**PEMBUATAN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DITRAINING MENGGUNAKAN GPU**

**SKRIPSI**

**Oleh:**

**ADWI ARIFIN**

**115090300111008**

****

**JURUSAN FISIKA**

**FAKULTAS MATEMATIKA DAN   
ILMU PENGETAHUAN ALAM**

**UNIVERSITAS BRAWIJAYA**

**MALANG**

**2017**

**COVER DALAM**

**LEMBAR PENGESAHAN SKRIPSI**

**LEMBAR PERNYATAAN**

**ABSTRAK**

**ABSTRACT**

**KATA PENGANTAR**

DAFTAR ISI

[DAFTAR ISI iii](#_Toc513570147)

[DAFTAR GAMBAR v](#_Toc513570148)

[DAFTAR TABEL vi](#_Toc513570149)

[BAB I PENDAHULUAN 7](#_Toc513570150)

[1.1 Latar Belakang 7](#_Toc513570151)

[1.2 Rumusan Masalah Penelitian 8](#_Toc513570152)

[1.3 Batasan Masalah 8](#_Toc513570153)

[1.4 Tujuan Penelitian 8](#_Toc513570154)

[1.5 Manfaat Penelitian 8](#_Toc513570155)

[BAB II TINJAUAN PUSTAKA 9](#_Toc513570156)

[2.1 Artificial Intelegence 9](#_Toc513570157)

[2.2 Machine Learning 9](#_Toc513570158)

[2.3 Artificial Neural Network 10](#_Toc513570159)

[2.4 Deep Learning 10](#_Toc513570160)

[2.5 Python 11](#_Toc513570161)

[2.6 Keras 11](#_Toc513570162)

[2.7 TensorFlow 11](#_Toc513570163)

[2.8 CPU 12](#_Toc513570164)

[2.9 GPU 13](#_Toc513570165)

[2.10 Mean Squared Error 13](#_Toc513570166)

[BAB III METODE PENELITIAN 14](#_Toc513570167)

[3.1 Lokasi dan Waktu 14](#_Toc513570168)

[3.2 Alat dan Bahan 14](#_Toc513570169)

[3.3 Tahapan Penelitian 14](#_Toc513570170)

[3.3.1. Install Software 15](#_Toc513570171)

[3.3.2. Generate Data 16](#_Toc513570172)

[3.3.3. Training 19](#_Toc513570173)

[3.3.4. Testing 21](#_Toc513570174)

[BAB IV HASIL DAN PEMBAHASAN 23](#_Toc513570175)

[4.1 Hasil Generate Data 23](#_Toc513570176)

[4.2 Hasil Proses Training 24](#_Toc513570177)

[4.2.1. Training dengan GPU 24](#_Toc513570178)

[4.2.2. Training dengan CPU 25](#_Toc513570179)

[4.2.3. Perbandingan Training CPU dan GPU 26](#_Toc513570180)

[4.3 Hasil Testing 26](#_Toc513570181)

[4.3.1. Data Testing GPU 27](#_Toc513570182)

[4.3.2. Data Testing CPU 28](#_Toc513570183)

[4.3.3. Data Testing GPU 100 Epoch 29](#_Toc513570184)

[4.3.4. Interpretasi Data Testing 30](#_Toc513570185)

[4.3.5. Visual Testing 31](#_Toc513570186)

[BAB V PENUTUP 33](#_Toc513570187)

[5.1 KESIMPULAN 33](#_Toc513570188)

[5.2 SARAN 33](#_Toc513570189)

[DAFTAR PUSTAKA 35](#_Toc513570190)

[LAMPIRAN 37](#_Toc513570191)

DAFTAR GAMBAR

[Gambar 2.1 Artificial Neural Network 10](#_Toc513570402)

[Gambar 2.2 Chip CPU 12](#_Toc513570403)

[Gambar 2.3 Chip GPU 13](#_Toc513570404)

[Gambar 2.4 Persamaan MSE 13](#_Toc513570405)

[Gambar 3.1 Alur Penelitian 15](#_Toc513570406)

[Gambar 3.3 Tampilan Program Simulator Self Driving Car 18](#_Toc513570407)

[Gambar 3.4 Tampilan Pilih Folder 18](#_Toc513570408)

[Gambar 3.5 Model Convolutional Neural Network 20](#_Toc513570409)

[Gambar 3.6 Tampilan Autonomous Drive 22](#_Toc513570410)

[Gambar 4.1 File CSV hasil training. 23](#_Toc513570411)

[Gambar 4.2 Gambar yang diambil dari tampak 24](#_Toc513570412)

[Gambar 4.3 Proses Training dengan GPU 25](#_Toc513570413)

[Gambar 4.4 Proses Training dengan CPU 25](#_Toc513570414)

[Gambar 4.5 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan GPU 27](#_Toc513570415)

[Gambar 4.6 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan CPU 28](#_Toc513570416)

[Gambar 4.7 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan GPU dengan 100 Epoch 29](#_Toc513570417)

[Gambar 4.8 Visual Testing 31](#_Toc513570418)

[Gambar 4.9 Area rawan pinalti 32](#_Toc513570419)

DAFTAR TABEL

BAB I  
PENDAHULUAN

1. BAB I PENDAHULUAN
   1. Latar Belakang

Perkembangan self-driving car semakin lama semakin baik, banyak perusahaan berlomba-lomba membuat mobilnya sendiri untuk dapat dikendarai tanpa pengemudi. Hal salah satunya disebabkan oleh kemajuan di bidang teknik Machine Learning, yaitu Deep Learning. Pada dasarnya Deep Learning adalah Artificial Neural Network dengan jumlah Hidden Layer yang sangat banyak, maka dari itu disebut “*deep*”. Teknik ini pertama kali dikenalkan sudah sejak 1986 dengan nama Backpropagation (Rumelhart, Hinton and Williams, 1986), namun teknik ini tidak berkembang dengan pesat karena sulit diaplikasikan ke masalah yang lebih besar. Setelah itu pada tahun 2006 muncul ide bahwa seharusnya neural network seharusnya tidak dibangun atas 2 atau 3 layer saja, namun lebih dari itu, dari situ lah muncul istilah *Deep Learning* (Bengio and Lecun, 2007). Seiring dengan perkembangan jaman, pada tahun 2012 perkembangan GPU meningkat secara signifikan, yang salah satu efeknya adalah memungkinkan melatih neural network dengan menggunakan GPU (Krizhevsky, Sutskever and Geoffrey E., 2012). Sejak saat itu banyak penelitian tentang Deep Neural Network, beberapa aplikasi yang digunakan antara lain Image Recognition, Voice Recognition, Pattern Recognition, Natural Languange Processing, dan lain-lain. Perkembangan Deep Neural Network di bidang Image Recognition, dalam beberapa aspek dapat mengimbangi dan bahkan lebih baik dibanding manusia (He *et al.*, 2015).

Dari sini penulis ingin mencoba menunjukkan bagaimana pemanfaatan Deep Learning spesifik pada kasus Self Driving Car. Kendala dari topik tersebut membutuhkan kompleksitas dan biaya yang cukup tinggi, karena membutuhkan mobil dengan spesifikasi yang memadai, lalu dibutuhkan modifikasi pada mobil itu, hingga memasangkan perangkat lunak untuk mengontrol mobil tersebut. Pada bulan Februari 2017, salah satu perusahaan Online Course yang bernama Udacity, meng-open-source kan salah satu produknya yaitu Udacity Self Driving Simulator (Darrell Etherington, 2017). Simulator ini adalah program yang bisa dijalankan di komputer, lalu bertindak selayaknya mobil yang akan kita program agar dapat mengendarai dirinya sendiri. Simulator ini dapat menyediakan / menghasilkan data training, sekaligus dapat diberikan perintah untuk mengontrol mobil tersebut. Oleh karena itu penelitian ini tidak akan menggunakan mobil nyata, namun menggunakan simulator sebagai penggantinya.

* 1. Rumusan Masalah Penelitian

Berdasrkan latar belakang yang telah disebutkan, dapat dirumuskan beberapa rumusan masalah pada tugas akhir ini adalah:

1. Bagaimana membuat program untuk dapat mengontrol simulator dengan menggunakan Deep Neural Network?
2. Bagaimana proses training Deep Neural Network?
3. Bagaimana perbandingan proses training menggunakan CPU dan GPU?
4. Bagaimana proses testing pada simulator? Seberapa akurat?
   1. Batasan Masalah

Ruang lingkup penelitian tugas akhir ini dibatasi beberapa hal yaitu:

1. Simulator digunakan sebagai pengganti objek mobil yang sebenarnya.
2. Hanya mengamati bagaimana hasil training maupun testing, tidak membahas simulator.
   1. Tujuan Penelitian

Tujuan penelitian pada tugas akhir ini adalah:

1. Membuat program training self-driving car pada simulator.
2. Membandingkan training menggunakan CPU maupun GPU.
3. Menganalisa tingkat presisi hasil training self-driving car pada simulator
   1. Manfaat Penelitian

Manfaat penelitian tugas akhir ini adalah:

1. Sebagai tahap awal penelitian tentang self-driving car, agar kedepannya nanti dapat dibuat model-model yang lebih baik.
2. Terciptanya model self-driving car yang aman dan nyaman. Setidaknya pada tingkat simulator.

BAB II  
TINJAUAN PUSTAKA

1. BAB IV HASIL DAN PEMBAHASAN
   1. Artificial Intelegence

Artificial Intelegence (AI) atau dalam bahasa indonesia disebut dengan Kecerdasan Buatan adalah suatu teknik pengembangan mesin yang dapat berpikir ataupun bertindak layaknya manusia. Cakupan Kecerdasan Buatan cukup luas, meliputi perencanaan (planning), dapat memahami bahasa, dapat mengenali objek ataupun suara, penyelesaian masalah, dan bahkan pembelajaran.

Secara umum kecerdasan buatan dapat dibagi menjadi dua kategori, yaitu kecerdasan umum (General AI) yaitu kecerdasan buatan yang dapat melakukan semua hal yang telah disebutkan. Kategori berikutnya yaitu kecerdasan khusus (Spesific AI) yaitu kecerdasan yang dapat melakukan satu tugas dengan sangat baik, namun tidak dapat melakukan tugas yang lain. Contohnya ada kecerdasan yang dapat mengenali objek dengan baik, namun tidak dapat mengenali suara sama sekali. (McClelland, 2017)

* 1. Machine Learning

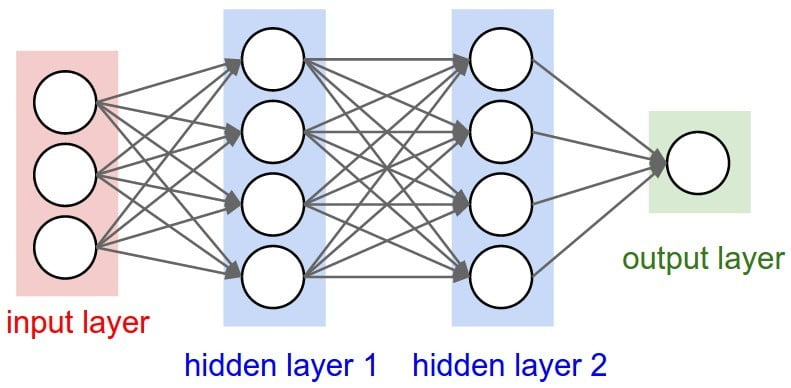
Machine Learning (ML) atau dalam bahasa indonesia disebut dengan pembelajaran mesin, adalah suatu pendekatan untuk dapat mewujudkan kecerdasan buatan. Pada tahun 1959, Arthur Samuel pernah mengemukakan pendapatnya tentang Machine Learning yaitu “suatu kemampuan untuk belajar tanpa harus diprogram secara eksplisit”. Dari pernyataan Arthur dapat kita ketahui bahwa pada dasarnya kita bisa mendapatkan kecerdasan buatan tanpa menggunakan pembelajaran mesin, yaitu dengan cara membuat perintah percabangan yang cukup banyak dan kompleks.

Dibanding dengan cara percabangan yang cukup banyak, pembelajaran mesin hadir dengan cara melatih program agar dapat mengerti input ataupun output yang diberikan. Dalam proses ini biasanya kita memberikan sejumlah input ataupun output yang sangat banyak, lalu membiarkan program untuk menyesuaikan sendiri apa korelasi antara input dan output yang diberikan.

Salah satu contoh dari dari pembelajaran mesin yaitu untuk mengenali suatu objek dalam gambar. Misalnya kita memberikan sejumlah gambar dan melabeli beberapa gambar tersebut dengan gambar yang memiliki objek kucing dan yang tidak memiliki kucing. Pada akhirnya algoritma pembelajaran mesin ini dapat membentuk suatu model yang dapat mengenali ada atau tidaknya kucing dalam suatu gambar. (McClelland, 2017)

* 1. Artificial Neural Network

Artificial Neural Network tersusun atas beberapa elemen, yaitu node, network, dan layer. Node disimbolkan dengan simbol lingkaran, yaitu merepresentasikan suatu informasi. Network disimbolkan dengan tanda panah, yaitu menghubungkan antar satu node dengan node yang lainnya, satu tanda panah berarti node awal akan mempengaruhi node yang ditunjuk. Dalam network ini terdapat properti yang disebut dengan “weight” atau “bobot” dan juga “bias”. Layer merepresentasikan sekumpulan node yang tergabung dalam satu kelompok. Dalam satu Artificial Neural Network dipastikan memiliki satu input layer, satu output layer dan berapapun hidden layer. Jumlah hidden layer yang sedikit sering disebut dengan Shallow Neural Network, sedangkan jumlah hidden layer yang cukup banyak disebut dengan Deep Neural Network (Agatonovic-Kustrin and Beresford, 2000)



Gambar 2.1 Artificial Neural Network

* 1. Deep Learning

Deep Learning adalah salah satu teknik agar dapat mengimplementasikan pembelajaran mesin. Selain Deep Learning, terdapat beberapa teknik lain seperti Decision Tree Learning, Inductive Logic Programming, Clustering, Reinforcement Learning, Bayesian Network, dan masih banyak lagi yang lainnya. (McClelland, 2017)

Deep Learning merupakan suatu teknik pembelajaran mesin dengan menggunakan Deep Neural Network. Teknik Deep Learning ini awalnya tidak cukup populer karena dengan jumlah hidden layer yang cukup besar, maka kompleksitas program semakin besar, sehingga waktu komputasi yang dibutuhkan pun cukup besar. Namun dengan seiring perkembangan teknologi perangkat keras yang memadai, teknik Deep Learning kembali diminati. Karena dapat menyelesaikan permasalahan yang cukup kompleks.

* 1. Python

Python merupakan suatu bahasa pemrograman tingkat tinggi yang bersifat open source. Python memiliki struktur data tingkat tinggi yang efisien dan pendekatan terhadap pemrograman berorientasi object (OOP) yang sederhana namun efektif. Python telah banyak digunakan di pengembangan perangkat lunak, bisnis, pendidikan, pemerintahan, sains dan teknik. Kelebihan python yang lain yaitu didukung library yang cukup banyak dan terjamin karena di maintance oleh komunitasnya masing-masing (Ceder, 2010).

* 1. Keras

Keras merupakan salah satu library python yang ditujukan untuk pengolahan neural network. Keras mendukung banyak algoritma termasuk di dalamnya yaitu deep neural network. Kelebihan keras yaitu dengan sintaks yang sederhana, keras dapat menjalankan library lain antara lain TensorFlow, Theano, ataupun Microsoft Cognition Toolkit (*Keras: The Python Deep Learning library*, 2015).

* 1. TensorFlow

Tensor Flow sendiri adalah machine learning engine yang dipakai Google di banyak aplikasi mereka, mulai dari pengenalan suara, SmartReply yang membantu pengguna dengan mengidentifikasi email penting sekaligus memberikan usulan balasannya, pengenalan gambar yang memungkinkan kita melakukan pencarian berdasarkan foto, mengenali dan menerjemahkan tulisan dari sebuah foto, dan lain-lain.

Menurut Google, engine ini dapat digunakan baik dalam riset maupun komersial, mulai dari mesin besar sampai telepon genggam. TensorFlow menggunakan metode deep learning dalam prosesnya, namun dapat juga menggunakan reinforcement learning and logistic regression.

Tensorflow mendukung metode learning dengan menggunakan CPU ataupun GPU. Dengan melakukan learning menggunakan GPU, maka waktu yang dibutuhkan untuk melakukan proses training akan lebih cepat. Untuk saat ini GPU yang didukung hanya GPU dari NVIDIA saja (TensorFlow, 2018).

* 1. CPU

CPU (singkatan dari Central Processing Unit) adalah sebuah perangkat keras yang fungsi utamanya adalah untuk melakukan pemrosesan data. Sebuah CPU biasanya diukur dari berapa kalkulasi yang dapat dilakukan dalam satu detik, ukuran ini biasa ditampilkan dalam besaran Hz. Sebuah CPU dengan 3.7 GHz menyatakan prosesor tersebut dapat melakukan kalkulasi 3,7 milyar kalkulasi per detik. CPU memproses secara sequensial, yang berarti memproses data satu-persatu secara bergantian.



Gambar 2.2 Chip CPU

* 1. GPU

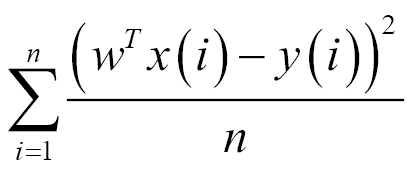
GPU (singkatan dari Graphics Processing Unit) adalah sebuah prossesor khusus untuk untuk bagian grafis 3D dari microprocessor. Alat ini digunakan di sistem benam, telepon genggam, komputer pribadi, workstation, dan konsol game. GPU Modern sangat efisien dalam memanipulasi komputer grafis dan struktur paralel, membuatnya lebih efektif dari fungsi umum CPU yang digunakan untuk bebagai perhitungan alogaritma. Pada komputer pribadi (PC), GPU biasanya terdapat di video card atau di motherboard. Lebih dari komputer desktop dan notebook mempunyai GPU yang terintegrasi, yang biasanya jauh daripada yang ada di video card.



Gambar 2.3 Chip GPU

* 1. Mean Squared Error

Mean Square Error (MSE) adalah rata-rata dari nilai error yang dikuadartkan. Persamaan ditunjukkan pada persamaan berikut



Gambar 2.4 Persamaan MSE

BAB III  
METODE PENELITIAN

1. BAB IV HASIL DAN PEMBAHASAN
   1. Lokasi dan Waktu

Penelitian ini dilakukan pada bulan Februari sampai bulan April 2018 bertempat di rumah penulis dan Student Activity Center Jurusan Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya Malang.

* 1. Alat dan Bahan

Alat yang digunakan dalam penelitian kali ini adalah seperangkat komputer dengan spesifikasi Processor Intel i3 6100 3.7 GHz, Motherboard Colorful Battle Axe C.B150M-D V23, Memory Corsair Value Select 1x8 GB, VGA Zotac GTX 1050 2 GB, Hard Drive 1 TB, Power Supply 450 Watt, serta Monitor AOC 19 inch dengan Resolusi 1366x768 pixel.

* 1. Tahapan Penelitian

Penelitian ini dilakukan dengan beberapa tahap sebagai berikut:

1. Install Software
2. Generate Data
3. Training
4. Testing
5. Interpretasi Hasil dan Pembahasan



Gambar 3.1 Alur Penelitian

* + 1. Install Software

Pada Tahap ini, akan dilakukan instalasi software-software pendukung yang akan digunakan untuk membuat script training ataupun testing. Software akan ditulis dalam bahasa python, sehingga kita perlu menginstall python interpreter, versi python yang akan digukanan adalah python versi 3.5.2.

Untuk memudahkan proses instalasi software pendukung / library python, digunakan aplikasi conda. Pustaka yang diperlukan antara lain, numpy, matplotlib, jupyter, opencv3, pillow, scikit-learn, scikit-image, scipy, h5py, eventlet, flask-socketio, seaboarn, pandas, imageio, moviepy, tensoflow, dan keras. Library yang telah disebutkan disimpan kedalam satu file yang disebut environment-gpu.yml, seperti gambar dibawah ini:



Gambar 3.2. Daftar software yang diinstall, disimpan dalam file environment-gpu.yml

Setelah file disimpan dalam file environtment-gpu.yml, maka dijalankan perintah berikut:

conda env create -f environment-gpu.yml

* + 1. Generate Data

Pada tahap ini diperlukan sejumlah data yang fungsinya nanti sebagai input pada di Deep Neural Network. Input yang dibutuhkan yaitu gambar tampak depan dari mobil serta aksi apa yang diberikan kepada mobil. Dalam penilitan ini gambar diambil dari 3 sudut, yaitu dari kiri, kanan serta tengah. Aksi yang direkam antara lain, steering-wheele, throttle, reverse, dan speed.

Metode pengambilan data ini dilakukan dengan cara membuka program simulator yang disediakan oleh Udacity Self Driving Simulator.



(a)



(b)

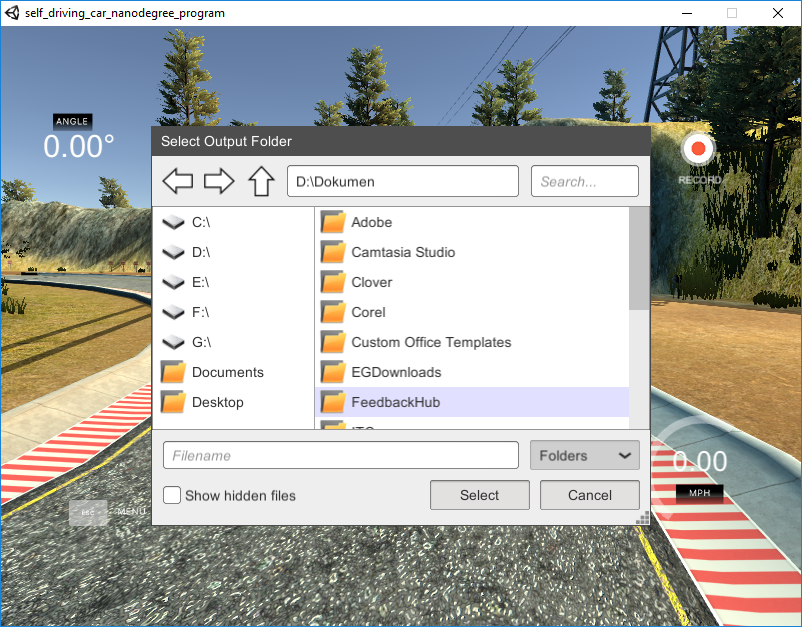


(c)

Gambar 3.3 Tampilan Program Simulator Self Driving Car

(a) Tampilan awal (b) Tampilan kontrol (c) Tampilan Training

Saat telah masuk dalam progam, klik Training Mode, setelah itu akan muncul tampilan Training, lalu tekan tombol [R] untuk melakukan perekaman. Setelah itu kita diminta untuk memilih satu folder yang digunakan untuk menyimpan data hasil perekaman.



Gambar 3.4 Tampilan Pilih Folder

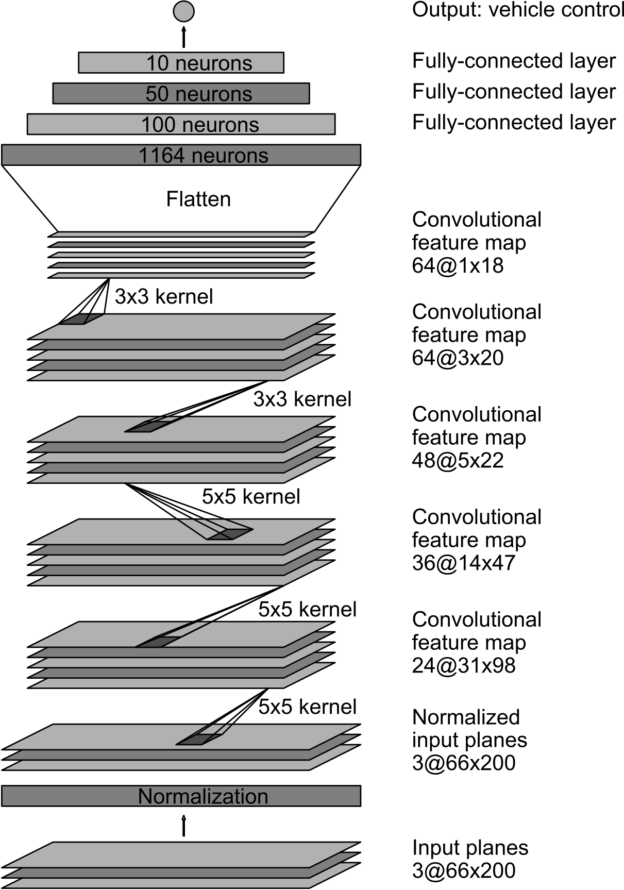
Setelah itu kita diminta untuk mengendarai mobil, diusahakan minimal mengendarai 1 putaran agar mobil dapat mengenali berbagai kondisi pengendaraan. Setelah dikendarai sejauh 1 putaran, ditekan lagi tombol [R] untuk berhenti melakukan perekaman.

* + 1. Training

Pada tahap ini dilakukan penulisan program training untuk dapat membangun model dari Deep Learning. Pada tahap training ini, program harus dapat membaca file gambar dan aksi yang telah kita simpan, lalu dapat menyimpan hasil training dalam suatu file. Dalam penelitian ini digunakan file hdf5 (Hierarchical Data Format versi 5) sebagai penyimpanan hasil training dari Deep Learning, hal ini dikarenakan hdf5 merupakan file binary yang terkompresi, sehingga tidak membutuhkan memori yang cukup banyak.

Dalam penelitian digunakan model Deep Learning yang terdiri dari Convolutional Neural Network. Alasan digunakan Convolutional Neural Network adalah dengan digunakannya Convolutional Neural Network, maka Network dapat mengekstraksi fitur-fitur abstrak, lalu dapat mengklasifikasi berdasarkan fitur-fitur abstrak yang telah dibuat. Konfigurasi jaringan yang digunakan dalam penelitian ini adalah Convolutional Neural Network dengan 10 hidden layer. Konfigurasi ini diinspirasi dari konfigurasi yang telah dipaparkan oleh NVIDIA (Bojarski *et al.*, 2016), dan susunannya ditunjukkan pada gambar 3.5. Pada tahap training data, digunakan algoritma optimisasi yang disebut *Adam Optimizer*. Algoritma ini dipilih karena algoritma ini merupakan algoritma terbaik menurut Kingma (Kingma and Ba, 2014).

Dalam proses training ini dilakukan dua kali macam training, yang pertama dilakukan adalah proses training menggunakan CPU, sedangkan proses yang kedua adalah proses training menggunakan GPU. Langkah ini dilakukan untuk membandingkan performa dan efisiensi antara CPU dan GPU.



Gambar 3.5 Model Convolutional Neural Network

Pada layer pertama terdapat layer input yang berupa citra digital dengan panjang 200 pixel, lebar 66 pixel dan 3 channel (red, green, blue). Setelah itu layer input dinormalisasi terlebih dahulu dengan persamaan x = x / 127.5 - 1.0. Setelah dilakukan normalisasi, masuk ke layer convolutional ke-1 dengan kernel sebesar 5x5, stride 2x2, dan memiliki 24 output convolutional, sehingga didapatkan output dari layer ini sebesar 24x98x31. Setelah itu masuk ke layer convolutional ke-2 dengan kernel sebesar 5x5, stride 2x2 dan memiliki output 36, sehingga didapatkan output dari layer ini sebesar 36x47x41. Setelah itu masuk ke layer convolutional ke-3 dengan kernel sebesar 5x5, stride 2x2 dan memiliki output 48 node, sehingga didapatkan output dari layer ini sebesar 48x22x5. Setelah itu masuk ke layer convolutional ke-4 dengan kernel sebesar 3x3, stride 1x1 dan memiliki output 64 node, sehingga didapatkan output dari layer ini sebesar 64x20x3. Setelah itu masuk ke layer convolutional ke-5 dengan kernel sebesar 2x2, stride 1x1 dan memiliki output 64 node, sehingga didapatkan output dari layer ini sebesar 64x18x1. Setelah itu diberikan fungsi flatten yang merubah bentuk data yang semula 64x18x1 menjadi satu baris dengan 1152 data. Setelah itu masuk ke layer selanjutnya yaitu fully connected layer dengan output sebesar 100 node. Setelah itu masuk ke layer selanjutnya yaitu fully connected layer ke-2, dengan output sebesar 50 node. Setelah itu masuk ke layer selanjutnya yaitu fully connected layer ke-3, dengan output sebesar 10 node. Setelah itu masuk ke layer output 1 node yang berupa hasil steering wheel.

* + 1. Testing

Pada tahap ini dilakukan testing data dengan dua cara, cara pertama yaitu dengan melihat accuracy dan loss value dari model yang dihasilkan. Setelah itu testing cara kedua dilakukan dengan cara membaca hasil model yang telah disimpan dalam format hdf5, lalu mengirimkannya melalui ke simulator dalam mode Autonomous Drive. Testing yang kedua ini untuk selanjutnya akan kita sebut dengan visual Testing. Dalam visual testing ini dikatakan berhasil jika mobil dapat berkendara secara mandiri dan dapat melewati 1 lap penuh tanpa menabrak halangan sama sekali.

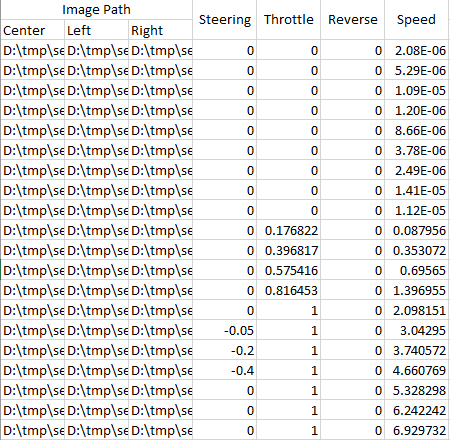


Gambar 3.6 Tampilan Autonomous Drive

BAB IV  
HASIL DAN PEMBAHASAN

1. BAB IV HASIL DAN PEMBAHASAN
   1. Hasil Generate Data

Data yang dihasilkan pada tahap ini berupa 1 file csv serta kumpulan gambar yang terdapat dalam folder images. Berikut cuplikan file csv dan cuplikan hasil gambar yang dihasilkan.



Gambar 4.1 File CSV hasil training.

Dari gambar diatas proses training menghasilkan 7 macam data. Tiga data pertama merupakan data alamat gambar disimpan, gambar yang dihasilkan juga terdapat 3 macam gambar, yaitu gambar dari sisi kiri mobil, gambar dari sisi tengah mobil dan gambar dari sisi kanan mobil, untuk lebih jelasnya lihat gambar 4.2.

(a) (b) (c)

Gambar 4.2 Gambar yang diambil dari tampak

(a) kiri (b) tengah (c) kanan

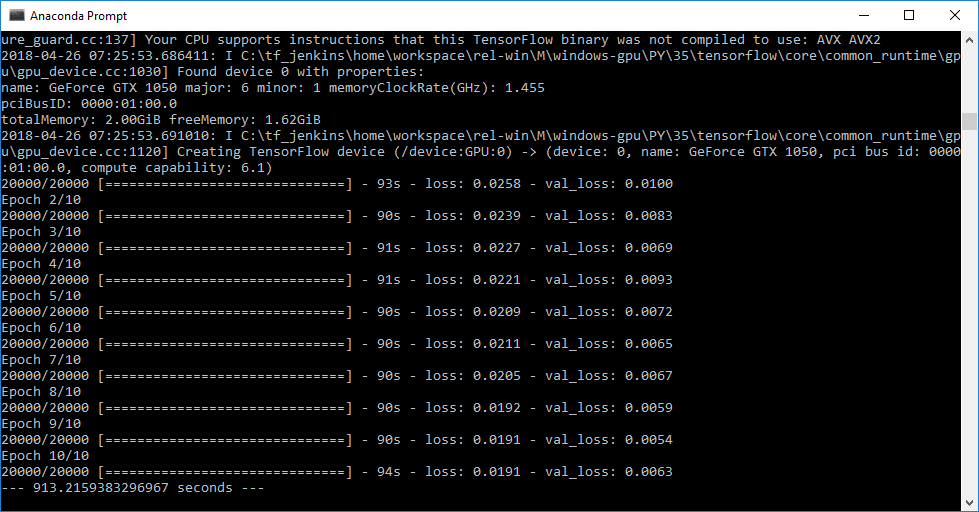
Tiga data berikutnya adalah data aksi yang kita berikan ke simulator yaitu steering yang mempunyai 3 macam nilai yaitu 0, bilangan positif atau bilangan negatif. Angka nol mengartikan bahwa steering tidak digerakkan ke arah kanan ataupun kiri, sedangkan bilangan positif menunjukkan steering digerakkan ke arah kanan dan bilangan negatif menunjukkan steering digerakkan ke arah kiri. Steering dalam simulator diwakili dengan pemberian input keyboard [A] untuk kiri dan [D] untuk kanan, atau input drag mouse. Aksi yang berikutnya yaitu throttle yang mewakili dari pedal gas yang ditekan, dalam simulator diwakili dengan penekanan tombol keyboard [W]. Aksi berikutnya yang direkam adalah reverse, mewakili apakah mobil dalam keadaan mundur atau tidak, nilai ini bernilai 0 untuk False dan 1 untuk True, dalam simulator aksi ini berkorelasi dengan penekanan tombol keyboard [S]. Kolom terakhir yang direkam yaitu kecepatan mobil dalam simulator, data ini bukanlah merupakan input melainkan output yang mana dari aksi-aksi yang kita berikan menghasilkan kecepatan seperti yang tertera pada kolom ini.

* 1. Hasil Proses Training

Proses Training dilakukan dengan dua cara yaitu dengan menggunakan CPU dan GPU. Langkah ini dilakukan untuk membandingkan performa dan efisiensi antara CPU dan GPU.

* + 1. Training dengan GPU

Training yang pertamakali dilakukan oleh penulis adalah training dengan menggunakan GPU. Dan hasilnya adalah sebagai berikut:

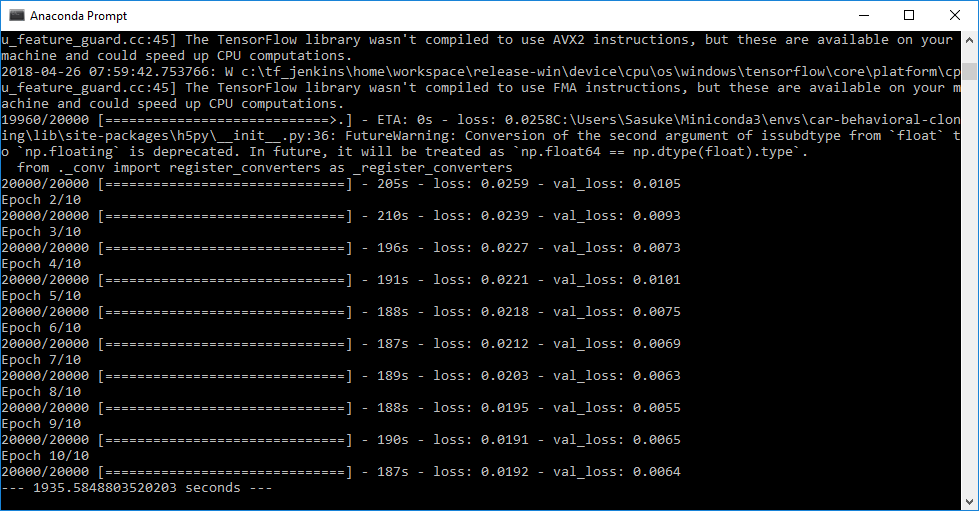


Gambar 4.3 Proses Training dengan GPU

Dari gambar diatas, dapat kita ketahui bahwa proses training dengan menggunakan GPU memakan waktu sebanyak 913 detik atau 15 menit 13 detik. Training menggunakan GPU ini juga dibatasi hingga 10 epoch atau 10 kali proses training, dari kesepuluh data tersebut, didapatkan data terbaik pada epoch ke-9 dimana pada epoch tersebut didapatkan nilai loss function 0.191 pada data training, dan 0.0054 pada data test.

* + 1. Training dengan CPU

Training berikutnya adalah training dengan menggunakan CPU, dan hasilnya dapat dilihat pada gambar berikut.



Gambar 4.4 Proses Training dengan CPU

Dari gambar diatas, dapat kita ketahui bahwa proses training dengan menggunakan CPU memakan waktu sebanyak 1935 detik atau 32 menit 15 detik. Training menggunakan CPU ini juga dibatasi hingga 10 epoch atau 10 kali proses training, dari kesepuluh data tersebut, didapatkan data terbaik pada epoch ke-8 dimana pada epoch tersebut didapatkan nilai loss function 0.195 pada data training, dan 0.0055 pada data test.

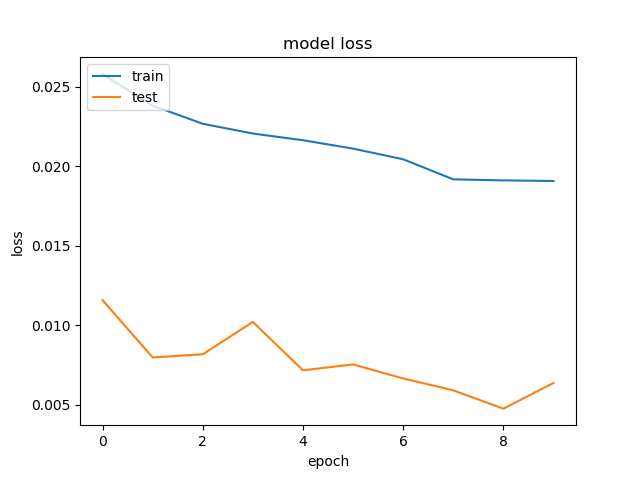
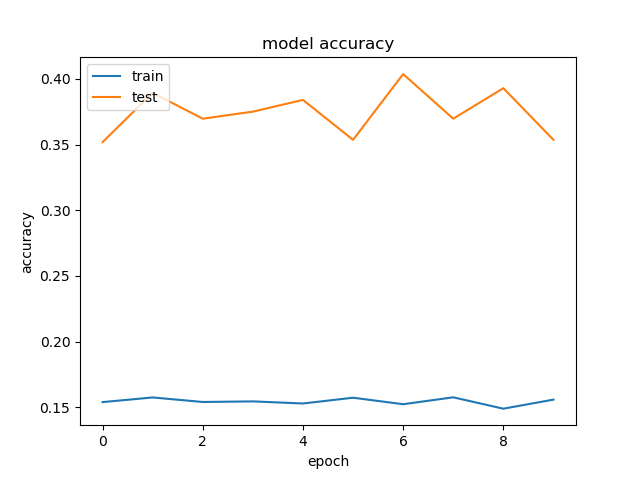
* + 1. Perbandingan Training CPU dan GPU

Secara umum training dengan CPU maupun GPU menghasilkan data yang tidak jauh berbeda, dapat dilihat pada loss function yang dihasilkan berkisar pada kisaran angka yang sama, yaitu 0.0054 pada training dengan menggunakan GPU sedangkan 0.0055 pada training dengan menggunakan CPU. Namun dari kedua proses training tersebut, waktu yang dibutuhkan berbeda. Pada proses training dengan CPU memakan waktu sebanyak 32 menit 15 detik, sedangkan pada GPU memakan waktu sebanyak 15 menit 13 detik. Dapat dilihat bahwa dengan menggunakan GPU terdapat peningkatan 2 kali lebih cepat dibanding dengan menggunakan CPU. Artinya proses training dengan menggunakan GPU lebih efisien dibanding dengan CPU tanpa mengorbankan hasil training yang dihasilkan.

* 1. Hasil Testing

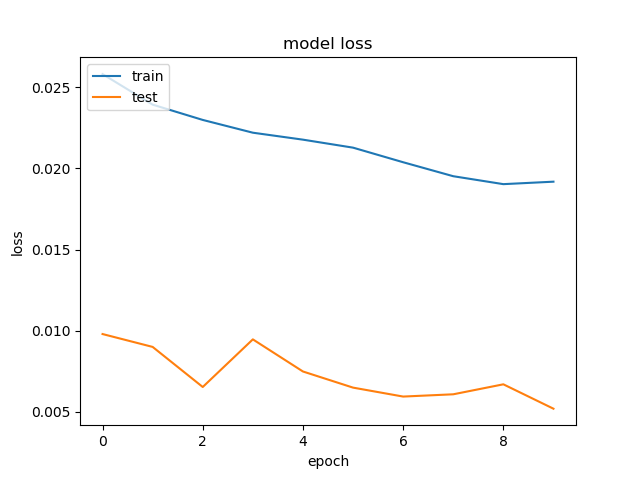
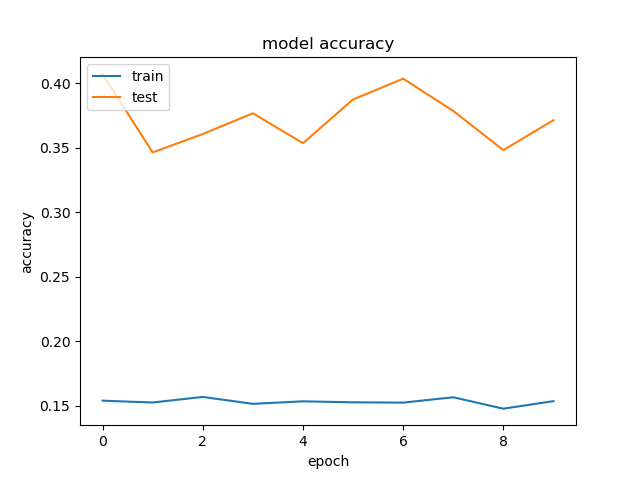
Pada tahap ini dilakukan testing data dengan dua cara, cara pertama yaitu dengan melihat accuracy dan loss value dari model yang dihasilkan. Setelah itu testing cara kedua dilakukan dengan cara membaca hasil model yang telah disimpan dalam format hdf5, lalu mengirimkannya melalui ke simulator dalam mode Autonomous Drive. Testing yang kedua ini untuk selanjutnya akan kita sebut dengan visual Testing. Dalam visual testing ini dikatakan berhasil jika mobil dapat berkendara secara mandiri dan dapat melewati 1 lap penuh tanpa menabrak halangan sama sekali.

* + 1. Data Testing GPU



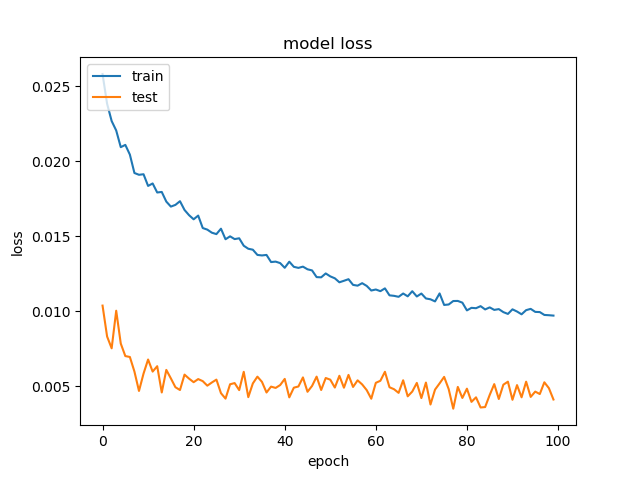
Gambar 4.5 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan GPU

* + 1. Data Testing CPU



Gambar 4.6 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan CPU

* + 1. Data Testing GPU 100 Epoch



Gambar 4.7 Akurasi dan Loss yang dihasilkan oleh model yang ditraining menggunakan GPU dengan 100 Epoch

* + 1. Interpretasi Data Testing

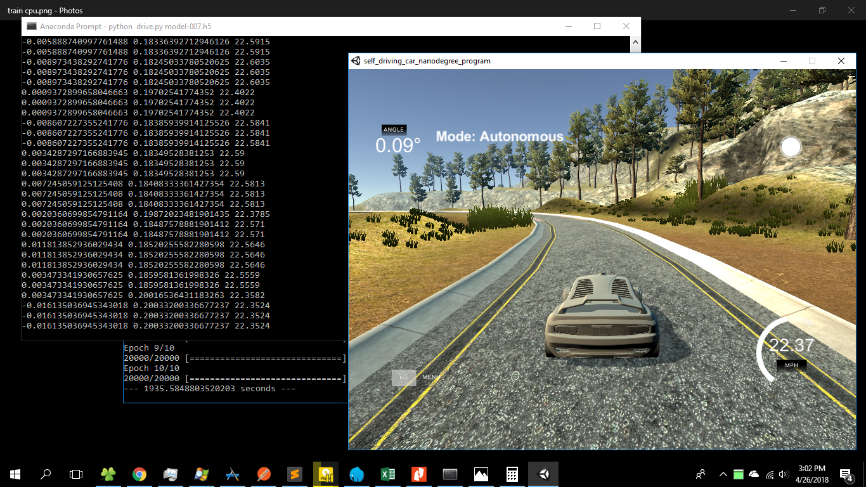
Dari data testing yang dihasilkan oleh GPU ataupun CPU, sekilas nampak sama. Dikarenakan loss value yang dihasilkan oleh data training baik oleh CPU ataupun GPU memiliki trend yang sama, yaitu semakin lama semakin menurun. Data ini sesuai dengan teori bahwa, semakin bertambahnya iterasi / epoch, maka semakin kecil loss value yang dihasilkan. Untuk data training antara CPU dan GPU memiliki kisaran angka di sekitar 0.020. Sedangkan untuk data testing memiliki nilai loss value di kisaran 0.005. Hal ini disebabkan oleh data training yang yang berisikan 80% total data, sedangkan data testing berisikan 20% total data. Kombinasi ini menyebabkan data training akan memiliki nilai loss value yang lebih besar dibandingkan dengan data testing.

Untuk data akurasi antara GPU dan CPU memiliki karakteristik yang terbalik dibanding dengan data loss value, jika pada loss value nilai data training lebih besar dibanding data testing, maka pada data akurasi, nilai akurasi data training akan lebih kecil dibanding data testing. Hal ini dapat terjadi karena loss value dan berbanding terbalik dengan accuracy.

Untuk data berikutnya yaitu data training yang dijalankan oleh GPU dengan 100 epoch. Langkah ini dilakukan untuk mengetahui bagaimana perilaku model ini dengan data epoch yang cukup besar, yaitu 100 epoch. Dari grafik yang dihasilkan, data loss value pada data training memiliki trend yang semakin lama semakin menurun, dan ini menunjukkan trend yang baik. Namun untuk data testing, secara visual data turun cukup drastis pada 20 epoch pertama saja, untuk epoch ke 21 hingga 100, data berfluktuasi pada nilai 0.005 saja. Hal ini dimungkinkan karena 100 epoch dinilai terlalu besar, atau data yang dibutuhkan untuk training kurang besar. Hal tersebut dapat divalidasi melalui visual testing, dimana pada test tersebut kita mengamati bagaimana performa self-driving car.

* + 1. Visual Testing

Pada testing ini, kita memperhatikan pada simulator apakah mobil yang dikendarai memiliki performa yang baik atau tidak. Definisi memiliki performa yang baik adalah mobil yang dapat mengendarai tanpa menabrak ataupun keluar dari jalur. Skema testing yang dilakukan yaitu memberikan pinalti setiap mobil menabrak pembatas jalan atau keluar dari jalur.



Gambar 4.8 Visual Testing

Dari visual testing, didapatkan hasil sebagai berikut:

|  |  |
| --- | --- |
| **Putaran** | **Pinalti** |
| 1 | 0 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |

Tabel 4.1 Hasil visual testing

Dari data tersebut dapat kita ketahui, bahwa secara umum keseluruhan putaran memiliki pinalti sebesar 1 poin, kecuali pada putaran pertama. Hal ini dapat terjadi karena pinalti terjadi pada transisi antara awal dan akhir dari sirkuit. Sehingga pada putaran pertama tidak mengalami pinalti sama sekali, sedangkan pada putaran kedua dan seterusnya memiliki pinalti yang sama, yaitu satu poin. Dari pengujian visual keseluruhan pinalti yang didapatkan berada pada area yang sama, ditunjukkan pada gambar 4.7.



Gambar 4.9 Area rawan pinalti

Dari data yang didapatkan dimungkinkan bahwa data yang digunakan untuk training kurang banyak, atau saat proses generate data user cenderung terlalu dekat dengan pembatas, sehingga proses autonomous drive berusaha meniru input yang telah diberikan oleh user, yaitu terlalu dekat dengan pembatas.

BAB V  
PENUTUP

1. BAB V PENUTUP
   1. KESIMPULAN

Berdasarkan dari hasil penelitian yang telah dilakukan, telah didapatkan satu model Deep Learning yang dapat digunakan untuk aplikasi Self-Driving Car. Model dibuat dengan 10 layer dimana 5 diantaranya adalah Convolutional Network, 4 diantaranya adalah Fully Connected Layer, serta Flatten Layer. Proses Training Deep Neural network dapat dilakukan dengan CPU ataupun GPU, keduanya memiliki hasil yang hampir sama, perbedaan terletak pada eksekusi waktu yang dibutuhkan, penggunaan GPU memakan waktu lebih sedikit dibanding dengan penggunaan CPU. Testing yang dihasilkan oleh training 100 epoch memiliki data yang cukup baik ditandai dengan loss value berkisar pada 0.015 pada data training, dan 0.005 pada data testing. Pengujian secara visual rata-rata mendapatkan pinalti 1 poin kecuali pada putaran pertama, dikarenakan data untuk training kurang banyak, atau penulis memberikan input yang tidak baik, sehingga model cenderung meniru perilaku yang tidak baik tersebut.

* 1. SARAN

Saran untuk penelitian berikutnya yaitu, diharapkan pada penelitian berikutnya dapat dicoba model-model deep learning yang lain. Diusahakan pada proses generate data cukup banyak dan sebagus mungkin, karena kualitas input yang baik akan menghasilkan output yang baik. Serta diharapkan dapat dibuat prototipe self driving car skala kecil.

DAFTAR PUSTAKA

Agatonovic-Kustrin, S. and Beresford, R. (2000) ‘Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research’, *Journal of Pharmaceutical and Biomedical Analysis*. Elsevier, 22(5), pp. 717–727. doi: 10.1016/S0731-7085(99)00272-1.

Bengio, Y. and Lecun, Y. (2007) ‘Scaling Learning Algorithms towards AI’, *Large Scale Kernel Machines*, (1), pp. 321–360. doi: 10.1.1.72.4580.

Bojarski, M. *et al.* (2016) ‘End to End Learning for Self-Driving Cars’, pp. 1–9. Available at: http://arxiv.org/abs/1604.07316.

Ceder, V. L. (2010) *The Quick Python Book*, *Python*. doi: 10.1016/j.jinsphys.2008.04.013.

Darrell Etherington (2017) *Udacity open sources its self-driving car simulator for anyone to use*, *Techcrunh.com*. Available at: https://techcrunch.com/2017/02/08/udacity-open-sources-its-self-driving-car-simulator-for-anyone-to-use/.

He, K. *et al.* (2015) ‘Delving deep into rectifiers: Surpassing human-level performance on imagenet classification’, *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter, pp. 1026–1034. doi: 10.1109/ICCV.2015.123.

*Keras: The Python Deep Learning library* (2015) *Keras.Io*. Available at: https://keras.io/%0Apapers3://publication/uuid/07295FF5-0ED2-40F6-ACD1-95ADC043E29C (Accessed: 30 April 2018).

Kingma, D. P. and Ba, J. (2014) ‘Adam: A Method for Stochastic Optimization’, pp. 1–15. doi: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

Krizhevsky, A., Sutskever, I. and Geoffrey E., H. (2012) ‘ImageNet Classification with Deep Convolutional Neural Networks’, *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pp. 1–9. doi: 10.1109/5.726791.

McClelland, C. (2017) *The Difference Between Artificial Intelligence, Machine Learning, and Deep Learning*, *Medium.com*. Available at: https://medium.com/iotforall/the-difference-between-artificial-intelligence-machine-learning-and-deep-learning-3aa67bff5991 (Accessed: 1 May 2018).

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986) ‘Learning representations by back-propagating errors’, *Nature*. Nature Publishing Group, 323, p. 533. Available at: http://dx.doi.org/10.1038/323533a0.

TensorFlow (2018) *An open-source machine learning framework for everyone*. Available at: https://www.tensorflow.org/ (Accessed: 30 April 2018).

LAMPIRAN