: 99%
plate: 87%
plate: 99%
plate: 99%
plate: 99%
plate: 100%
plate: 99%
plate: 99%
plate: 99%
plate: 99%
plate: 99%
```

Gambar 6. 39 Proses Pengujian Vidio

Tensorflow1 adalah environment yang digunakan untuk me-manage keseluruhan library yang digunakan. Kemudian path diarahkan pada direktori penyimpanan file object\_detection\_vedio.py. Dapat dilihat dalam proses pengujian ini menggunakan tensorflow-gpu yaitu Geforce MX230. Hasil persentase plate yang tercatat adalah 87%-100% hal ini berarti bahwa confidence rate dari objek yang dideteksi sudah tinggi dan akurat dan model yang digunakan untuk objek deteksi sudah bagus.

Selanjutnya setelah code program berjalan maka, akan ditampilkan window object detection. Tahap ini merupakan tahap terakhir dalam pembuatan sistem objek deteksi, dimana kode program cv2.imshow('Plate Detector', image) akan menampilkan window objek deteksi plat. Window tersebut akan menampilkan gambar, video dan live webcam hasil inputan. Terdapat kotak (box) yang sudah mendetek objek platnya, kemudian confidence rate dari objek tersebut ditampilkan dalam bentuk persentase (%) untuk mengetahui berapa % keakuratan pada objek yang terdeteksi, sehingga dapat diketahui model yang di buat dan di training dengan Faster Region-Convolution Neural Network (R-CNN) apakah berhasil melakukan testing dengan benar. Berikut adalah gambar 6.40 merupakan hasil pengujian data vidio :



Gambar 6. 40 Menampilkan Objek Deteksi Vidio

Dari gambar 6.13 dapat dilihat penulis menguji sistem objek deteksi dengan inputan vidio kendaraan motor. Hasil pengujian menyatakan bahwa objek class plate yang terdeteksi ada dua yang ditandai dengan box

berwarna hijau. persentase dari objek yang terdeteksi adalah 99% dan 100% plate. Hal ini berarti model objek deteksi yang telah dibuat sudah bisa mengenali objek dengan benar dan akurat.

#### 6.4.3 Pengujian Webcam

Tabel 6. 9 Kode Program Pengujian Webcam

Object_detection_webcam.py
----------------------------

```
#import library
import numpy as np
import cv2
from detector import detector
from imutils import paths
import os
import time
#import plot_cv dari folder utils
from utils import plot_cv
#untuk notifikasi
print("[INFO] starting video stream...")
#untuk mengaktifkan camera
vs = cv2.VideoCapture(0, cv2.CAP_DSHOW)
#mengatur ukuran camera
vs.set(cv2.CAP_PROP_FRAME_WIDTH, 1080)
vs.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
#melakukan jeda
time.sleep(2.0)
#variabel det untuk memanggil class detector
det = detector.detector()
#nama class
```

```

name_class = ["plate"]

# Mengulang setiap gambar

while True:

    #print(imageName)

    ret, image = vs.read()

    #untuk mendetek

    (boxes, scores, classes, num, category_index) = det.detect_plate(image)

    #jika box tidak ada maka next

    if len(boxes) == 0:

        print('skip detection')

        continue

    #ukuran gambar original

    h, w = image.shape[:2]

    boxes = boxes[0]

    scores = scores[0]

    classes = classes[0]

    #untuk filter box mana yang akan ditampilkan

    for box, score, class_idx in zip(boxes, scores, classes):

        (startY, startX, endY, endX) = box

        #koordinat box dikali dengan lebar dan tinggi gambar

        startX = int(startX * w)

        startY = int(startY * h)

        endX = int(endX * w)

        endY = int(endY * h)

        #box yang di dapat

        box = (startX, startY, endX, endY)

```

```

#untuk menampilkan persentase confidence rate dalam %
text = name_class[int(class_idx-1)] + " " + str(int(score *
100)) + "%"

#jika score lebih besar dari 0,7 maka box akan ditampilkan
if score > 0.7:
    image = plot_cv.plot_object(image, box, text,
int(class_idx-1))

# Semua hasil telah diambil pada gambar. Sekarang
tampilkan gambar.

cv2.namedWindow("detector",
cv2.WINDOW_KEEPAspectRatio)
cv2.setWindowProperty("detector",
cv2.WND_PROP_ASPECT_RATIO,
cv2.WINDOW_KEEPAspectRatio)

cv2.imshow('detector', image)

# tekan apapun untuk menutup gambar
key = cv2.waitKey(1) & 0xFF
# jika menekan q maka proses loop berhenti
if key == ord("q"):
    break

# Menutup window
cv2.destroyAllWindows()

```

Berikut merupakan penjelasan dari kode pada tabel 5.9 :

1. Baris 1 - 7 dan seterusnya melakukan import library.
2. Baris 10 import plot\_cv dari folder utils
3. Baris 13 untuk notifikasi.
4. Baris 16 untuk mengaktifkan camera

5. Baris 19 - 20 mengatur ukuran camera
6. Baris 23 melakukan jeda
7. Baris 26 variabel det untuk memanggil class detector
8. Baris 28 nama class
9. Baris 32 Mengulang setiap gambar
10. Baris 35 print(imageName)
11. Baris 37 untuk mendetek
12. Baris 39 - 41 jika box tidak ada maka next
13. Baris 43 - 46 ukuran gambar original
14. Baris 49 - 50 untuk filter box mana yang akan ditampilkan
15. Baris 52 - 55 koordinat box dikali dengan lebar dan tinggi gambar
16. Baris 57 box yang di dapat
17. Baris 59 untuk menampilkan persentase confidence rate dalam %
18. Baris 61 - 62 jika score lebih besar dari 0,7 maka box akan ditampilkan
19. Baris 65 – 67 Semua hasil telah diambil pada gambar. Sekarang tampilkan gambar.
20. Baris 70 tekan apapun untuk menutup gambar
21. Barid 73 jika menekan q maka proses loop berhenti
22. Baris 76 Menutup window.

Untuk melakukan pengujian inputan secara live di webcam maka jalankan file object\_detection\_webcam.py pada anaconda cmd. Kode program didalamnya akan mengakses camera yang akan digunakan jika ingin menggunakan camera internal maka atur nilai menjadi 0 namun jika ingin menggunakan camera external maka atur nilai menjadi 1 pada code berikut video = cv2.VideoCapture(0). Pengujian pada penelitian ini dilakukan dengan menggunakan camera internal atau kamera computer.

Berikut adalah hasil pengujian secara live menggunakan webcam :

```

[■] Anaconda Prompt - python object_detection_image.py
(tensorflow) D:\INTERNSHIP_TELKOM\detection_plate_number\models\research\object_detection\python object_detection_webcam.py
2019-12-09 10:39:11.850719 I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX
2019-12-09 10:39:12.630499 I tensorflow/core/common_runtime/gpu/gpu_device.cc:142] Found device 0 with properties:
name: GeForce MX230
pciBusID: 0000:01:00.0
memoryClockRate(GHz): 1.531
pciMemoryClockRate(GHz): 1.531
totalMemory: 2.08GB
2019-12-09 10:39:12.635481: I tensorflow/core/common_runtime/gpu/gpu_device.cc:151] Adding visible gpu devices: 0
2019-12-09 10:39:14.957708 I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-12-09 10:39:14.961022 I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-12-09 10:39:14.962847 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-12-09 10:39:14.963817 I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1376 MB memory) -> physical GPU (device: 0, name: GeForce MX230, pciBusID: 0000:01:00.0, memoryClockRate(GHz): 1.531)
2019-12-09 10:39:22.709046 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.0561B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:22.879111 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.5561B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:22.675742 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.0661B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:22.710176 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.5861B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:22.503034 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.7661B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:22.941264 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.3461B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:23.139098 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 3.5061B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-12-09 10:39:23.151649 W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.9261B. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
plate: 90%
plate: 99%
plate: 100%

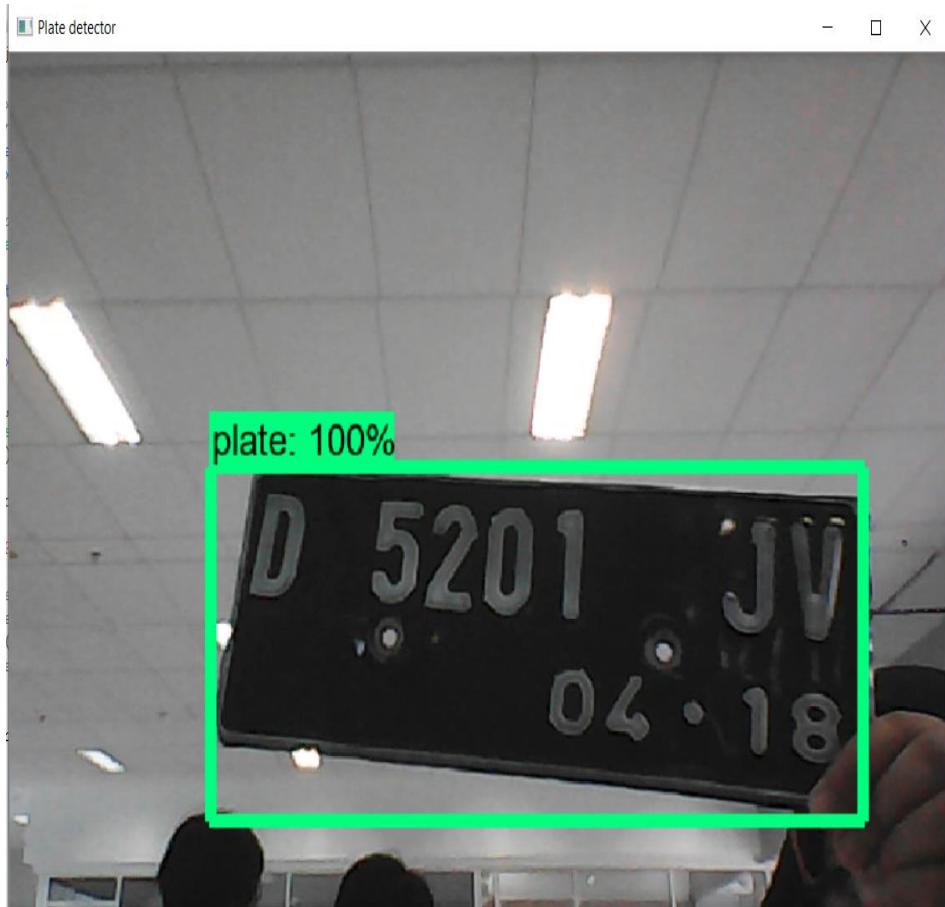
```

Gambar 6. 41 Proses Pengujian Webcam

Tensorflow1 adalah environment yang digunakan untuk me-manage keseluruhan library yang digunakan. Kemudian path diarahkan pada direktori penyimpanan file object\_detection\_webcam.py. Dapat dilihat dalam proses pengujian ini menggunakan tensorflow-gpu yaitu Geforce MX230. Hasil persentase plate yang tercatat adalah 90%-100% hal ini berarti bahwa confidence rate dari objek yang dideteksi sudah tinggi dan akurat dan model yang digunakan untuk object deteksi sudah bagus.

Selanjutnya setelah code program berjalan maka, akan ditampilkan window object detection. Tahap ini merupakan tahap terakhir dalam pembuatan sistem objek deteksi, dimana kode program cv2.imshow('Plate Detector', image) akan menampilkan window objek deteksi plat. Window tersebut akan menampilkan gambar, video dan live webcam hasil inputan. Terdapat kotak (box) yang sudah mendetek objek platnya, kemudian confidence rate dari objek tersebut ditampilkan dalam bentuk persentase (%) untuk mengetahui berapa % keakuratan pada objek yang terdeteksi, sehingga dapat diketahui model yang di buat dan di training dengan Faster

Region-Convolution Neural Network (R-CNN) apakah berhasil melakukan testing dengan benar. Berikut adalah gambar 6.40 merupakan hasil pengujian data webcam :



Gambar 6. 42 Menampilkan Objek Deteksi Webcam

Dari gambar 6.14 dapat dilihat penulis menguji sistem objek deteksi secara live pada webcam dengan menggunakan plat nomor motor. Hasil pengujian menyatakan bahwa objek class plate yang terdeteksi hanya satu yang ditandai dengan box berwarna hijau. persentase dari objek yang terdeteksi adalah 100% plate. Hal ini berarti model objek deteksi yang telah dibuat sudah bisa mengenali objek dengan benar dan akurat.

## BAB VII

---

### HASIL PENGUJIAN

#### 7.1.1 Kesimpulan Pengujian

Berdasarkan hasil sistem yang telah dibuat dan hasil pengujian yang telah dilakukan telah berhasil menjawab masalah pada penelitian ini, yaitu bagaimana membuat sistem objek deteksi dengan mengimplementasikan metode CNN untuk mendeteksi plat nomor kendaraan. Hasil penelitian menggunakan python untuk menjalankan perintah membuat model dengan tensorflow, tensorflow sebagai alat pembantu dalam pembuatan sistem agar data dapat diolah dan ditraining. Kemudian metode CNN dengan arsitektur Faster R-CNN digunakan untuk proses training yang model hasil training-nya digunakan untuk proses pengujian. Hasil pengujian dari penelitian ini dapat dilihat pada table 7.1 sebagai berikut:

Tabel 7. 1 Hasil Pengujian

No.	Device	Inputan	Format	Klasifikasi	Confidence rate (%)
1.	Laptop Asus	Gambar	JPG	Plate	100%
2.	Laptop Asus	Vidio	MP4	Plate	80-100%
3.	Laptop Asus	Webcam	MP4	Plate	80-100%