**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**

**SKRIPSI**

**Oleh:**

**ADWI ARIFIN**

**115090300111008**

****

**JURUSAN FISIKA**

**FAKULTAS MATEMATIKA DAN   
ILMU PENGETAHUAN ALAM**

**UNIVERSITAS BRAWIJAYA**

**MALANG**

**2017**

**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**

**SKRIPSI**

Sebagai Salah Satu Syarat Untuk Meraih Gelar   
Sarjana Sains dalam Bidang Fisika

**Oleh:**

**ADWI ARIFIN**

**115090300111008**

****

**JURUSAN FISIKA**

**FAKULTAS MATEMATIKA DAN   
ILMU PENGETAHUAN ALAM**

**UNIVERSITAS BRAWIJAYA**

**MALANG**

**2017**

LEMBAR PENGESAHAN SKRIPSI

**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**

Oleh:

ADWI ARIFIN

115090300111008

**Setelah di pertahankan di depan Majelis Penguji pada tanggal .......................... dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika**

|  |  |
| --- | --- |
| **Pembimbing I**  **Dr. Eng. Agus Naba,S.Si.,MT**  **NIP. 19720806 199512 1 001** | **Pembimbing II**  **Ahmad Nadhir, S.Si., MT., Ph.D. NIP.197412031999031002** |
| **Mengetahui,**  **Ketua Jurusan Fisika**  **FMIPA Universitas Brawijaya**  **Prof.Dr.rer.nat.Muhammad Nurhuda**  **NIP. 196409101990021001** | |

LEMBAR PERNYATAAN

**Saya yang bertanda tangan di bawah ini :**

**Nama : Adwi Arifin**

**NIM : 115090300111008**

**Jurusan : Fisika**

**Penulis Skripsi Berjudul :**

**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**

**Dengan ini menyatakan bahwa :**

1. **Isi dari proposal skripsi yang saya tulis dan saya buat adalah benar-benar karya saya sendiri dan tidak menjipliak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam proposal skripsi ini.**
2. **Apabila di kemudian hari ternyata proposal skripsi yang saya tulis terbukti hasil menjiplak, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

**Demikian pernyataan ini dibuat dengan segala kesadaran.**

**Malang, ……………………**

**Yang menyatakan,**

**Adwi Arifin**

**115090300111008**

**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**

ABSTRAK

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 penelitian ini penulis ingin mengusulkan bagaimana pemanfaatan Deep Learning spesifik pada kasus Self Driving Car, namun topik penelitian ini memiliki kompleksitas yang tinggi. Oleh karena itu, digunakan simulator mobil untuk menyederhanakan masalah. Metode yang digunakan pada penelitian ini yaitu dengan memanfaatkan convolutional layer, serta fully connected layer yang ditumpuk sedemikian rupa, jumlah hidden layer yang digunakan sebanyak 10 hidden layer. Loss function yang digunakan yaitu Mean Squared Error (MSE). Algoritma Optimasi yang digunakan yaitu Adam Optimizer. Hasil penelitian menunjukkan bahwa model yang dibuat lebih efisien ketika menggunakan GPU dibanding CPU. Untuk data dengan 10 epoch, memiliki jeda yang cukup besar antara data training dan data testing. Untuk data dengan 100 epoch, memiliki data jeda yang cukup kecil, namun untuk data testing trend yang dihasilkan hanya menurun pada 20 epoch pertama saja selanjutnya berfluktuatif pada nilai yang sama. Saat dilakukan pengujian secara visual, mobil dapat bergerak secara mandiri dan mendapatkan pinalti 1 kali setiap putaran kecuali pada putaran pertama.

**Kata kunci**: Machine Learning, Deep Learning, Self-Driving Car, Convolutional, Fully Connected

**DESIGN OF DEEP LEARNING MODEL FOR SELF DRIVING CAR SIMULATOR OPTIMIZED WITH GPU**

ABSTRACT

The development of self-driving car become more popular and every year got better, many company compete to make its own self-driving car capable. Its development cased by advance in Machine Learning, especially Deep Learning. In this study, author would like to propose utilization of Deep Learning specific to Self-Driving Car Problem, but this study have high complexity. Therefore, simulator is used instead of real Car. Mehotds used in this study is utilize convolutional layer to extract feature and fully connected layer to classify output to control vehicle. Total 10 hidden layer is used to compose the model. Loss function used in this study is Mean Squared Error (MSE). Algorithm to optimize the weights is Adam Optimize. The result shown by this study, training model with GPU is faster than CPU. Result with 10 epoch, have large difference between training set and testing set. Result with 100 epoch, have small difference between training set and testing set, but at the testing set loss value decreased at first 20 epoch, after that fluctuative at fixed value. Visual Testing used to verify if the model can be used to drive the vehicle. Result shown that in this study vehicle pinaltized 1 each lap except in the first lap.

**Keywords**: Machine Learning, Deep Learning, Self-Driving Car, Convolutional, Fully Connected

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul “**RANCANG BANGUN MODEL DEEP LEARNING UNTUK SIMULATOR SELF DRIVING CAR YANG DIOPTIMASI MENGGUNAKAN GPU**” sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika. Oleh karena itu, penulis mengucapkan rasa hormat dan terimakasih kepada:

1. Allah SWT atas nikmat dan karunia-Nya yang tak pernah putus sebagai pemilik ilmu.
2. Kedua orang tua penulis, Nugroho Priyo Santoso dan Siti Khilmiyah, yang selalu memberi doa, motivasi dan saran sehingga penulis dapat menyelesaikan skripsi.
3. Dr. Eng. Agus Naba,S.Si.,MT, sebagai dosen pembimbing I dan Ahmad Nadhir, S.Si., MT., Ph.D., sebagai dosen pembimbing II yang telah memberikan pengarahan dan masukan kepada penulis selama penyusunan skripsi.
4. Prof.Dr.rer.nat.Muhammad Nurhuda selaku ketua jurusan Fisika Universitas Brawijaya
5. Bapak dan Ibu dosen, laboran, serta karyawan jurusan Fisika atas didikan dan bantuan selama proses perkuliahan.
6. Medium, Hackernoon, StackOverflow, Siraj Raval, Udacity, Prof Andrew Ng, Coursera, Youtube, Google, serta Mendeley karena telah menunjang penulisan skripsi ini.
7. Teman-teman seperjuangan fisika, teman-teman Phy-Ro-C, dan semua sahabat saya serta semua pihak-pihak yang tidak dapat penulis sebutkan satu per satu atas kebersamaan, dorongan dan bantuan kepada penulis.
8. Pembaca yang telah meluangkan waktunya untuk membaca laporan skripsi ini.

Penulis memohon maaf apabila didapati kekurangan dalam penyusunan skripsi ini. Diharapkan skripsi ini dapat bermanfaat bagi kita semua.

Penulis

DAFTAR ISI

[LEMBAR PENGESAHAN SKRIPSI iii](#_Toc513807037)

[LEMBAR PERNYATAAN v](#_Toc513807038)

[ABSTRAK vii](#_Toc513807039)

[ABSTRACT ix](#_Toc513807040)

[KATA PENGANTAR xi](#_Toc513807041)

[DAFTAR ISI xiii](#_Toc513807042)

[DAFTAR GAMBAR xvi](#_Toc513807043)

[DAFTAR TABEL xvii](#_Toc513807044)

[BAB I PENDAHULUAN 18](#_Toc513807045)

[1.1 Latar Belakang 18](#_Toc513807046)

[1.2 Rumusan Masalah Penelitian 19](#_Toc513807047)

[1.3 Batasan Masalah 19](#_Toc513807048)

[1.4 Tujuan Penelitian 19](#_Toc513807049)

[1.5 Manfaat Penelitian 19](#_Toc513807050)

[BAB II TINJAUAN PUSTAKA 1](#_Toc513807051)

[2.1 Citra 1](#_Toc513807052)

[2.2 Model Warna RGB 2](#_Toc513807053)

[2.3 Artificial Intelegence 3](#_Toc513807054)

[2.4 Machine Learning 4](#_Toc513807055)

[2.5 Artificial Neural Network 4](#_Toc513807056)

[2.6 Deep Learning 5](#_Toc513807057)

[2.7 Loss Function 6](#_Toc513807058)

[2.8 Mean Squared Error 6](#_Toc513807059)

[2.9 Convolutional Layer 6](#_Toc513807060)

[2.10 Fully Connected Layer 9](#_Toc513807061)

[2.11 Adam Optimizer 9](#_Toc513807062)

[2.12 Python 10](#_Toc513807063)

[2.13 Keras 10](#_Toc513807064)

[2.14 TensorFlow 11](#_Toc513807065)

[2.15 CPU 11](#_Toc513807066)

[2.16 GPU 11](#_Toc513807067)

[BAB III METODE PENELITIAN 13](#_Toc513807068)

[3.1 Lokasi dan Waktu 13](#_Toc513807069)

[3.2 Alat dan Bahan 13](#_Toc513807070)

[3.3 Tahapan Penelitian 13](#_Toc513807071)

[3.3.1. Install Software 14](#_Toc513807072)

[3.3.2. Generate Data 15](#_Toc513807073)

[3.3.3. Training 18](#_Toc513807074)

[3.3.4. Testing 22](#_Toc513807075)

[BAB IV HASIL DAN PEMBAHASAN 25](#_Toc513807076)

[4.1 Hasil Generate Data 25](#_Toc513807077)

[4.2 Hasil Proses Training 26](#_Toc513807078)

[4.2.1. Training dengan GPU 27](#_Toc513807079)

[4.2.2. Training dengan CPU 28](#_Toc513807080)

[4.2.3. Perbandingan Training CPU dan GPU 28](#_Toc513807081)

[4.3 Hasil Testing 29](#_Toc513807082)

[4.3.1. Data Testing 10 Epoch 29](#_Toc513807083)

[4.3.2. Data Testing GPU 100 Epoch 30](#_Toc513807084)

[4.3.3. Visual Testing 31](#_Toc513807085)

[BAB V PENUTUP 33](#_Toc513807086)

[5.1 KESIMPULAN 33](#_Toc513807087)

[5.2 SARAN 33](#_Toc513807088)

[DAFTAR PUSTAKA 35](#_Toc513807089)

[LAMPIRAN 39](#_Toc513807090)

DAFTAR GAMBAR

[Gambar 2.1 Citra Lena dan citra kapal 1](#_Toc513797802)

[Gambar 2.2 Model Warna RGB 3](#_Toc513797803)

[Gambar 2.3 Artificial Neural Network 5](#_Toc513797804)

[Gambar 2.4 Persamaan MSE 6](#_Toc513797805)

[Gambar 2.5 Convolutional layer 7](#_Toc513797806)

[Gambar 2.6 Kepala unta 7](#_Toc513797807)

[Gambar 2.7 Operasi konvolusi beserta hasilnya. 8](#_Toc513797808)

[Gambar 2.8 Fully Connected Layer 9](#_Toc513797809)

[Gambar 2.9 Perbandingan algoritma Adam dan algoritma lainnya 10](#_Toc513797810)

[Gambar 3.1 Alur Penelitian 14](#_Toc513797811)

[Gambar 3.2. Daftar software yang diinstall, disimpan dalam file environment-gpu.yml 15](#_Toc513797812)

[Gambar 3.3 Tampilan Program Simulator Self Driving Car 17](#_Toc513797813)

[Gambar 3.4 Tampilan Pilih Folder 18](#_Toc513797814)

[Gambar 3.5 Skema proses training 19](#_Toc513797815)

[Gambar 3.5 Model Convolutional Neural Network 20](#_Toc513797816)

[Gambar 3.6 Tampilan Autonomous Drive 22](#_Toc513797817)

[Gambar 3.7 Skema pengiriman data 23](#_Toc513797818)

[Gambar 4.1 File CSV hasil training. 25](#_Toc513797819)

[Gambar 4.2 Gambar yang diambil dari tampak 26](#_Toc513797820)

[Gambar 4.3 Proses Training dengan GPU 27](#_Toc513797821)

[Gambar 4.4 Loss yang dihasilkan dengan 10 Epoch 30](#_Toc513797822)

[Gambar 4.5 Loss yang dihasilkan dengan 100 Epoch 30](#_Toc513797823)

[Gambar 4.6 Visual Testing 31](#_Toc513797824)

[Gambar 4.7 Area rawan pinalti 32](#_Toc513797825)

DAFTAR TABEL

[Tabel 4.1 Hasil Training dengan GPU 27](#_Toc513797908)

[Tabel 4.2 Hasil Training dengan CPU 28](#_Toc513797909)

[Tabel 4.3 Hasil visual testing 31](#_Toc513797910)

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 *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 latar belakang yang telah dijabarkan pada paragraf sebelumnya, penulis ingin mengusulkan bagaimana pemanfaatan *Deep Learning* spesifik pada kasus *Self Driving Car*, namun topik penelitian ini memiliki kendala. Kendala dari penelitian ini yaitu kompleksitas dan biaya yang cukup tinggi, karena dibutuhkan mobil dengan spesifikasi yang memadai, lalu dibutuhkan modifikasi pada mobil itu, hingga memasangkan perangkat lunak untuk sistem kontrol 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 dapat kita program agar dapat berkendara secara 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.

***(Halaman ini sengaja dikosongkan)***

BAB II  
TINJAUAN PUSTAKA

1. BAB IV HASIL DAN PEMBAHASAN
   1. Citra

Secara harafiah, citra (*image*) adalah gambar pada bidang dua dimensi. Gambar 2.1 adalah citra seorang gadis model yang bernama Lena, dan gambar di sebelah kanannya adalah citra kapal di sebuah pelabuhan. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam. Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat:

1. optik berupa foto,
2. analog berupa sinyal video seperti gambar pada monitor televisi,
3. digital yang dapat langsung disimpan pada suatu pita magnetik.

Citra yang dimaksudkan di dalam keseluruhan isi buku ini adalah “citra diam” (*still images*). Citra diam adalah citra tunggal yang tidak bergerak. Gambar 2.1 adalah dua buah citra diam. Untuk selanjutnya, citra diam kita sebut citra saja. Citra digital dapat memiliki berbagai model warna, salah satunya yaitu model warna RGB.

(a) Lena (b) Kapal

Gambar 2.1 Citra Lena dan citra kapal

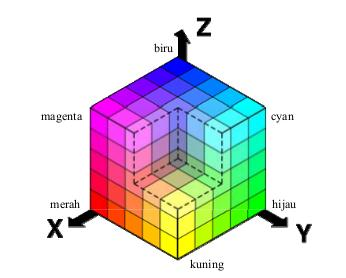
Citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara beruntun (*sequential*) sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut frame.Gambar-gambar yang tampak pada film layar lebar atau televisi pada hakikatnya terdiri atas ratusan sampai ribuan frame.

* 1. Model Warna RGB

Sesuai dengan namanya, RGB merupakan model warna yang menggunakan tiga warna dasar, yaitu merah (*red*), hijau (*green*), dan biru (*blue*), dimana masing-masing warna memiliki tingkat intensitas warna dari 0 sampai dengan 255, yang menyatakan tingkat keterangan warna mulai dari yang paling glap hingga yang paling terang. Memang tidak semua warna dapat diperoleh dengan menggunakan variasi nilai dari ketiga warna tersebut, akan tetapi sebagian warna yang diperlukan sudah tercakup dalam model warna RGB ini. Banyaknya warna yang dapat dibentuk oleh model warna RGB ini adalah 256 (intensitas piksel merah) x 256 (intensitas piksel hijau) x 256 (intensitas piksel biru), hasilnya lebih kurang enam belas juta tujuh ratus ribu warna, merupakan varian warna yang sangat banyak. Sebuah citra berwarna dapat dibentuk dengan mengatur nilai dari tingkat kecerahan gambar pada setiap piksel, dengan menggunakan warna-warna dasar seperti merah, hijau, dan biru dari cahaya yang terdeteksi. Salah satu caranya dapat dilakukan dengan menggunakan sebuah kamera digital, dimana sensornya dapat mengukur radiasi panjang gelombang dari merah, hijau, dan biru untuk semua titik dalam citra. Adapun arsitektur warna dari model warna RGB sebagai berikut:

1. Warna merah akan terbentuk jika piksel hijau dan biru bernilai 0, sedangkan piksel merah bernilai antara 1 - 254.
2. Warna hijau akan terbentuk jika piksel merah dan biru bernilai 0, sedangkan piksel hijau bernilai antara 1 - 254.
3. Warna biru akan terbentuk jika piksel merah dan hijau bernilai 0, sedangkan piksel biru bernilai antara 1 - 254.
4. Warna kuning akan terbentuk jika hanya piksel biru saja yang bernilai 0, sedangkan piksel merah dan hijau bernilai sama besar. 12
5. Warna magenta akan terbentuk jika hanya piksel hijau saja yang bernilai 0, sedangkan piksel merah dan biru bernilai sama besar.
6. Warna cyan akan terbentuk jika hanya piksel merah saja yang bernilai 0, sedangkan piksel hijau dan biru bernilai sama besar.
7. Warna keabuan akan terbentuk jika semua warna dasar RGB bernilai sama.
8. Warna hitam akan terbentuk jika semua warna dasar RGB bernilai 0.
9. Warna putih akan terbentuk jika semua warna dasar RGB bernilai 255.
10. Selain kesembilan penjelasan di atas, warna yang dibentuk merupakan varian model warna RGB.

Illustrasi dan penjelasan dari model warna RGB dapat dilihat pada gambar 2.2



Gambar 2.2 Model Warna RGB

* 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

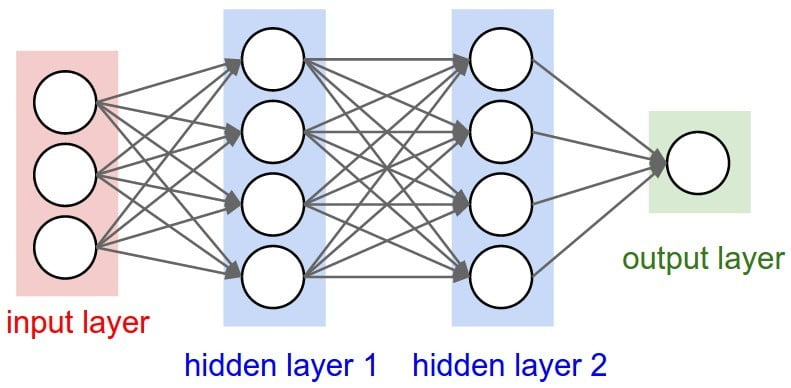
Machine Learning (ML) atau dalam bahasa indonesia disebut dengan pembelajaran mesin, adalah suatu pendekatan untuk dapat mewujudkan kecerdasan buatan. Pada tahun 1959, Arthur Samuel pernah mengemukakan pendapatnya tentang Machine Learning yaitu “suatu kemampuan untuk belajar tanpa harus diprogram secara 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 2.3 Artificial Neural Network

Gambar 2.3 merupakan skema dari *Artificial Neural Network*. Pada blok berwarna merah merupakan layer input, blok berwarna biru merupakan hidden layer, dan blok berwarna hijau merupakan layer output. Simbol berbentuk bulat merupakan *Node*. Simbol berupa garis dengan arah panah merupakan koneksi antar layer, dan dalam koneksi tersebut terdapat “*weight*” dan “*bias*”.

* 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)

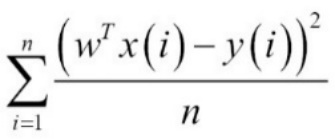
Deep Learning merupakan suatu teknik pembelajaran mesin dengan menggunakan Deep Neural Network. Teknik Deep Learning ini awalnya tidak cukup populer karena dengan jumlah hidden layer yang cukup besar, maka kompleksitas program semakin besar, sehingga waktu komputasi yang dibutuhkan pun cukup besar. Namun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

* 1. Loss Function

Loss function merupakan suatu fungsi yang digunakan untuk mencari tahu seberapa akurat data yang kita estimasi dengan dengan data yang sebenarnya. Terdapat banyak macam loss function, diantaranya Mean Squared Error, Root Mean Squared Error, dan banyak yang lainnya. Loss function dalam neural network dapat diukur dengan membandingkan hasil pada output layer dengan data output yang seharusnya.

* 1. Mean Squared Error

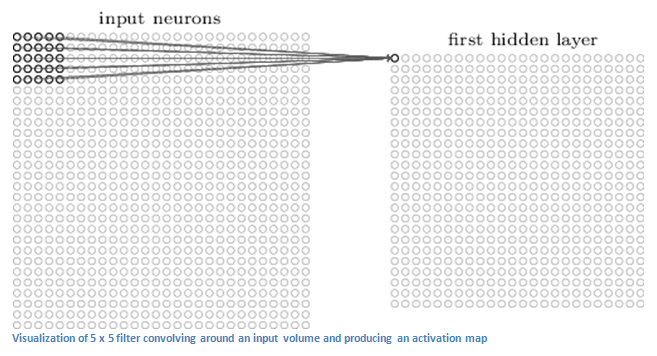
Mean Squared Error adalah suatu besaran yang menunjukkan seberapa dekat data perkiraan dengan data yang sebenarnya. Hal ini dapat dilakukan dengan menghitung selisih dari data perkiraan dengan dengan data sebenarnya, selisih ini kita sebut dengan error, lalu mengkuadratkan error yang dihasilkan. Proses pengkuadratan ini penting karena untuk menghilangkan nilai-nilai negatif. Semakin kecil nilai yang dihasilkan, menunjukkan seberapa akurat data perkiraan yang kita hitung. Persamaan Mean Squared Error dapat ditunjukkan pada gambar 2.4.



Gambar 2.4 Persamaan MSE

* 1. Convolutional Layer

Convolutional Layer dalam Deep Learning merupakan suatu layer yang bertugas untuk mengekstraksi sebuah fitur. Convolutional layer dapat dimisalkan dengan memberi cahaya senter pada pada pojok sebuah citra, dengan jarak tertentu, maka hanya akan sebagian dari citra saja yang terkena cahaya senter tersebut, bagian lain dari citra tidak terpapar cahaya dari senter tersebut. Dalam machine learning, bagian yang terkena cahaya dari senter disebut dengan *filter* (ada juga yang menyebut *kernel*), sedangkan hasil dari operasi konvolusi ini disebut dengan *feature map*. Contoh proses operasi konvolusi ditunjukkan pada gambar 2.5 (Deshpande, 2017).

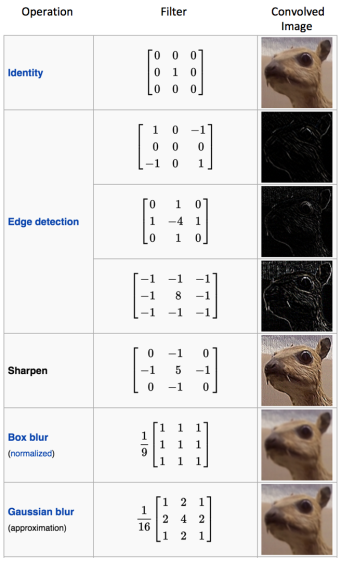


Gambar 2.5 Convolutional layer

Untuk dapat memahami fungsi dari layer konvolusi, maka digunakan contoh gambar pada gambar 2.6, yaitu gambar sebuah kepala unta. Gambar 2.6 jika dikenakan suatu filter, maka dapat menghasilkan gambar yang sesuai dengan filternya. Contoh gambar 2.7 merupakan sebagian contoh gambar kepala unta yang dikenai beberapa filter beserta hasilnya. (Ujjwal Karn, 2016)



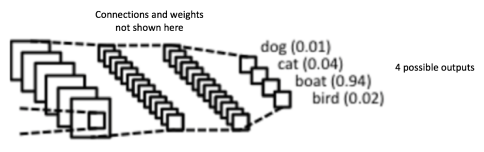
Gambar 2.6 Kepala unta



Gambar 2.7 Operasi konvolusi beserta hasilnya.

* 1. Fully Connected Layer

Fully connected layer merupakan Multi Layer Perceptron. Pada arsitektur tersebut, seluruh neuron yang terdapat pada layer sebelumnya terkoneksi dengan seluruh neuron yang terdapat pada layer selanjutnya. Gambar 2.8 merupakan contoh skema dari *fully connected layer*.

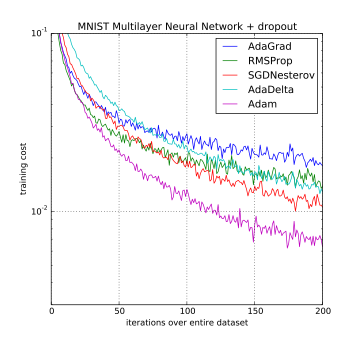


Gambar 2.8 Fully Connected Layer

Fully connected layer dapat difungsikan sebagai klasifikasi. Contoh klasifikasi dapat dilihat pada gambar 2.8, bahwa output dari fully connected layer menghasilkan 4 macam kemungkinan output. Selain sebagai klasifikasi, *fully connected layer* merupakan cara termudah untuk mempelajari kombinasi fitur-fitur yang dihasilkan oleh *convolutional layer*. Hasil dari fitur-fitur yang dihasilkan oleh convolutional layer mungkin saja sudah dapat digunakan sebagai klasifikasi, namun kombinasi fitur-fitur tersebut mungkin saja dapat menghasilkan hasil yang lebih baik (Ujjwal Karn, 2016).

* 1. Adam Optimizer

Adam Optimizer merupakan salah satu algoritma optimasi untuk dapat mengupdate bobot dari neural network. Algoritma ini dapat digunakan untuk permasalahan-permasalah *stocastic* *gradient descent* untuk mengupdate bobot yang ditraining. Adam bukan merupakan nama seseorang, ataupun singkatan dari orang, nama Adam ini diambil dari *Adaptive Moment Estimation*. Terdapat banyak algoritma yang dapat menyelesaikan permasalahan *stocastic gradient descent* diantaranya adalah RMSProp, SGDNesterov, AdaDelta dan Adam. Pada salah satu penelitian menunjukkan Adam Optimizer merupakan algoritma terbaik diantara Algoritma-algoritma yang telah disebutkan sebelumnya. Gambar 2.9 menunjukkan perbandingan performa algoritma untuk menyelesaikan *Stocastic Gradient Descent* (Kingma and Ba, 2015)



Gambar 2.9 Perbandingan algoritma Adam dan algoritma lainnya

* 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.

* 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.

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 beberapa 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 tampak pada gambar 3.2.



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)



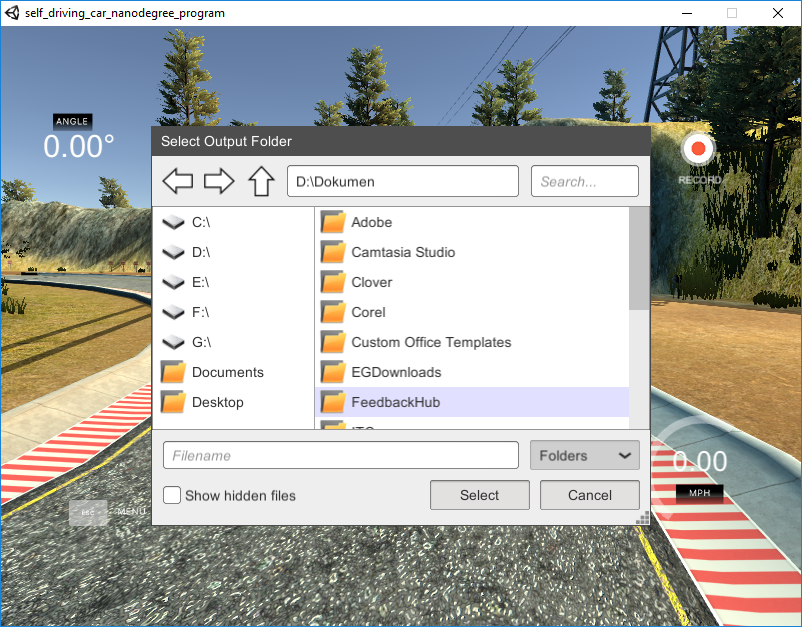
(c)

Gambar 3.3 Tampilan Program Simulator Self Driving Car

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

Gambar 3.3 menunjukkan tampilan program simulator self-driving car yang dibuat oleh penyedia kursus online yang bernama Udacity. Program tersebut bersifat open source, dan bisa digunakan untuk kepentingan apapun. Pada gambar 3.3 terdapat tiga macam tampilan program simulator self-driving car. Gambar 3.3 butir (a) menunjukkan tampilan awal, pada tampilan tersebut terdapat tombol training mode, autonomous mode, controls, serta dua pilihan jalur lintasan yang bisa dipilih di bagian bawah. Untuk penelitian ini kita batasi pada lintasan pertama saja. Gambar 3.3 butir (b) menunjukkan halaman informasi yang berisikan kontrol apa saja yang bisa diberikan pada simulator ini. Kontrol yang tersedia yaitu tombol [W] untuk maju, tombol [S] untuk mundur, tombol [A] untuk berbelok ke kiri, tombol [D] untuk berbelok ke kanan, tombol [R] untuk melakukan perekaman pada training mode, serta tombol [Esc] untuk kembali ke halaman awal. Selain menggunakan keyboard, simulator ini juga dapat menerima input berupa drag mouse yang difungsikan sama dengan belok kanan maupun belok kiri, melakukan drag ke arah kiri akan sama dengan belok ke kiri, sedangkan drag ke arah kanan akan sama dengan belok ke kanan. Gambar 3.3 butir (c) menunjukkan tampilan training mode. Pada tampilan training mode, kita bisa menggerakkan mobil dengan input keyboard ([W], [A], [S], [D]) atau pun drag mouse, serta input keyboard [R] untuk melakukan perekaman data.

Saat telah masuk dalam progam simulator 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. Tampilan pemilihan folder yang terjadi ditunjukkan pada gambar 3.4



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. Skema proses training ditunjukkan pada gambar 3.5.

Citra Digital

Deep Learning

Target

Update weight

MSE

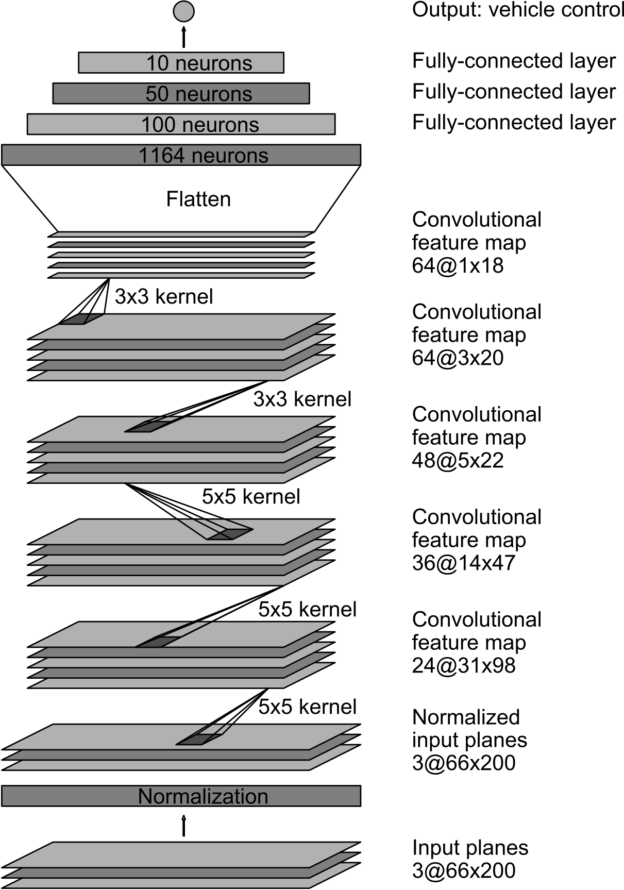
Gambar 3.5 Skema proses training

Pada gambar 3.5 telah ditunjukkan skema proses training yang diusulkan, data input yang digunakan merupakan sebuah citra digital yang akan dimasukkan ke dalam model Deep Learning, model Deep Learning tersebut menghasilkan sebuah output yang mewakili kontrol dari mobil, yaitu *steering wheel* dengan ketentuan sebagai berikut:

1. Jika out bernilai nol, maka kontrol yang diberikan adalah lurus.
2. Jika out bernilai bilangan positif, maka kontrol yang diberikan adalah belok kanan
3. Jika out bernilai bilangan negatif, maka kontrol yang diberikan adalah belok kiri

Simbol lingkarang dengan label e, merupakan representasi error yang dihasilkan. Error didapatkan dengan membandingkan hasil out dan target yang didapat dari tabel input, lalu menghitungnya dengan *Mean Squared Error*. Hasil error ini, lalu digunakan untuk mengupdate nilai *weight* pada deep learning. Algoritma yang digunakan untuk mengupdate nilai weight adalah *Adam Optimizer*, hal ini dilakukan karena menurut (Kingma and Ba, 2015), algoritma ini merupakan algoritma terbaik yang didapat pada penelitian yang dijalankan.

Dalam penelitian digunakan model Deep Learning yang terdiri dari Convolutional Layer dan Fully Connected Layer. 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 dengan memanfaatkan Fully Connected Layer. Konfigurasi jaringan yang digunakan dalam penelitian ini adalah 10 hidden layer. Konfigurasi ini diinspirasi dari konfigurasi yang telah dipaparkan oleh NVIDIA (Bojarski *et al.*, 2016).



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.

Secara umum, proses neural network pada penelitian ini dibagi menjadi 4 tahap, yaitu normalisasi, filter, flatten, dan klasifikasi. Tahap pertama yaitu normalisasi, tahap ini memastikan bahwa citra digital berada pada rentang yang diinginkan. Selanjutnya proses filter, pada tahap ini diharapkan neural network dapat mengekstraksi fitur-fitur abstrak yang dapat memaksimalkan performa dari neural network. Alasan digunakan hingga lima kali proses konvolusi adalah agar dapat mengekstrak fitur namun tetap menjaga dimensi ruang dari citra digital. Terdapat penelitian yang memaparkan bahwa lebih baik menggunakan dua tahap atau lebih proses konvolusi agar dapat mendapat performa yang maksimal (LeCun, Bengio and Hinton, 2015). Setelah proses konvolusi ini selesai dilakukan, maka diubah fitur fitur yang memiliki dua dimensi menjadi satu baris / kolom dengan bantuan proses flatten layer. Setelah proses flatten layer selesai dilakukan, maka proses klasifikasi dilakukan dengan memanfaatkan *Fully Connected Layer*, proses klasifikasi ini dilakukan melalui 3 tahap, karena sesuai dengan jurnal yang dipaparkan oleh LeCun, bahwa seiring bertambahnya layer dapat meningkatkan performa neural network (LeCun, Bengio and Hinton, 2015).

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.

* + 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

Pada gambar 3.6 ditunjukkan tampilan program simulator self-driving car pada mode Autnomous. Pada tampilan ini, mobil akan diam saja, dan masih bisa digerakkan secara manual. Pada tampilan ini, selain simulator dapat menerima input secara manual, simulator juga dapat menerima input berupa kontrol yang dikirimkan melalui socket 4567. Ketika ada sambungan yang terhubung pada socket tersebut, maka program akan bertindak sebagai input otomatis. Skema pengiriman data ditunjukkan pada gambar 3.7.

drive.py

Simulator

Saved Model

Load Model

socket 4567

emit

receive

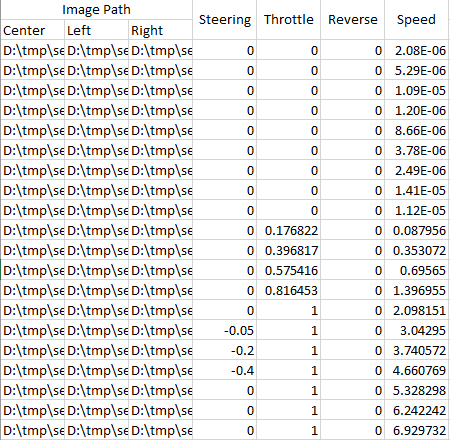
Gambar 3.7 Skema pengiriman data

***(Halaman ini sengaja dikosongkan)***

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. Gambar 4.1 merupakan cuplikan file csv sedangkan Gambar 4.2 merupakan cuplikan hasil gambar yang dihasilkan.



Gambar 4.1 File CSV hasil training.

Dari gambar 4.1 dapat diketahui bahwa 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)

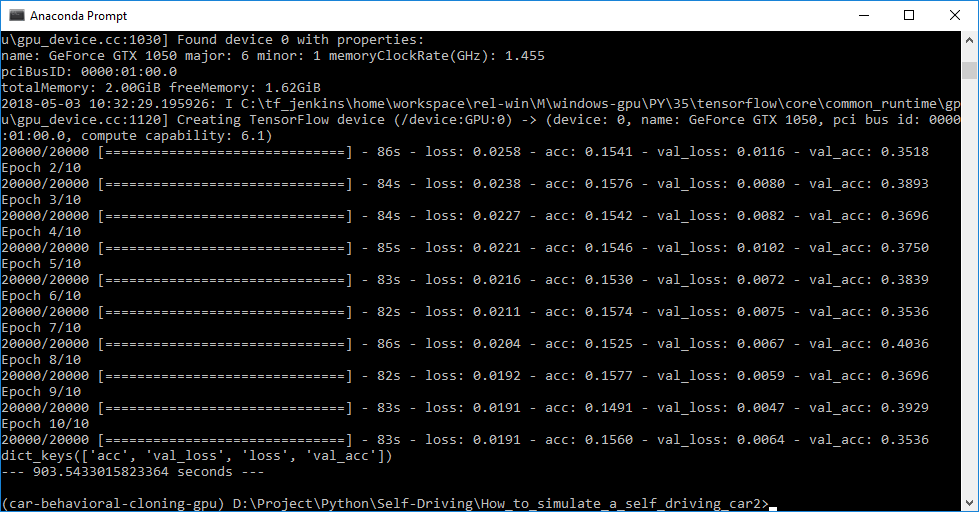
Gambar 4.2 Gambar yang diambil dari tampak

(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. Tujuannya adalah untuk mengetahui apakah training dengan dengan menggunakan GPU meningkatkan proses training secara siginiikan dan apakah training menggunakan GPU mendapatkan hasil yang lebih baik dibanding dengan CPU. Contoh hasil proses training dapat dilihat pada gambar 4.3 Pada gambar tersebut terdapat informasi berapa epoch yang telah dijalankan, waktu yang dibutuhkan untuk menjalankan satu epoch, akurasi, dan loss.



Gambar 4.3 Proses Training dengan GPU

* + 1. Training dengan GPU

Training yang pertamakali dilakukan oleh penulis adalah training dengan menggunakan GPU. Dan hasilnya ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil Training dengan GPU

|  |  |  |  |
| --- | --- | --- | --- |
| **Epoch** | **Time** | **Accuracy** | **Loss** |
| 1 | 86 | 0.1541 | 0.0258 |
| 2 | 84 | 0.1576 | 0.0238 |
| 3 | 84 | 0.1542 | 0.0227 |
| 4 | 85 | 0.1546 | 0.0221 |
| 5 | 83 | 0.1530 | 0.0216 |
| 6 | 82 | 0.1574 | 0.0211 |
| 7 | 86 | 0.1525 | 0.0204 |
| 8 | 82 | 0.1577 | 0.0192 |
| 9 | 83 | 0.1491 | 0.0191 |
| 10 | 83 | 0.1560 | 0.0191 |
| Total | 838 |  | |

Pada Tabel 4.1, dapat kita ketahui bahwa proses training dengan menggunakan GPU memakan waktu sebanyak 838 detik atau 13 menit 58 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 sebesar 0.0191 dan akurasi sebesar 0.1491.

* + 1. Training dengan CPU

Training berikutnya adalah training dengan menggunakan CPU, dan hasilnya dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Training dengan CPU

|  |  |  |  |
| --- | --- | --- | --- |
| **Epoch** | **Time** | **Accuracy** | **Loss** |
| 1 | 191 | 0.1541 | 0.0258 |
| 2 | 190 | 0.1527 | 0.0239 |
| 3 | 193 | 0.1570 | 0.0230 |
| 4 | 191 | 0.1516 | 0.0222 |
| 5 | 205 | 0.1536 | 0.2180 |
| 6 | 197 | 0.1528 | 0.0213 |
| 7 | 232 | 0.1526 | 0.0204 |
| 8 | 211 | 0.1567 | 0.0195 |
| 9 | 291 | 0.1479 | 0.0190 |
| 10 | 229 | 0.1537 | 0.0192 |
| Total | 2130 |  | |

Pada Tabel 4.2, dapat kita ketahui bahwa proses training dengan menggunakan CPU memakan waktu sebanyak 2130 detik atau 35 menit 30 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-9 dimana pada epoch tersebut didapatkan nilai loss sebesar 0.0190, dan akurasi sebesar 0.1479.

* + 1. Perbandingan Training CPU dan GPU

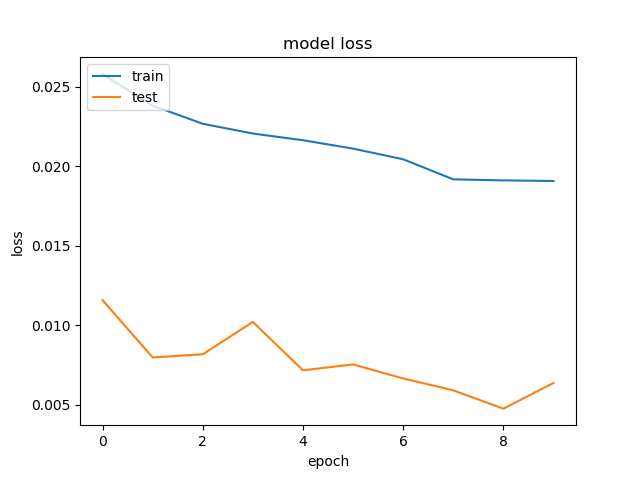
Secara umum training dengan CPU maupun GPU menghasilkan data yang tidak jauh berbeda, dapat dilihat pada loss yang dihasilkan berkisar pada kisaran angka yang sama, yaitu 0.0191 pada training dengan menggunakan GPU sedangkan 0.0190 pada training dengan menggunakan CPU. Namun dari kedua proses training tersebut, waktu yang dibutuhkan berbeda. Pada proses training dengan CPU memakan waktu sebanyak 35 menit 30 detik, sedangkan pada GPU memakan waktu sebanyak 13 menit 58 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. Untuk poin-poin selanjutnya, data yang ditampilkan hanya data yang ditraining dengan GPU.

* 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.

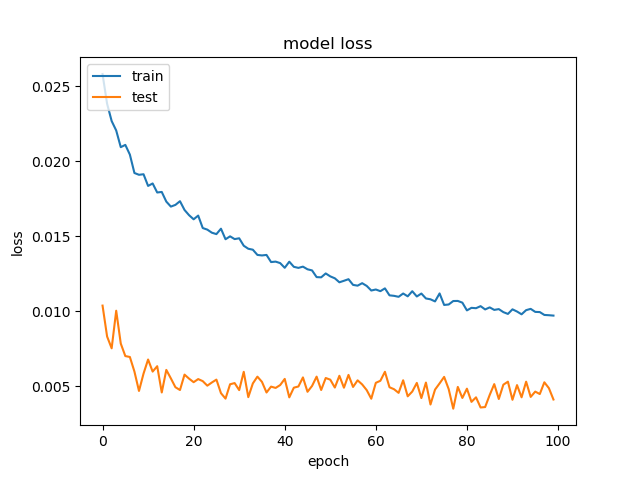
* + 1. Data Testing 10 Epoch

Data testing dengan GPU ditunjukkan pada dan gambar 4.5. Pada gambar 4.5 tersebut data ditunjukkan dalam bentuk grafik. Grafik yang ditunjukkan dalam gambar 4.5 merupakan grafik antara epoch dengan loss yang dihasilkan. Pada Gambar 4.5 dapat kita ketahui bahwa nilai loss yang dihasilkan seiring bertambahnya epoch maka akan semakin menurun. Pada data training maupun data test memiliki trend yang sama-sama saling turun, namun antara data training dan data testing memiliki jarak yang cukup jauh yaitu berkisar sekitar 0.015. Perbedaan yang cukup jauh tersebut dapat dikarenakan nilai epoch yang kecil atau karena jumlah data yang dibutuhkan untuk proses training kurang banyak. Untuk itu perlu dikonfirmasi bagaimana jika dilakukan proses training lagi dengan data yang sama, namun dengan jumlah epoch yang lebih besar.



Gambar 4.4 Loss yang dihasilkan dengan 10 Epoch

* + 1. Data Testing GPU 100 Epoch

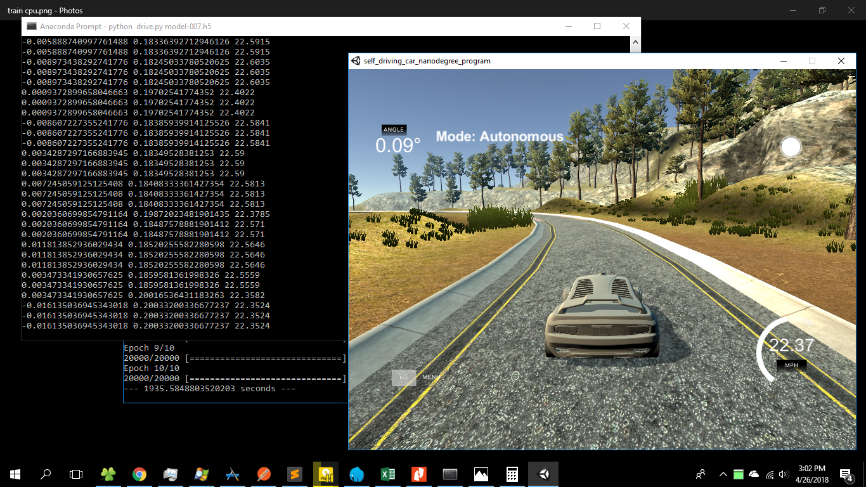


Gambar 4.5 Loss yang dihasilkan dengan 100 Epoch

Gambar 4.5 merupakan grafik yang dihasilkan proses training dengan 100 epoch. Pada gambar 4.5 dapat dilihat bahwa data training seiring bertambahnya epoch, maka loss yang dihasilkan semakin kecil. Pada data testing data yang dihasilkan juga menunjukkan trend yang sama, yaitu semakin lama semakin menurun, namun trend ini hanya terjadi 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 jumlah data yang dibutuhkan untuk proses training kurang banyak. 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. Visual Testing ditunjukkan pada gambar 4.6. Dari visual testing, didapatkan hasil sebagaimana ditunjukkan pada Tabel 4.3.



Gambar 4.6 Visual Testing

Tabel 4.3 Hasil visual testing

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

Dari data tabel 4.1 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 4.7 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.

***(Halaman ini sengaja dikosongkan)***

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/.

Deshpande, A. (2017) *A Beginner’s Guide to Deep Convolutional Neural Networks (CNNs)*. Available at: https://deeplearning4j.org/convolutionalnetwork.html#a-beginners-guide-to-deep-convolutional-neural-networks-cnns (Accessed: 1 May 2018).

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. L. (2015) ‘Adam: A Method for Stocastic Optimization’, pp. 1–15.

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.

LeCun, Y. A., Bengio, Y. and Hinton, G. E. (2015) ‘Deep learning’, *Nature*. doi: 10.1038/nature14539.

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).

Ujjwal Karn (2016) *An Intuitive Explanation of Convolutional Neural Networks*, *The Data Science Blog*. Available at: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/ (Accessed: 1 May 2018).

LAMPIRAN