

**UNIVERSITAS GUNADARMA
FAKULTAS TEKNOLOGI INDUSTRI**



**ESTIMASI POSE TIGA DIMENSI DARI GAMBAR
MONOKULER MENGGUNAKAN DEEP NEURAL
NETWORK**

Disusun oleh:

Nama	: Denilson
NPM	: 51416815
Jurusan	: Teknik Informatika
Pembimbing	: Dr. Dharmayanti, ST., MMSI.

**Diajukan Guna Melengkapi Sebagian Syarat
Dalam Mencapai Gelar Sarjana Strata Satu (S1)**

Jakarta

2020

LEMBAR PENGESAHAN

Komisi Pembimbing

No	Nama	Kedudukan
1	Dr. Dharmayanti, ST., MMSI.	Ketua
2	DIGANTI NAMA PENGUJI 2	DIGANTI JABATAN PENGUJI 2
3	DIGANTI NAMA PENGUJI 3	DIGANTI JABATAN PENGUJI 3

Tanggal Sidang : tgl/bln/thn

Panitia Ujian

No	Nama	Kedudukan
1	DIGANTI NAMA PENGUJI 1	DIGANTI JABATAN PENGUJI 1
2	DIGANTI NAMA PENGUJI 2	DIGANTI JABATAN PENGUJI 2
3	DIGANTI NAMA PENGUJI 3	DIGANTI JABATAN PENGUJI 3
4	DIGANTI NAMA PENGUJI 4	DIGANTI JABATAN PENGUJI 4
5	DIGANTI NAMA PENGUJI 5	DIGANTI JABATAN PENGUJI 5

Tanggal Lulus : tgl/bln/thn

Mengetahui,

Pembimbing

Bagian Sidang Sarjana

(Dr. Dharmayanti, ST., MMSI.)

(NAMA BAGIAN SARJANA)

LEMBAR ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini,

Nama : Denilson
NPM : 51416815
Judul Penulisan Ilmiah : Estimasi Pose Tiga Dimensi dari Gambar Monokuler
Menggunakan Deep Neural Network
Tanggal Sidang : tanggal
Tanggal Lulus : tanggal

menyatakan bahwa tulisan ini adalah merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah, etika yang berlaku. Mengenai isi dan tulisan adalah merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Jakarta, April 2020

Denilson

ABSTRAKSI

Denilson, 51416815

ESTIMASI POSE TIGA DIMENSI DARI GAMBAR MONOKULER MENGUNAKAN DEEP NEURAL NETWORK

Tugas Akhir. Jurusan Teknik Informatika, Fakultas Teknologi Industri,

Universitas Gunadarma, 2020

Kata Kunci: Estimasi Pose, Gambar Monokuler, Jaringan Saraf Tiruan,
Pemelajaran Dalam, Visi Komputer

(xiii + 38 + lampiran)

Perkembangan teknologi digital yang pesat baik pada aplikasi atau ilmu pengetahuan dapat menghasilkan rekam jejak digital yang bermanfaat. Jumlah data digital yang tersedia sangat banyak dan diprediksi akan semakin bertambah. Salah satu penggunaan data adalah membuat suatu fungsi pemetaan yang mencari korelasi antara suatu domain ke domain lainnya dengan menggunakan data terkait sebagai acuan dasar. Data digital berbentuk rangkaian gambar atau video merupakan data yang bersifat laten yang berarti data tersebut memiliki informasi semantik yang tersembunyi. Penelitian ini membahas pembuatan sebuah fungsi yang memetakan gambar dua dimensi terhadap titik kunci pose tiga dimensi yang bersifat laten menggunakan permodelan *deep neural network*. Perangkat lunak yang dibangun dengan pemrosesan data, perancangan arsitektur model, pemelajaran model secara mandiri, dan menampilkan visualisasi penggunaan model. Arsitektur model yang digunakan terdiri dari beberapa blok *residual network* yang menambahkan *input* terhadap *output* masing-masing blok. Hasil dari uji coba menjelaskan bahwa teori dan data yang dipakai benar dan penggunaan aplikasi terhadap data baru berjalan sesuai prediksi.

Daftar Pustaka (1986-2020)

ABSTRACT

Denilson, 51416815

THREE DIMENSIONAL POSE ESTIMATION FROM MONOCULAR IMAGE USING DEEP NEURAL NETWORK

Essay. Informatics Engineering, Faculty of Industrial Technology,
Gunadarma University, 2020

Keywords: Artificial Neural Network, Computer Vision, Deep Learning,
Monocular Image, Pose Estimation

(xiii + 38 + attachments)

Digital technologies have been developed rapidly in application and science may produce digital track records that are actually useful. Digital data are available in a huge number and are predicted to increase. One way to utilize this data is to create a mapping function that finds a correlation between domains from the data itself as a reference. Digital data in form of sequence of images or videos are latent which mean data itself has some hidden semantic meanings. This research is about making a mapping function that maps two dimensional images into three dimensional human pose keypoints using deep neural network modeling. The software is built in steps that involve data preprocessing, model architecture design, self-training deep neural network, and visualization. The model consists of some blocks of residual networks that sum up its inputs and outputs. The result from testing explains that the theories and data are correct and runs correctly using new data as input.

Bibliography (1986-2020)

KATA PENGANTAR

Segala puji dan syukur penulis ucapkan ke hadirat Tuhan Yang Maha Esa yang telah memberikan berkat, anugerah dan karunia yang melimpah, sehingga penulis dapat menyelesaikan tugas akhir ini pada waktu yang telah ditentukan.

Tugas akhir ini disusun guna melengkapi sebagian syarat untuk memperoleh gelar Sarjana Teknik Informatika Universitas Gunadarma. Adapun judul tugas akhir ini adalah "Estimasi Pose Tiga Dimensi Dari Gambar Monokuler Menggunakan Deep Neural Network".

Walaupun banyak kesulitan yang penulis harus hadapi ketika menyusun tugas akhir ini, namun berkat bantuan dan dorongan dari berbagai pihak, akhirnya tugas akhir ini dapat diselesaikan dengan baik. Untuk itu penulis tidak lupa mengucapkan terima kasih kepada:

1. Prof. Dr. E. S. Margianti, SE., MM., selaku rektor Universitas Gunadarma.
2. Prof. Dr. Ir. Bambang Suryawan, MT., selaku Dekan Fakultas Teknologi Industri Universitas Gunadarma.
3. Prof. Dr. - Ing. Adang Suhendra, SSi., SKom., MSc., selaku Ketua Jurusan Teknik Informatika Universitas Gunadarma.
4. Dr. Edi Sukirman, SSi., MM., selaku Kepala Bagian Sidang Ujian Universitas Gunadarma.
5. Dr. Dharmayanti, ST., MMSI., sebagai dosen pembimbing penulis yang ditengah-tengah kesibukannya telah membimbing penulis sehingga penulisan ini dapat diselesaikan.
6. Keluarga yang selalu mendukung dan terus memberikan motivasi.
7. Semua pihak yang terlibat dalam membantu penyelesaian Tugas Akhir ini.

Sebagai manusia biasa yang tak luput dari kesalahan, maka penulis meminta maaf atas segala kekurangan dan keterbatasan dalam penyusunan tugas akhir ini. Penulis sadari bahwa penulisan ini masih jauh dari sempurna, disebabkan karena berbagai keterbatasan yang penulis miliki. Untuk itu penulis mengharapkan kritik dan saran yang bersifat membangun untuk menjadi perbaikan di masa yang akan datang.

Akhir kata, penulis berharap penulisan ini dapat bermanfaat bagi semua pihak dan bagi penulis pribadi khususnya, serta dapat digunakan sebagaimana mestinya.

Jakarta, April 2020

Penulis

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PENGESAHAN	ii
LEMBAR ORIGINALITAS DAN PUBLIKASI	iii
ABSTRAKSI	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	ix
DAFTAR TABEL	x
DAFTAR GAMBAR	xii
DAFTAR LAMPIRAN	xiii
BAB I : PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	2
1.5 Metode Penelitian	2
1.6 Sistematika Penulisan	4
BAB II : TINJAUAN PUSTAKA	5
2.1 Teorema Penaksiran Universal	5
2.2 Jaringan Saraf	5
2.3 Jaringan Saraf Tiruan	7
2.4 Fungsi Aktivasi	8
2.5 <i>Residual Network</i>	9
2.6 Optimisasi Model	9
2.6.1 <i>Backpropagation</i>	10
2.6.2 <i>Learning Rate</i>	11
2.6.3 <i>Mean Squared Error</i>	11

2.7	Estimasi Pose Dua Dimensi	11
2.8	Estimasi Pose Tiga Dimensi	13
2.9	PyTorch	14
2.10	<i>Unified Modeling Language</i>	14
BAB III	: METODOLOGI PENELITIAN	16
3.1	Gambaran Umum	16
3.2	Kerangka Penelitian	17
3.3	Tahap Praproduksi	17
3.3.1	Analisis Kebutuhan Proyek	17
3.3.2	Analisis Struktur Proyek	18
3.3.3	Analisis Data	20
3.4	Tahap Produksi	22
3.4.1	Prapemrosesan Data Pelatihan	22
3.4.2	Arsitektur Model	24
3.4.3	Pemelajaran Model	25
3.5	Tahap Uji Coba	26
3.5.1	Prapemrosesan Data Inferensi	27
3.5.2	OpenPose	27
3.5.3	Inferensi Model	29
3.5.4	Visualisasi	31
BAB IV	: HASIL DAN PEMBAHASAN	32
4.1	Hasil Pemelajaran Model	32
4.2	Kekurangan Aplikasi	32
BAB V	: PENUTUP	35
5.1	Kesimpulan	35
5.2	Saran	35
DAFTAR PUSTAKA	36
LAMPIRAN	L1

DAFTAR TABEL

2.1	Simbol-Simbol <i>Activity Diagram</i>	15
3.1	Spesifikasi Perangkat Keras	18
3.2	Spesifikasi Perangkat Lunak	18
3.3	Data Pemelajaran Model	20

DAFTAR GAMBAR

2.1	Ilustrasi Jaringan Saraf Manusia	6
2.2	Ilustrasi Sel Saraf Manusia	6
2.3	Ilustrasi Jaringan Saraf Tiruan	7
2.4	Ilustrasi Sel Saraf Tiruan	8
2.5	<i>Rectified Linear Unit</i>	8
2.6	<i>Residual Block</i>	9
2.7	<i>Gradient Descent</i>	10
2.8	Skema <i>Backpropagation</i>	10
2.9	Ilustrasi Perbedaan <i>Learning Rate</i>	11
2.10	Pendekatan <i>Top-Down</i> dan <i>Bottom-Up</i>	12
2.11	Pencarian Pose Tiga Dimensi Lokal	13
2.12	Pencarian Pose Tiga Dimensi Global	14
3.1	Kerangka Penelitian	17
3.2	Jupyter Lab	18
3.3	Struktur Direktori	19
3.4	<i>Frame</i> 00077	21
3.5	<i>Frame</i> 00173	21
3.6	<i>Frame</i> 00232	21
3.7	<i>Activity Diagram</i> Prapemrosesan Data	23
3.8	Arsitektur Lapisan ResLinear	24
3.9	Arsitektur Model	25
3.10	<i>Activity Diagram</i> Pemelajaran Model	26
3.11	Resolusi Gambar Data Inferensi	27
3.12	Spesifikasi Titik Kunci COCO-MS	28
3.13	Inferensi Pose Dua Dimensi OpenPose	28
3.14	Konversi Titik Kunci Dua Dimensi	29
3.15	Visualisasi Titik Kunci Dua Dimensi	29

3.16	Inferensi Model	30
3.17	Visualisasi Titik Kunci Tiga Dimensi	30
3.18	Visualisasi Aplikasi	31
4.1	Grafik Pemelajaran Model	32
4.2	Inferensi Pose Hilang	33
4.3	Kesalahan Deteksi	34

DAFTAR LAMPIRAN

Lampiran 1: Kelas Dataset	L1
Lampiran 2: Kelas ResLinear	L1
Lampiran 3: Arsitektur Model	L1
Lampiran 4: Options	L2
Lampiran 5: Pemuatan Data	L2
Lampiran 6: Instansiasi Dataset dan DataLoader	L2
Lampiran 7: Algoritma Pelatihan	L2
Lampiran 8: Algoritma Validasi	L2
Lampiran 9: Instansiasi Model	L2
Lampiran 10: Visualisasi Titik Kunci	L3
Lampiran 11: Visualisasi Inferensi	L3
Lampiran 12: Plot Grafik	L3

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pemanfaatan teknologi yang terkomputerisasi oleh manusia selalu meninggalkan jejak yang tersimpan dalam bentuk data digital. Rekam jejak ini merupakan bukti perilaku dan karakteristik manusia sehingga dijadikan sebagai acuan pengembangan teknologi dan ilmu pengetahuan pada masa mendatang. Data digital yang umumnya dimanfaatkan oleh manusia meliputi teks, citra audio, citra visual, dan citra audio visual yang disimpan ke dalam suatu media penyimpanan. Banyaknya jumlah data yang tersedia dan diprediksi akan semakin bertambah membuat gaya hidup manusia semakin bergantung pada teknologi digital.

Jejak digital bersifat laten yang berarti data digital memiliki makna khusus dan hanya dapat diolah dengan prosedur khusus. Citra visual terbentuk dari penangkapan pantulan gelombang elektromagnetik benda yang berada depan kamera dan kemudian disimpan dalam bentuk digital. Hasil akhir citra visual yang tercipta hanya memiliki informasi numerik mengenai warna, sedangkan informasi posisi benda saat penangkapan sudah hilang. Estimasi informasi laten dapat dilakukan komputer dengan membuat fungsi pemetaan dari suatu gambar terhadap suatu kondisi dengan memanfaatkan data dalam jumlah besar sebagai acuan.

Permodelan pembelajaran dalam atau *deep learning* dapat memetakan suatu domain ke domain lainnya secara mandiri menggunakan pembelajaran jaringan saraf tiruan dalam atau *deep neural network*. Pembelajaran dalam dapat dilakukan dengan komputasi mandiri yang sangat bergantung pada kuantitas dan kualitas data yang baik. Pembelajaran dalam menggunakan jaringan saraf tiruan dapat digunakan untuk mengembangkan teknologi khususnya di bidang visi komputer seperti melakukan estimasi pose tiga dimensi tubuh manusia yang terdapat dalam suatu gambar monokuler.

1.2 Rumusan Masalah

Permasalahan yang ingin diselesaikan adalah mengestimasi pose tubuh manusia yang bersifat laten dengan membuat sebuah pemetaan dari gambar yang ditangkap kamera monokuler ke koordinat setiap titik kunci pose menggunakan model *deep neural network*. Model yang telah terlatih mampu melakukan proses inferensi terhadap gambar baru sehingga dapat dijadikan sebagai acuan aplikasi yang mampu membaca pose tubuh manusia dari gambar monokuler.

1.3 Batasan Masalah

Penelitian ini menganggap setiap pose dua dimensi maupun pose tiga dimensi berada dalam koordinat lokal. Setiap pose ditransformasi ke dalam sistem koordinat kamera dengan titik kunci pinggang sebagai titik koordinat tengah. Hal ini dilakukan karena pemetaan hanya menggunakan gambar dua dimensi tanpa informasi kedalaman titik kunci sehingga dapat mengesampingkan masalah kedalaman yang ambigu.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat sebuah aplikasi yang dapat mengestimasi titik kunci dari sebuah citra visual datar dan melakukan transformasi ke pose lokal dua dimensi sehingga dapat dipetakan oleh jaringan saraf tiruan yang dimodelkan ke bentuk pose lokal tiga dimensi. Pose hasil juga divisualisasikan secara interaktif sehingga dapat digunakan untuk kepentingan yang sesuai.

1.5 Metode Penelitian

Penelitian dibagi menjadi tiga tahap besar terurut yang terdiri dari *data preprocessing*, pelatihan model jaringan saraf tiruan, dan visualisasi. Tahap pertama dan tahap ketiga tidak melibatkan pembelajaran mesin. Permasalahan utama dari penelitian ini berada pada tahap kedua tentang pelatihan jaringan saraf tiruan untuk melakukan pemetaan pose dua dimensi ke pose tiga dimensi.

Dataset yang digunakan dalam penelitian ini adalah *Human3.6M* yang berisi 3,6 juta pose unik yang dilakukan oleh sebelas aktor profesional dan direkam menggunakan empat sudut kamera yang berbeda beserta dengan koordinat setiap titik kunci dari hasil penangkapan alat *motion capture* [11].

Pose dua dimensi dan tiga dimensi merupakan variabel yang relevan untuk masalah. Pada tahap *data preprocessing* akan dilakukan ekstraksi variabel ini menjadi bentuk numerik sehingga mudah untuk digunakan saat melakukan pelatihan jaringan saraf tiruan. Tahap selanjutnya akan dilakukan pembuatan, permodelan, pelatihan, dan evaluasi jaringan saraf tiruan dalam untuk pemetaan titik kunci yang kemudian dilanjutkan dengan percobaan model dengan video. Tahap terakhir menampilkan visualisasi gambar, pose dua dimensi, dan pose tiga dimensi.

Perangkat keras yang digunakan dalam penelitian ini adalah satu unit laptop dengan spesifikasi:

- CPU Intel Core I7 7700HQ
- Memori 24 GB DDR4
- GPU NVIDIA GTX 1060 6GB
- SSD NVME SAMSUNG 120 GB
- HDD SATA 1 TB

Perangkat lunak yang digunakan dalam penelitian ini adalah satu unit laptop dengan spesifikasi:

- Python 3.7
- Jupyter Lab
- Git
- GitHub
- LaTeX

1.6 Sistematika Penulisan

Adapun sistematika penulisan yang digunakan dalam penulisan ini adalah sebagai berikut:

PENDAHULUAN, mengemukakan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, dan sistematika penulisan.

TINJAUAN PUSTAKA, menjelaskan kumpulan teori yang digunakan dalam mendukung proses penyelesaian program.

PENDEKATAN, mengemukakan langkah-langkah yang dicapai untuk membuat program.

HASIL DAN ANALISIS, mendalami hasil yang tercapai dengan analisis secara mendalam.

PENUTUP, mengulas lebih lanjut mengenai kesimpulan yang dapat ditarik dari hasil disertai dengan saran yang dapat menyempurnakan penelitian selanjutnya.

BAB II

TINJAUAN PUSTAKA

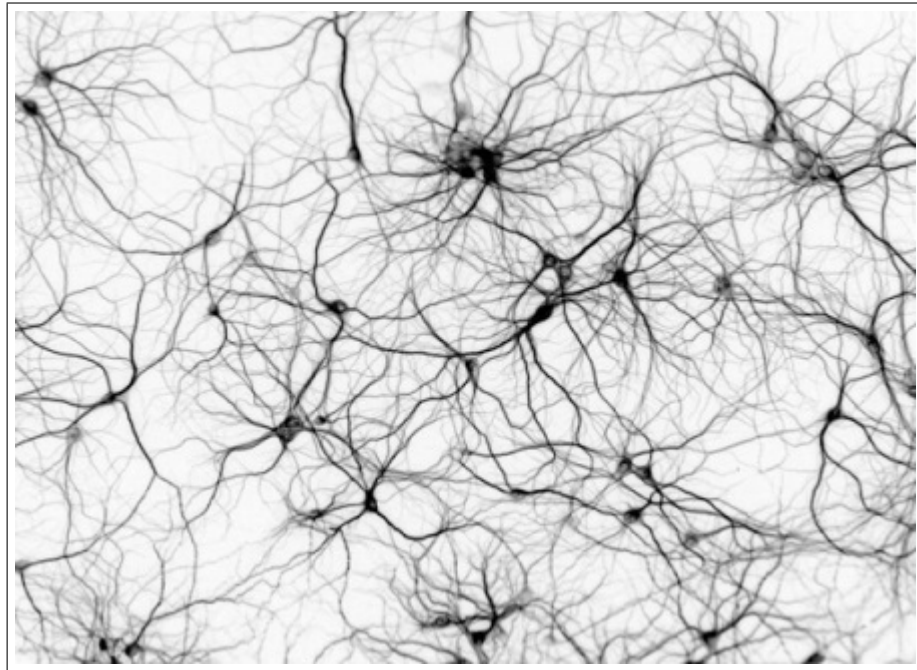
2.1 Teorema Penaksiran Universal

Teorema penaksiran universal atau *universal approximation theorem* menyatakan bahwa sebuah model jaringan *feed-forward* dapat membentuk fungsi apapun secara subjektif. Sebuah model jaringan saraf tiruan dibentuk dari serangkaian lapisan yang didalamnya terdapat deretan sel saraf atau *neuron* dengan kuantitas tertentu. Semakin panjang rangkaian lapisan yang tersedia, maka semakin banyak saraf yang tersedia sehingga dapat memetakan fungsi yang sulit. Model jaringan yang memiliki banyak saraf dapat mempelajari pola-pola yang ada dari satu domain ke domain lainnya [7].

Teorema penaksiran universal memiliki dua sifat yang dikategorikan berdasarkan pemanfaatannya dalam melakukan pembelajaran mesin. Sifat pertama adalah suatu model jaringan saraf tiruan dapat memperkirakan suatu fungsi dengan batasan-batasan tertentu sesuai dengan fungsi keluaran pada setiap lapisan yang terdapat dalam model khususnya lapisan yang terdapat pada bagian akhir. Sifat kedua adalah sebuah fungsi kontinu dengan jumlah variabel sembarang dapat ditiru sifatnya oleh sebuah jaringan saraf tiruan dengan jumlah parameter yang sembarang [13].

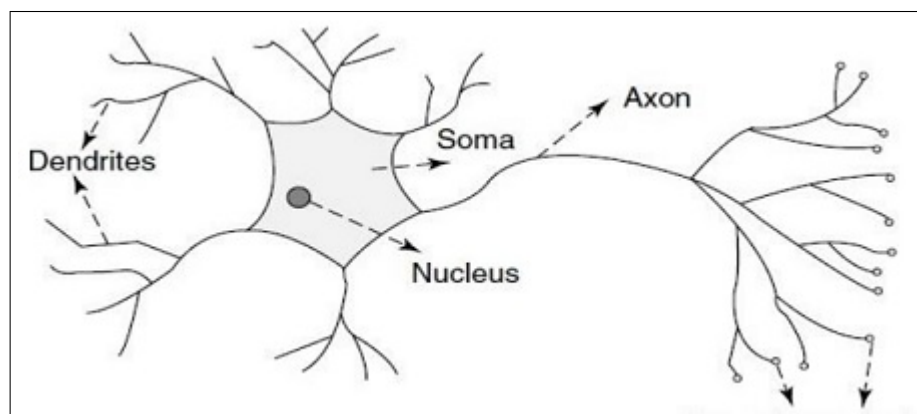
2.2 Jaringan Saraf

Otak manusia terdiri dari kumpulan sel saraf yang saling terkoneksi satu sama lain. Sebuah sel saraf adalah sel yang dapat memproses dan mengantarkan informasi apabila dirangsang dengan tegangan elektrokimia. Sel-sel saraf tidak membelah dirinya dan tidak digantikan apabila ada yang rusak. Jumlah sel saraf yang terdapat dalam otak manusia diperkirakan sebanyak satu miliar. Setiap sel saraf diperkirakan berkoneksi dengan sepuluh ribu sel saraf lainnya melalui sinapsis yang berarti otak manusia dewasa beroperasi seperti prosesor dengan kecepatan satu triliun bit per detik [9].



Gambar 2.1: Ilustrasi Jaringan Saraf Manusia

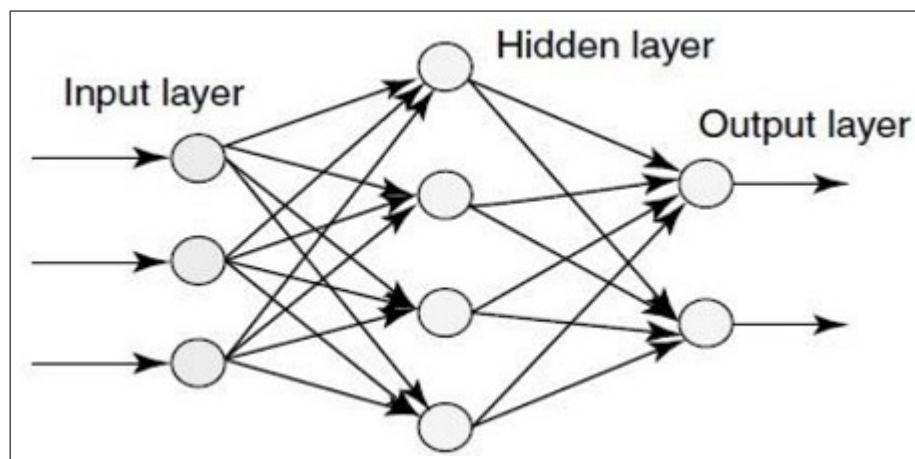
Bentuk sel saraf sangat bervariasi dengan berbagai ukuran, bentuk, dan sifat elektrokimianya. Sebuah sel saraf memiliki badan yang terdiri dari beberapa struktur penting meliputi *soma*, *dendrites*, *axon*, dan *synapses* seperti pada gambar 2.2. Sebuah sel saraf akan menerima beberapa masukan melalui *synapses*, memproses inputan tersebut melewati *dendrites*, kemudian diteruskan melalui *soma*, dan diberikan kepada sel saraf lainnya melalui *axon* [23].



Gambar 2.2: Ilustrasi Sel Saraf Manusia

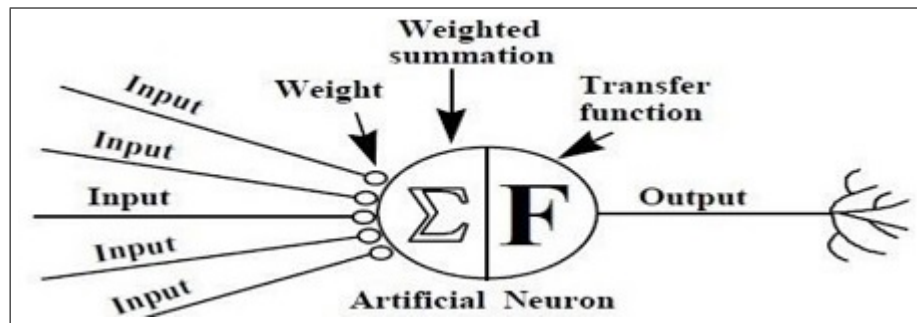
2.3 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan adalah sistem komputasi yang cara kerjanya menyerupai jaringan saraf pada otak makhluk hidup. Sebuah jaringan saraf tiruan dapat dengan mandiri memodelkan fungsi sembarang yang tingkat kesulitannya sesuai dengan jumlah koneksi yang tersedia. Jaringan ini menyerupai jaringan saraf asli dimana sebuah saraf tiruan menerima banyak masukan dari saraf lainnya kemudian dioperasikan dengan bobot yang terkandung pada sel tersebut dan akhirnya diteruskan ke sel berikutnya. Jaringan saraf umumnya terbentuk dari beberapa lapisan seperti pada gambar 2.3 [1].



Gambar 2.3: Ilustrasi Jaringan Saraf Tiruan

Sebuah sel saraf dapat dibagi menjadi empat bagian yang meliputi masukan, bobot, fungsi transfer atau aktivasi, dan keluaran. Jumlah masukan pada suatu sel saraf tiruan berjumlah sebanyak output dari sel-sel yang berada pada layer sebelumnya. Bobot sel adalah nilai numerik yang menjadi identitas dari sel yang merupakan hasil penyesuaian dari proses latihan. Fungsi aktivasi adalah sebuah fungsi menerima hasil operasi antara bobot dan masukan. Keluaran merupakan hasil dari sel yang diteruskan ke lapisan selanjutnya. Ilustrasi sebuah sel saraf tiruan dapat dilihat pada gambar 2.4 [20].



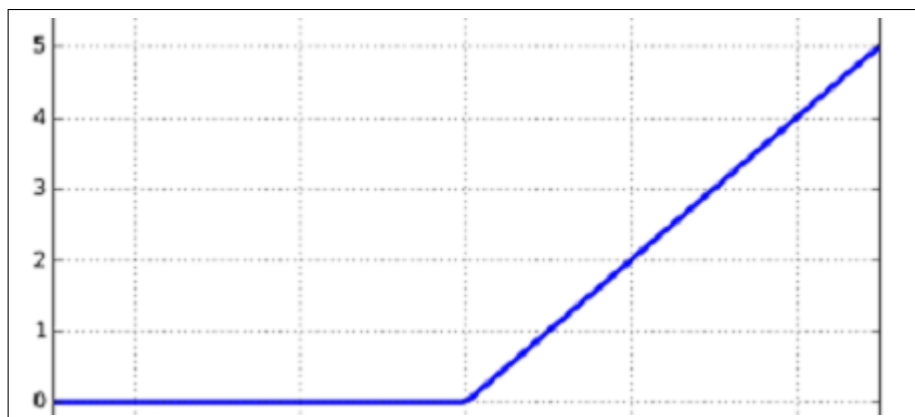
Gambar 2.4: Ilustrasi Sel Saraf Tiruan

2.4 Fungsi Aktivasi

Fungsi aktivasi pada sel saraf tiruan berfungsi untuk mengkonversi hasil operasi matriks antara masukan dan bobot sebuah sel dari sistem linier menjadi sistem nonlinier. Konversi nilai ini dilakukan agar setiap sel mengambil perannya pada saat proses pelatihan model. Beberapa fungsi aktivasi yang dipakai pada umumnya adalah *Sigmoid*, *ReLU*, *Tanh*, dan *Softmax* [2].

Rectified Linear Unit adalah fungsi aktivasi yang paling umum digunakan dalam aplikasi jaringan saraf tiruan. Fungsi aktivasi *ReLU* membatasi nilai masukannya dimana nilai yang kurang dari nol akan diubah menjadi nol [10, 16]. Persamaan fungsi aktivasi *Rectified Linear Unit* dapat dilihat seperti pada persamaan dibawah.

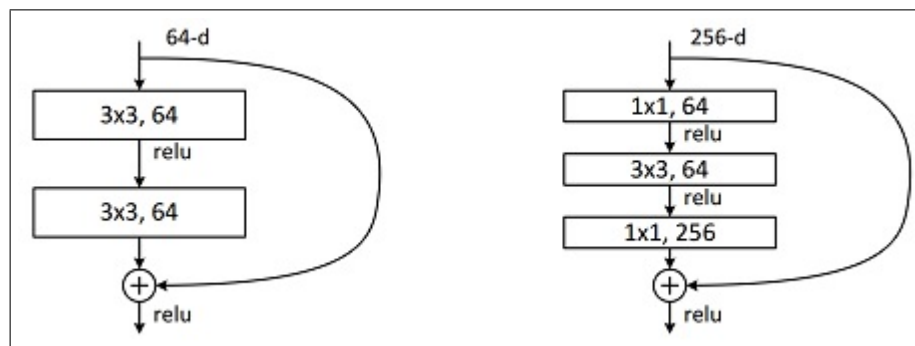
$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases}$$



Gambar 2.5: Rectified Linear Unit

2.5 Residual Network

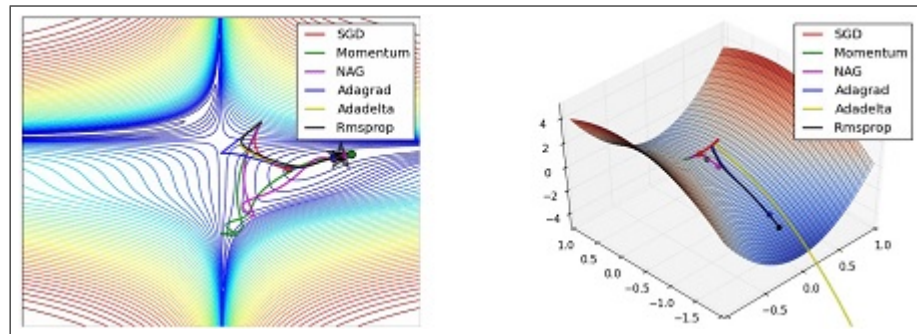
Residual Network atau *ResNet* merupakan model jaringan saraf tiruan yang menggunakan *residual block* sebagai dasar dari setiap lapisan. Sebuah *residual block* merupakan sebuah arsitektur jaringan saraf tiruan kecil yang terdiri dari beberapa lapisan. Setiap blok akan menjumlahkan masukan dan keluarannya sehingga layer di dalam suatu blok hanya menambahkan pola-pola yang dipelajari. Hal ini memungkinkan *ResNet* untuk memiliki jumlah blok yang sangat banyak sehingga dapat memetakan suatu fungsi sembarang yang sulit sesuai dengan teorema penaksiran universal [8]. Jenis-jenis *residual block* dapat dilihat pada gambar 2.6.



Gambar 2.6: *Residual Block*

2.6 Optimisasi Model

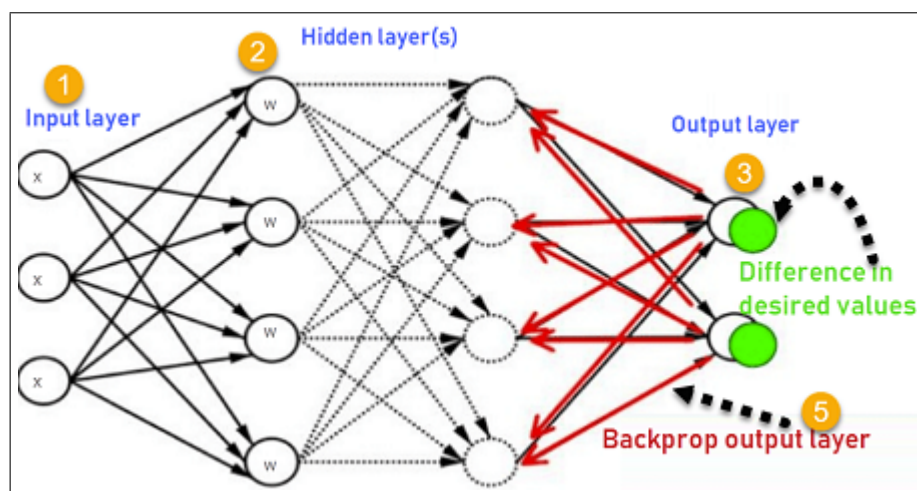
Algoritma optimisasi model yang paling populer dalam melakukan pembelajaran model *deep learning* adalah *gradient descent*. *Gradient descent* meminimalisir selisih antara prediksi model dan target sebenarnya dengan merubah bobot-bobot yang terdapat dalam model. Nilai bobot yang ditambahkan berbanding terbalik dengan gradien hasil fungsi kesalahan terhadap masing-masing bobot. Proses optimisasi dilakukan dengan melakukan *backpropagation* yang melibatkan beberapa elemen seperti *learning rate*, dan *loss function*. Tujuannya adalah untuk mencapai titik optimal pada sebuah bidang berdasarkan hasil dari *loss function* [7]. Ilustrasi titik optimal digambarkan pada gambar 2.7.



Gambar 2.7: Gradient Descent

2.6.1 Backpropagation

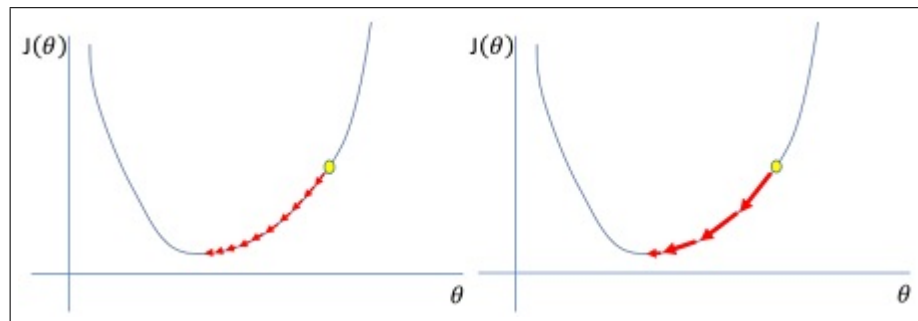
Backpropagation adalah sebuah prosedur pemelajaran jaringan saraf tiruan yang secara berulang-ulang kali menyesuaikan bobot setiap sel hingga selisih antara keluaran dan target menjadi lebih kecil. Algoritma ini merupakan mengorganisasikan sebuah model jaringan untuk secara mandiri mencari titik optimal secara berkala. Optimisasi dilakukan dengan melakukan *forward-pass* pada satu atau sebagian atau semua data yang ada ke dalam model untuk mendapatkan hasil prediksi. Hasil tersebut kemudian diukur selisihnya dengan target yang sebenarnya menggunakan sebuah fungsi kesalahan. Turunan parsial masing-masing bobot terhadap selisih kesalahan ini adalah kuantitas negatif yang harus ditambahkan bobot yang berkaitan [19]. Skema algoritma *backpropagation* dapat dilihat pada gambar 2.8.



Gambar 2.8: Skema Backpropagation

2.6.2 Learning Rate

Learning rate adalah sebuah nilai skalar yang menentukan seberapa besar sebuah bobot pada jaringan saraf akan ditambahkan. Nilai *learning rate* umumnya bernilai kecil karena besar turunan parsial pada setiap iterasi akan berubah-ubah. Nilai *learning rate* yang besar akan merubah bobot dengan skala yang besar sehingga akurasi model cenderung tidak stabil. Nilai *learning rate* yang kecil akan menghasilkan model yang stabil tetapi memerlukan jumlah iterasi yang banyak seperti pada gambar 2.9 [22].



Gambar 2.9: Ilustrasi Perbedaan *Learning Rate*

2.6.3 Mean Squared Error

Mean squared error adalah sebuah persamaan estimasi kesalahan dengan merata-ratakan hasil dari pangkat dua selisih antara dua vektor. Fungsi kesalahan ini dipakai untuk mengukur kesalahan model regresi dimana keluaran yang diukur bersifat kontinu. Persamaan ini akan selalu bernilai positif dan nilai yang mendekati nol menandakan bahwa selisih antara dua vektor semakin kecil seperti pada persamaan dibawah [21].

$$MSE = \frac{1}{n} \sum_{i=1}^n \left(\frac{d_i - f_i}{\sigma_i} \right)^2$$

2.7 Estimasi Pose Dua Dimensi

Estimasi pose dua dimensi melakukan lokalisasi titik kunci anatomi atau bagian tubuh manusia berdasarkan fungsi dari bagian tersebut pada sebuah gambar dua dimensi. Setiap gambar unik dapat berisi jumlah orang yang berbeda dan dapat

muncul dengan posisi dan ukuran yang berbeda-beda. Interaksi antara manusia dengan benda disekitarnya dapat menyebabkan oklusi dan hilangnya titik kunci anatomi yang dicari. Kompleksitas permasalahan ini juga semakin meningkat berbanding lurus dengan jumlah orang yang ada dalam suatu gambar [5] [6].

Pendekatan yang umum dilakukan adalah dengan mendeteksi setiap orang dalam setiap gambar secara individu. Hal pertama yang dideteksi adalah lokalisasi kemunculan suatu benda yang dianggap sebagai objek manusia dan kemudian mencari titik kunci anatomi dari objek tersebut. Pendekatan *top-down* seperti ini cocok dipakai untuk mendapatkan *single-person* dengan mudah tetapi kompleksitas yang proporsional dengan jumlah orang. Sebaliknya dengan pendekatan *bottom-up* dimana deteksi dilakukan dengan mencari alamat piksel pada gambar yang dianggap adalah titik kunci anatomi dan dilanjutkan dengan membangun pose manusia dari kandidat yang ada. Pendekatan *bottom-up* cocok untuk menyelesaikan masalah *multi-person* dikarenakan pencarian kandidat titik kunci dapat dilakukan terlebih dahulu tanpa harus mengetahui jumlah orang. Perbedaan pendekatan *top-down* dan *bottom-up* dapat dilihat pada gambar 2.10 [3].

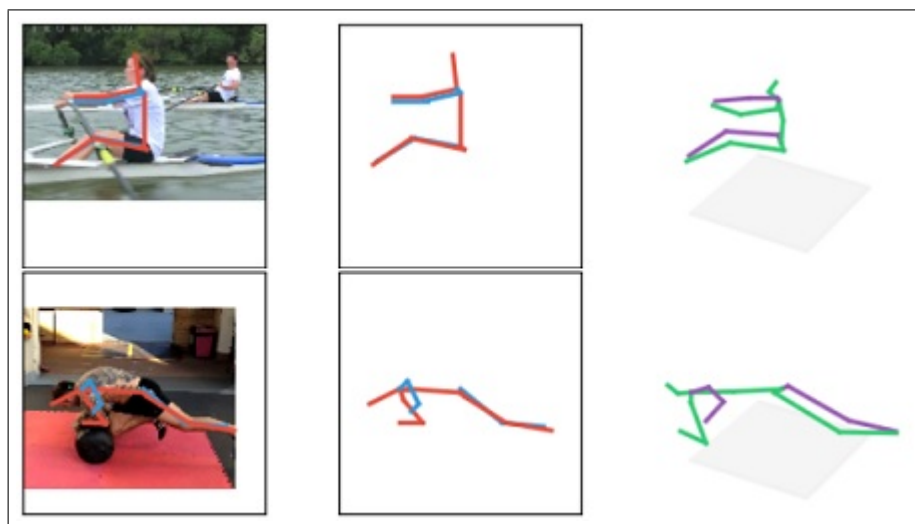


Gambar 2.10: Pendekatan *Top-Down* dan *Bottom-Up*

2.8 Estimasi Pose Tiga Dimensi

Algoritma estimasi pose tiga dimensi merekonstruksi titik kunci tiga dimensi tubuh manusia dari sebuah gambar dua dimensi. Proses pencarian titik kunci dibagi menjadi dua golongan yaitu pencarian secara langsung menggunakan gambar sebagai masukan dan pencarian bertahap dengan mencari titik kunci anatomi dua dimensi terlebih dahulu kemudian dilanjutkan dengan pencarian titik kunci tiga dimensi. Titik kunci yang didapatkan dapat berada dalam koordinat lokal dan global. Pencarian secara langsung tidak menghasilkan akurasi yang tinggi dan cenderung buruk dalam mendeteksi orientasi yang benar [4].

Titik kunci tiga dimensi yang berada dalam koordinat lokal menjadikan posisi pinggang sebagai titik nol yang berarti lokasi pinggang akan selalu bernilai $\vec{hip} = [0.0, 0.0, 0.0]$. Posisi titik kunci lain relatif terhadap titik pinggang. Hal ini menyebabkan pose yang ditangkap hanya bersifat lokal yang berarti hanya memiliki satu orientasi lokal. Pencarian pose tiga dimensi lokal dari gambar dua dimensi diilustrasikan pada gambar 2.11 [14].



Gambar 2.11: Pencarian Pose Tiga Dimensi Lokal

Titik kunci tiga dimensi dapat juga berada dalam koordinat global yang berarti orientasi kamera saat mengambil gambar diperhitungkan sebagai sebuah observasi virtual. Kompleksitas yang dihadapi berkaitan dengan orientasi kamera yang berubah. Estimasi pose tiga dimensi global umumnya dilakukan dengan

mendeteksi titik kunci anatomi tubuh manusia ke dalam suatu dunia virtual tiga dimensi sehingga dapat diamati dari berbagai sudut yang berkorelasi dengan kamera [24, 15]. Pencarian pose tiga dimensi global dari gambar dua dimensi diilustrasikan pada gambar 2.12.



Gambar 2.12: Pencarian Pose Tiga Dimensi Global

2.9 PyTorch

PyTorch merupakan *framework* untuk melakukan *deep learning* dengan bahasa pemrograman python. Penggiat data umumnya menggunakan bahasa pemrograman ini dalam melakukan riset pemanfaatan dan pengolahan data. PyTorch menyediakan implementasi grafik jaringan saraf tiruan yang bersifat dinamis. Hal ini memungkinkan pengguna untuk mengubah arsitektur model secara cepat dalam setiap iterasi [17].

2.10 Unified Modeling Language



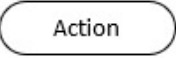

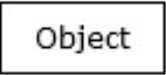
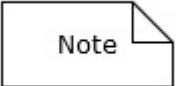
Unified Modeling Language (UML) merupakan bahasa pemodelan standar yang memiliki sintaks dan semantik. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. UML diaplikasikan dengan beberapa maksud seperti percangan perangkat lunak, sarana komunikasi antara perangkat lunak

dengan proses bisnis, penjabaran sistem secara rinci dan pendokumentasian sistem yang ada.

Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para pengembang aplikasi untuk bekerja sama dengan bahasa model yang sama dalam mengaplikasikan beragam sistem. UML merupakan Alat komunikasi yang konsisten dalam mendukung para pengembang saat ini. UML menyediakan sembilan jenis diagram yaitu : *Class Diagram*, *Packet Diagram*, *Use Case Diagram*, *Sequence Diagram*, *Communication Diagram*, *Statechart Diagram*, *Activity Diagram*, *Component Diagram* dan *Deployment Diagram*.

Activity Diagram merupakan penggabungan dari berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Tabel 2.1: Simbol-Simbol Activity Diagram

No	Nama	Gambar	Keterangan
1.		<i>Initial</i>	Bagaimana objek dibentuk atau diawali.
2.		<i>Final</i>	Bagaimana objek dihancurkan.
3.		<i>Action</i>	State eksekusi dari suatu aksi.
4.		<i>Fork</i>	Penggabungan atau pemisahan alur.
5.		<i>Object</i>	State hasil dari suatu aksi.
6.		<i>Note</i>	Keterangan tambahan suatu komponen.

BAB III

METODOLOGI PENELITIAN

3.1 Gambaran Umum

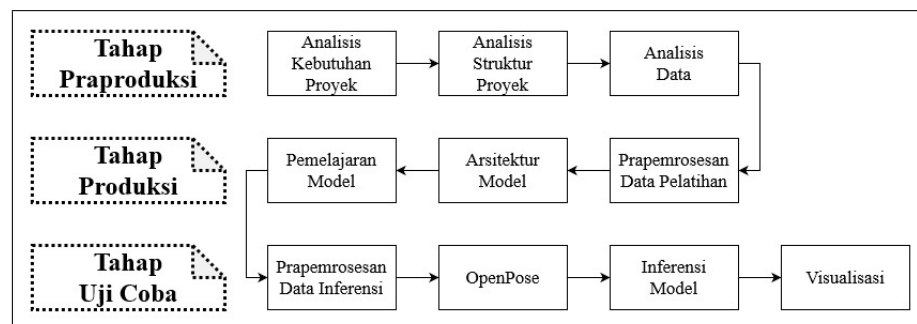
Penelitian ini membahas pemanfaatan data gambar sebagai acuan dalam melakukan pembelajaran dan implementasi model *deep neural network* untuk mencari dan memetakan koordinat tiga dimensi pose tubuh manusia dalam sebuah rangkaian gambar secara lokal. Pengerjaan aplikasi mengutamakan dua langkah penting yang meliputi pengolahan data dan pembuatan model. Aplikasi yang dibuat dapat menampilkan plot grafik tiga dimensi menyerupai struktur anatomi tubuh manusia sesuai dengan pose hasil estimasi dari gambar masukkan. Hasil pembelajaran model ditampilkan dalam grafik dua dimensi untuk analisis lebih lanjut.

Dataset yang digunakan dalam penelitian ini terbagi menjadi dua jenis yang meliputi *dataset* pembelajaran model dan *dataset* inferensi aplikasi. *Dataset* pembelajaran model dikategorikan menjadi data pelatihan model dan data validasi model. *Dataset* pembelajaran model berisi gambar dan target posisi titik kunci anatomi dalam jumlah besar. Data pelatihan model adalah data yang digunakan dalam proses pelatihan sebagai sampel bagi *deep neural network*. Data validasi model adalah data yang digunakan untuk menguji kebenaran fungsionalitas pemetaan yang dipelajari saat pelatihan model. *Dataset* inferensi aplikasi adalah data uji coba berbentuk video tanpa target titik kunci yang digunakan untuk estimasi pose tubuh manusia secara sekuensial.

Pemelajaran model *deep neural network* diimplementasikan menggunakan *framework* PyTorch. Kedua *dataset* yang digunakan diolah terlebih dahulu sehingga memenuhi syarat PyTorch dalam melakukan *deep learning*. Setiap model kemudian digunakan terhadap *dataset* inferensi aplikasi. Proses dan hasil estimasi diurai lebih lanjut dalam bentuk grafik visual.

3.2 Kerangka Penelitian

Kerangka penelitian yang jelas dibutuhkan untuk memudahkan proses penelitian sehingga dapat mempersingkat waktu pengerjaan. Proses penelitian dibagi menjadi tiga tahapan besar yang meliputi tahap praproduksi, tahap produksi, dan tahap uji coba. Setiap tahapan tersebut dikerjakan secara terurut dan sistematis. Alur setiap tahap diilustrasikan pada gambar 3.1.



Gambar 3.1: Kerangka Penelitian

3.3 Tahap Praproduksi

Tahap praproduksi berisi langkah-langkah analisis yang menentukan alur pada tahap selanjutnya. Tahap praproduksi dibagi menjadi beberapa langkah yang meliputi analisis kebutuhan proyek, analisis struktur proyek, dan analisis data. Tahap ini menganalisis bagian-bagian pokok yang diperlukan sehingga mengetahui langkah-langkah yang akan dilakukan pada tahap produksi.

3.3.1 Analisis Kebutuhan Proyek

Pemelajaran dan implementasi model ini memerlukan alat-alat pendukung berupa perangkat keras dan perangkat lunak yang mencukupi. Spesifikasi perangkat keras dan perangkat lunak yang lebih besar akan mempercepat proses pemelajaran model jaringan saraf tiruan. Spesifikasi perangkat keras yang digunakan dalam penelitian ini dapat dilihat pada tabel 3.1. Spesifikasi perangkat lunak yang digunakan dalam penelitian ini dapat dilihat pada tabel 3.2.

Tabel 3.1: Spesifikasi Perangkat Keras

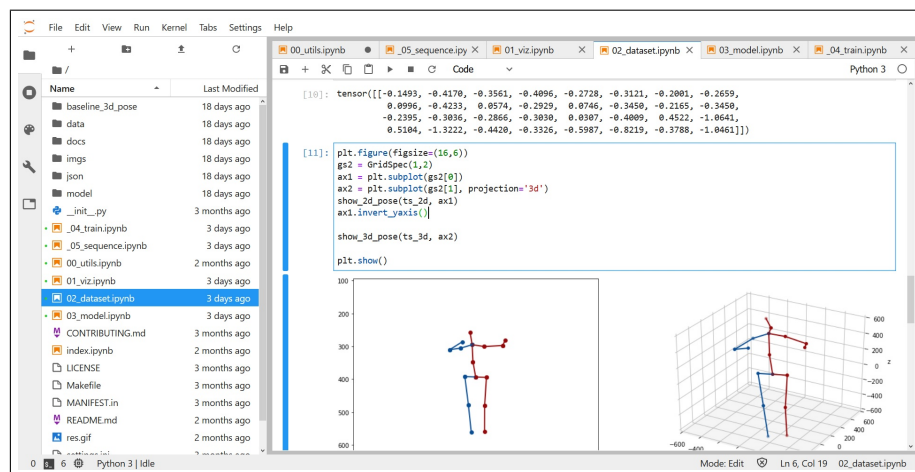
Perangkat Keras (Laptop)	
CPU	Intel Core I7 7700 HQ
GPU	NVIDIA GTX 1060 6 GB
RAM	24 GB DDR4
SSD	NVME SAMSUNG 120 GB
HDD	SATA 1 TB

Tabel 3.2: Spesifikasi Perangkat Lunak

Perangkat Lunak	
Sistem Operasi	Ubuntu 19.10
IDE	Jupyter Lab
Bahasa Pemrograman	Python 3.7
Framework	PyTorch 1.4

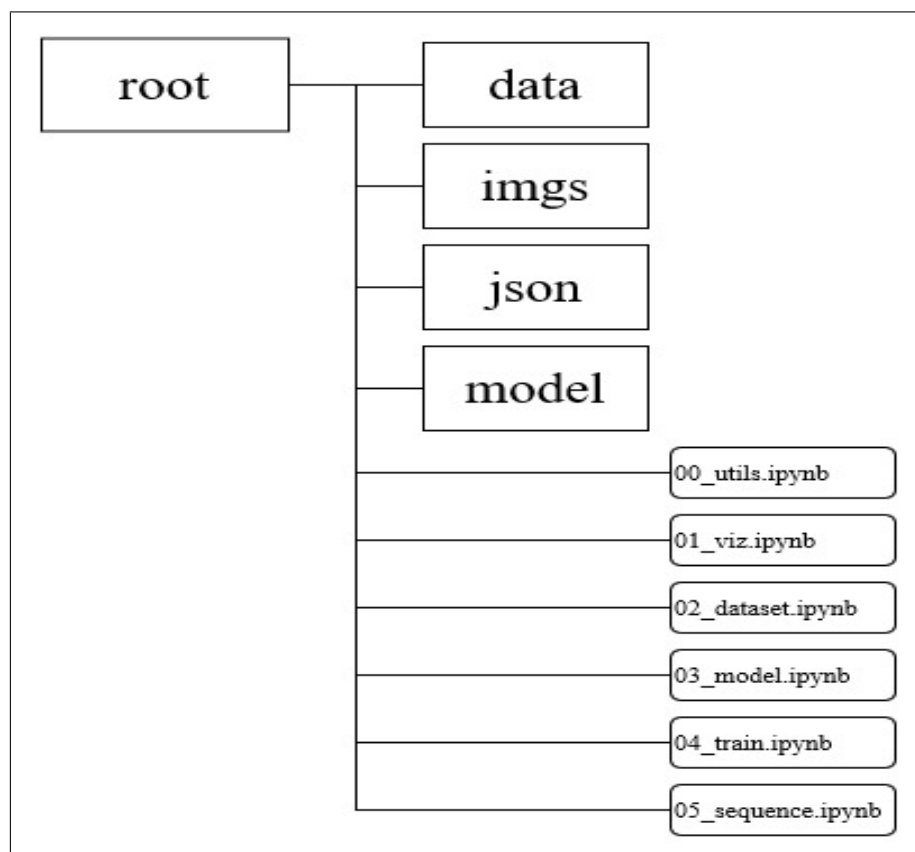
3.3.2 Analisis Struktur Proyek

Perancangan struktur proyek yang sistematis diperlukan untuk meminimalisir kompleksitas dalam melakukan pembuatan dan pembelajaran model. *Integrated development environment* Jupyter Lab memudahkan eksekusi perintah dengan sintaks bahasa pemrograman Python dalam bentuk sel interaktif dalam berkas berekstensi ipynb. Setiap sel terdiri dari *input* dan *output*. *Input* berisi perintah yang akan dieksekusi, sedangkan *output* berisi hasil eksekusi yang dapat berupa teks ataupun grafik. Tampilan Jupyter Lab dapat dilihat pada gambar 3.2.

**Gambar 3.2: Jupyter Lab**

Struktur direktori yang disusun terdiri dari empat folder dan enam berkas *interactive python notebook* berekstensi ipynb. Folder data berisi data latihan model seperti pada gambar 3.3. Folder imgs berisi rangkaian gambar yang diambil dari rekaman video. Folder json menampung berkas yang berisi informasi titik kunci dua dimensi saat inferensi. Folder model berisi *checkpoint* model selama pelatihan.

Berkas berekstensi ipynb berisi kode pemrograman yang dibagi menjadi enam modul. Berkas 00_utils merupakan modul utilitas yang memudahkan proses memuat data, menampilkan data, menyimpan data, dan memanipulasi data. Berkas 01_viz adalah modul percobaan menampilkan data pelatihan dan data inferensi. Berkas 03_model merupakan modul pendefinisian arsitektur model. Berkas 04_train adalah modul pelatihan model. Berkas 05_sequence adalah modul percobaan inferensi model terhadap rangkaian gambar.



Gambar 3.3: Struktur Direktori

3.3.3 Analisis Data

Data pembuatan model yang digunakan adalah data pemetaan dari pose dua dimensi ke pose tiga dimensi. Pose dua dimensi merupakan sampel, sedangkan pose tiga dimensi merupakan target. Sumber data adalah Human3.6M Dataset mengenai informasi perakaman gerakan pose manusia yang menyimpan gambar dari beberapa sisi beserta dengan pose dua dimensi dan tiga dimensinya. Informasi kedua pose tersebut kemudian dipisah dan disimpan dalam file berekstensi ".pt" [11]. Informasi mengenai data pembelajaran dapat dilihat pada tabel 3.3.

Tabel 3.3: Data Pembelajaran Model

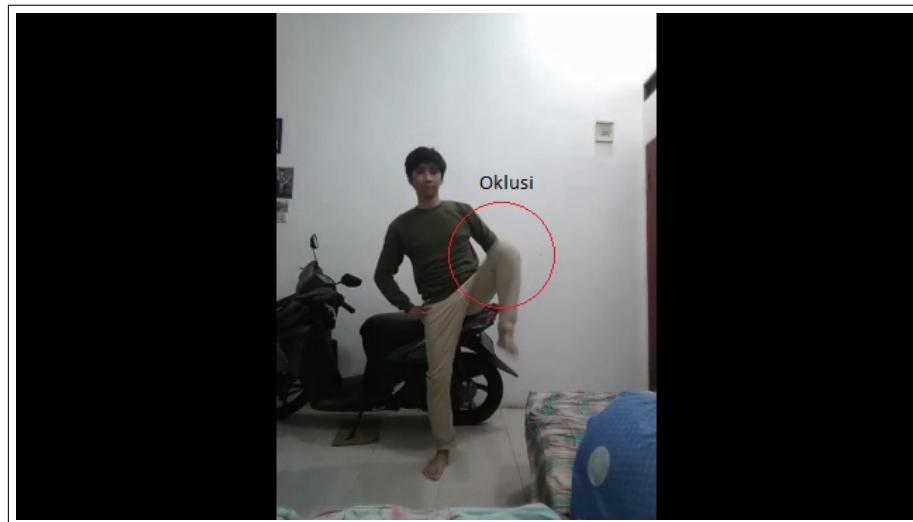
Nama Berkas	Isi
rcams.pt	matriks kamera <i>motion capture</i> terhadap kamera perekam
stat_2d.pt	<i>mean</i> , <i>standard-deviation</i> , dan <i>skeleton</i> pose dua dimensi
stat_3d.pt	<i>mean</i> , <i>standard-deviation</i> , dan <i>skeleton</i> pose tiga dimensi
test_2d.pt	data pose dua dimensi untuk validasi model
test_3d.pt	data pose tiga dimensi untuk validasi model
train_2d.pt	data pose dua dimensi untuk pelatihan model
train_3d.pt	data pose tiga dimensi untuk pelatihan model

Data inferensi model yang digunakan berupa video yang direkam menggunakan kamera diatas sebuah *tripod*. Hasil rekaman berupa sebuah video monokuler berisi seorang aktor yang memperagakan berbagai pose dasar. Pose-pose yang diperagakan mencakupi gerakan lengan, gerakan kaki, gerakan pinggang, gerakan kepala, dan gerakan memutar. Kompleksitas gerakan ini mengakibatkan terjadinya oklusi pada beberapa bagian badan yang berarti suatu anggota badan menutupi anggota badan lainnya.

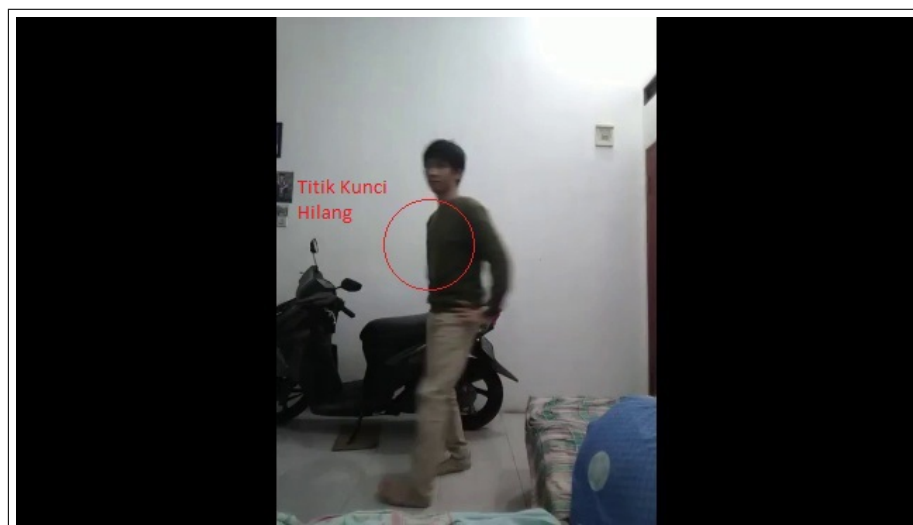
Kualitas perekaman video diturunkan dengan menggunakan pencahayaan yang satu arah. Efek *motion blur* diaktifkan sehingga gerakan yang cepat akan mengalami pengaburan. Kompleksitas gambar juga ditingkatkan dengan *aspect ratio* yang bernilai 19 : 6 dengan tipe rekaman *potrait* menyebabkan area rekaman hanya berada ditengah dan diapit oleh area piksel hitam. Beberapa *frame* dari data inferensi model dapat dilihat pada gambar 3.4, 3.5, dan 3.6.



Gambar 3.4: Frame 00077



Gambar 3.5: Frame 00173



Gambar 3.6: Frame 00232

3.4 Tahap Produksi

Tahap produksi merupakan tahap kedua yang dimana pengerjaan dilakukan. Tahap ini terdiri dari tiga langkah utama yaitu prapemrosesan data pelatihan, pembuatan arsitektur model, dan pembelajaran model.

3.4.1 Prapemrosesan Data Pelatihan

Dataset pembuatan model terbagi menjadi dua kategori yaitu data pelatihan (train_2d.pt dan train_3d.pt) dan data validasi (test_2d.pt dan test_3d.pt). Kedua data ini memiliki struktur dan bentuk yang sama sehingga prapemrosesan yang dilakukan juga sama. Setiap sampel pada data terdiri dari satu pose dua dimensi dan satu pose tiga dimensi. Perbedaan daripada kedua data ini terletak pada jumlah sampelnya. Data pelatihan berisi sebanyak 1.559.752 pasang sampel sedangkan data validasi memiliki 550.644 pasang sampel.

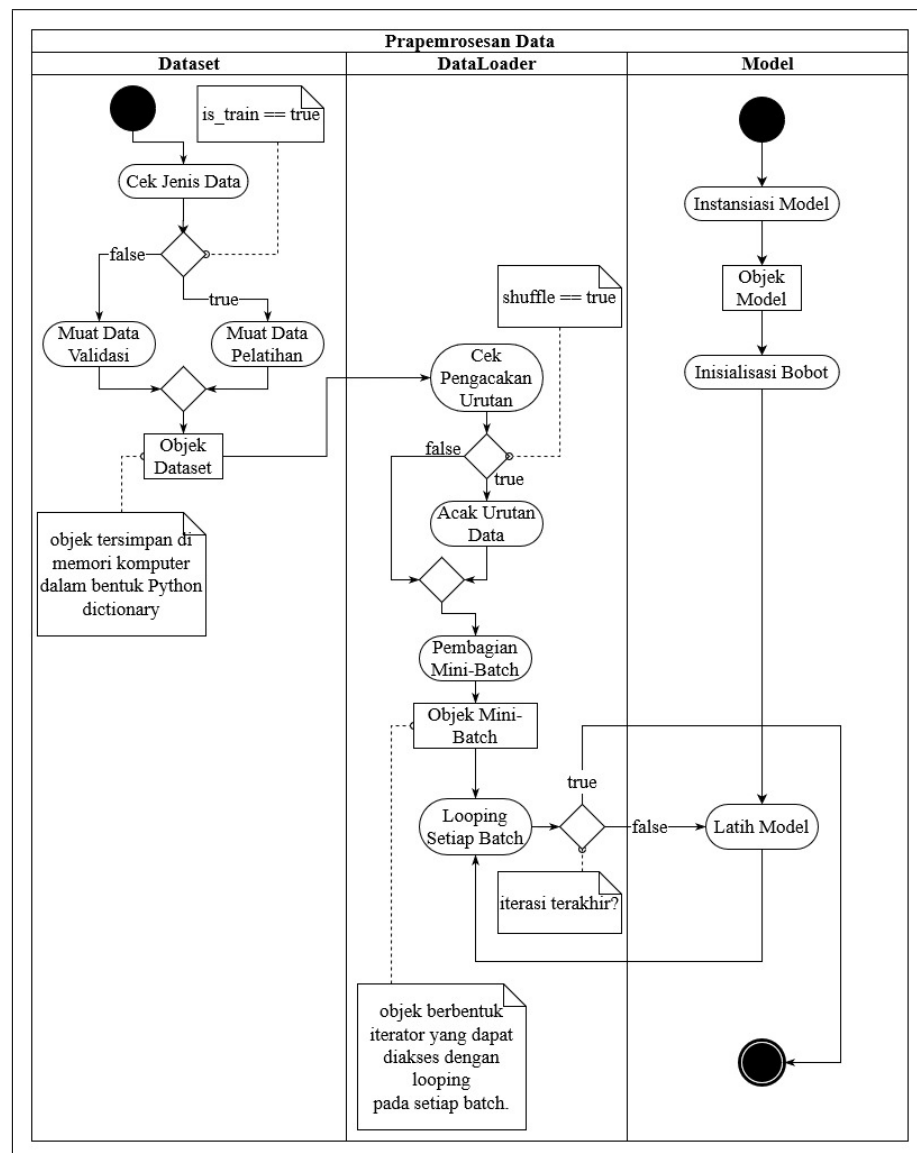
Berkas test_2d.pt, test_3d.pt, train_2d.pt, dan train_3d.pt merupakan berkas hasil konversi struktur data *dictionary* yang telah berbentuk *serialized*. Struktur *dictionary* yang berada didalam memori disimpan ke media SSD dalam bentuk *byte stream*. Berkas-berkas ini harus dibaca dengan mekanisme *deserializing* yaitu membangun ulang sebuah struktur data yang sama dengan membaca rangkaian *byte* sehingga dapat digunakan pada proses pelatihan model.

Mekanisme pemuatan data dilakukan secara *stochastic* dikarenakan hasil konversi data yang berukuran sangat besar. Serangkaian pasangan kunci dan target diproses dengan ukuran tertentu yang disebut dengan *mini-batch* sehingga VRAM pada GPU tidak mengalami *overflow*. Pemuatan data secara *stochastic* juga mempercepat proses pelatihan model karena lebih sedikit data yang harus dikalkulasi sebelum melakukan pemuktahiran.

Kelas *Dataset* dan *DataLoader* pada *framework* PyTorch memiliki fungsionalitas untuk melakukan pembacaan dan pembagian rangkaian data secara *stochastic*. Kelas *Dataset* dapat membaca berkas berekstensi ".pt" dari media SSD kemudian dimuat kedalam memori. Kelas *DataLoader* dapat memindahkan informasi didalam kelas *Dataset* ke VRAM pada GPU berbentuk *mini-batch*

sehingga dapat diproses secara *stochastic* dan paralel.

Prapemrosesan data pelatihan dan data validasi dilakukan dengan cara yang sama. Langkah pertama yang dilakukan adalah melakukan cek apakah data yang diinginkan adalah data pelatihan atau data validasi. Data tersebut kemudian dimuat dan disimpan kedalam memori. DataLoader menerima objek tersebut kemudian melakukan pengacakan data dan pembagian *mini-batch*. Hasil objek dapat diiterasi untuk melakukan pelatihan model. Skema prapemrosesan data dapat dilihat pada gambar 3.7.

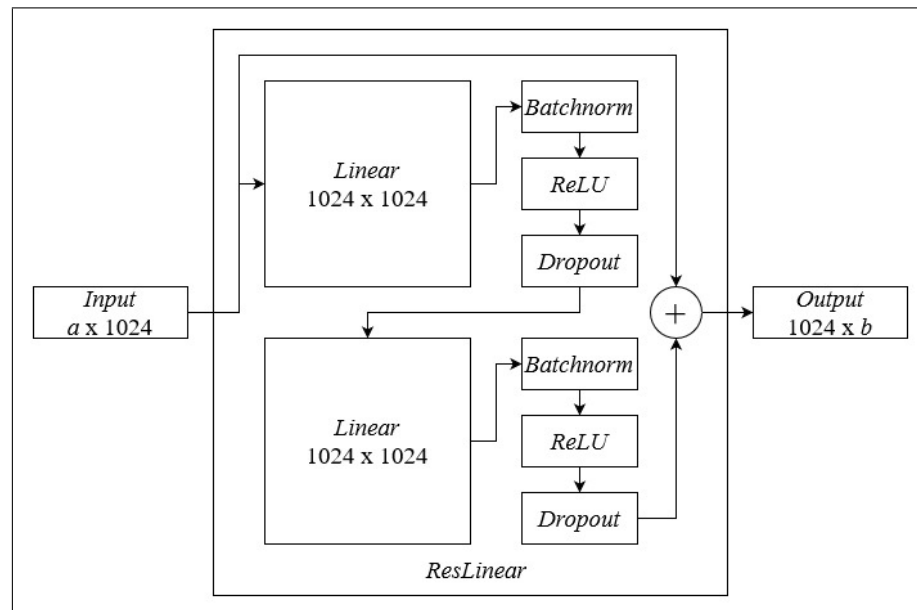


Gambar 3.7: Activity Diagram Prapemrosesan Data

3.4.2 Arsitektur Model

Arsitektur model jaringan saraf tiruan yang digunakan memiliki *input* berbentuk vektor dengan ukuran tiga puluh dua dan *output* berbentuk vektor dengan ukuran empat puluh delapan. Rangkaian lapisan yang menghubungkan *input* dan *output* berupa lapisan *residual network*. Bobot setiap lapisan diinisialisasi secara acak dengan distribusi normal.

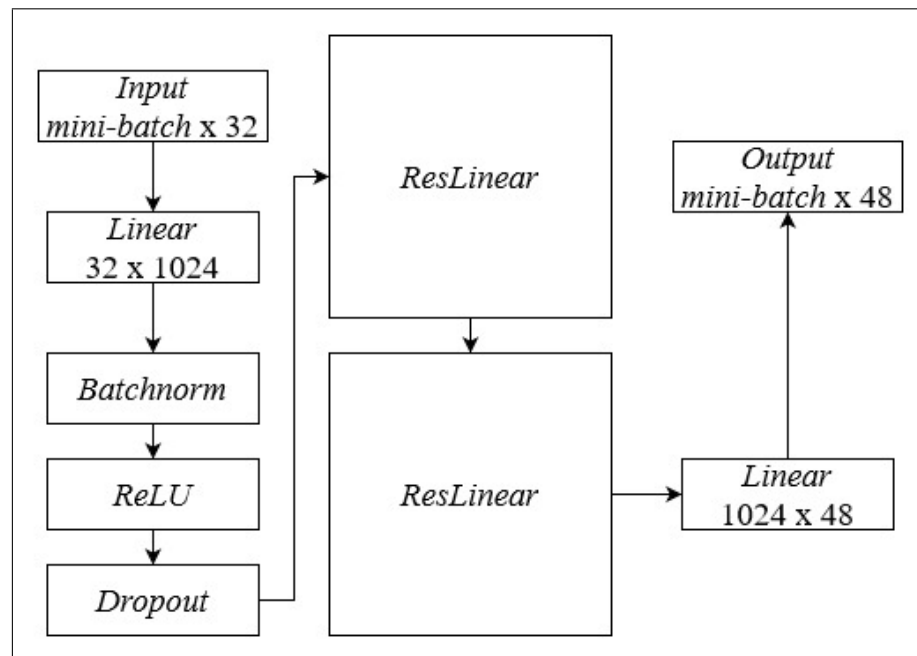
Sebuah lapisan *residual network* merupakan jaringan dengan arsitektur kecil dan sederhana yang dapat dipasang atau dibongkar secara modular yang disebut ResLinear. Lapisan ResLinear melakukan operasi penjumlahan antara *input* dan *output*. Sebuah ResLinear memiliki dua lapisan linier, dua lapisan *Batchnorm*, dua lapisan *Dropout*, dan dua lapisan *ReLU*. Komponen-komponen penyusun sebuah lapisan ResLinear dengan ukuran *input* a dan ukuran *output* b seperti pada gambar 3.8.



Gambar 3.8: Arsitektur Lapisan ResLinear

Arsitektur model secara keseluruhan terdiri dari tiga kelompok yang meliputi lapisan awal, lapisan *residual*, dan lapisan akhir. Lapisan awal menjembatani data *input* dan lapisan *residual* dengan menggunakan sebuah lapisan linier sebagai penyambung. Lapisan *residual* terdiri dari dua buah lapisan ResLinear yang berfungsi untuk memetakan pola data *input* terhadap data *output*.

Lapisan akhir menjembatani hasil pemetaan yang dilakukan oleh lapisan *residual* terhadap data *output*. Arsitektur model dapat dilihat pada gambar 3.9.

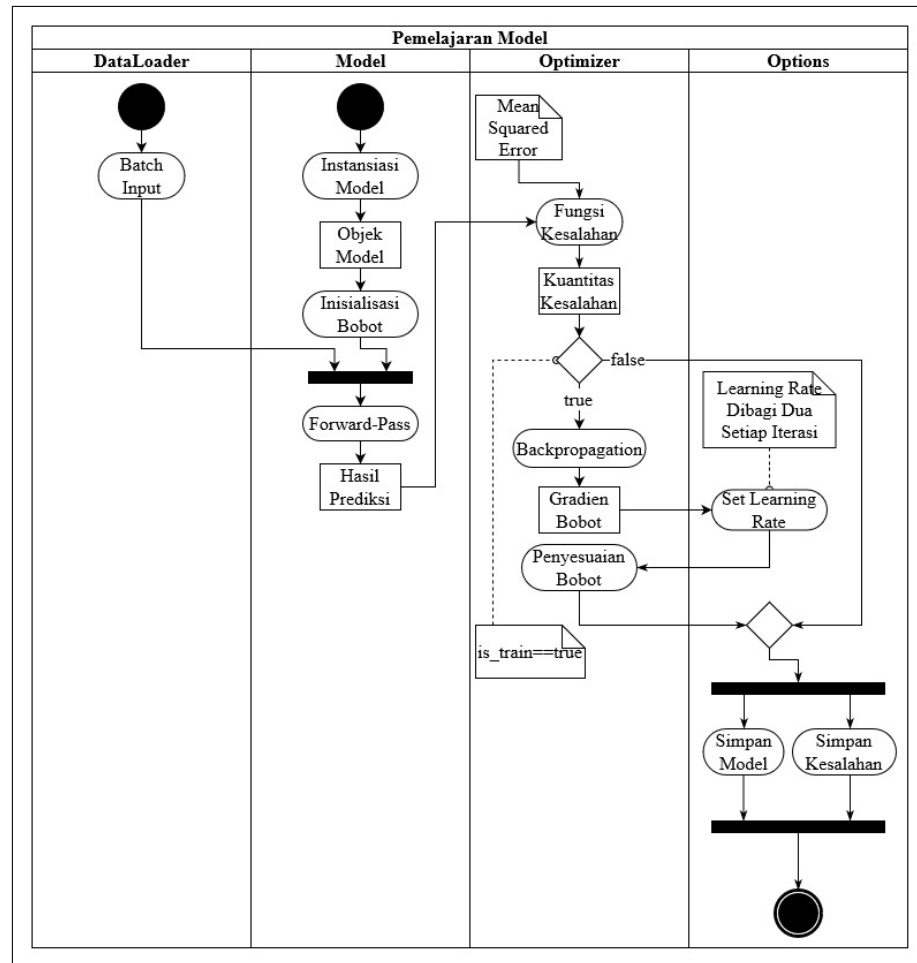


Gambar 3.9: Arsitektur Model

3.4.3 Pemelajaran Model

Pemelajaran model merupakan kelanjutan dari prapemrosesan data. Sebuah model baru diinstansiasii sehingga menghasilkan objek model. Objek model tersebut kemudian menginisialisasi bobot parameter dengan bilangan acak dari distribusi normal. *Batch input* yang berasal dari interaksi *DataLoader* diproses oleh model dengan metode *feed-forward*. Hasil prediksi sementara dari model didapatkan yang kemudian dibandingkan tingkat kebenarannya menggunakan fungsi kesalahan. Fungsi kesalahan *mean squared error* menghasilkan angka kuantitas kesalahan. Angka ini merupakan tolak ukur seberapa akurat kemampuan model dalam menghasilkan output yang relevan. Apabila model tidak berada dalam status "is_train", maka kuantitas kesalahan yang didapatkan langsung disimpan dalam bentuk *array* untuk keperluan analisis. Apabila model berada dalam status "is_train", maka model melakukan *backpropagation* untuk menghasilkan gradien bobot. *Learning rate* kemudian dibagi dengan dua sehingga

menjadi lebih kecil. Penyesuaian bobot dilakukan dengan menjumlahkan bobot dengan hasil operasi perkalian antara *learning rate* dan gradien bobot. Bobot model yang telah diperbarui disimpan berserta dengan kuantitas kesalahannya. Algoritma yang sama akan diulangi pada setiap *batch input*. Skema pembelajaran model dapat dilihat pada gambar 3.10.



Gambar 3.10: Activity Diagram Pemelajaran Model

3.5 Tahap Uji Coba

Tahap uji coba berisi langkah-langkah uji coba penggunaan model. Tahap uji coba dibagi menjadi beberapa langkah yang meliputi prapemrosesan data inferensi, penggunaan OpenPose, inferensi model, dan visualisasi. Tahap ini bertujuan untuk menggunakan model sebagai aplikasi untuk mengestimasi pose tiga dimensi dari gambar monokuler.

3.5.1 Prapemrosesan Data Inferensi

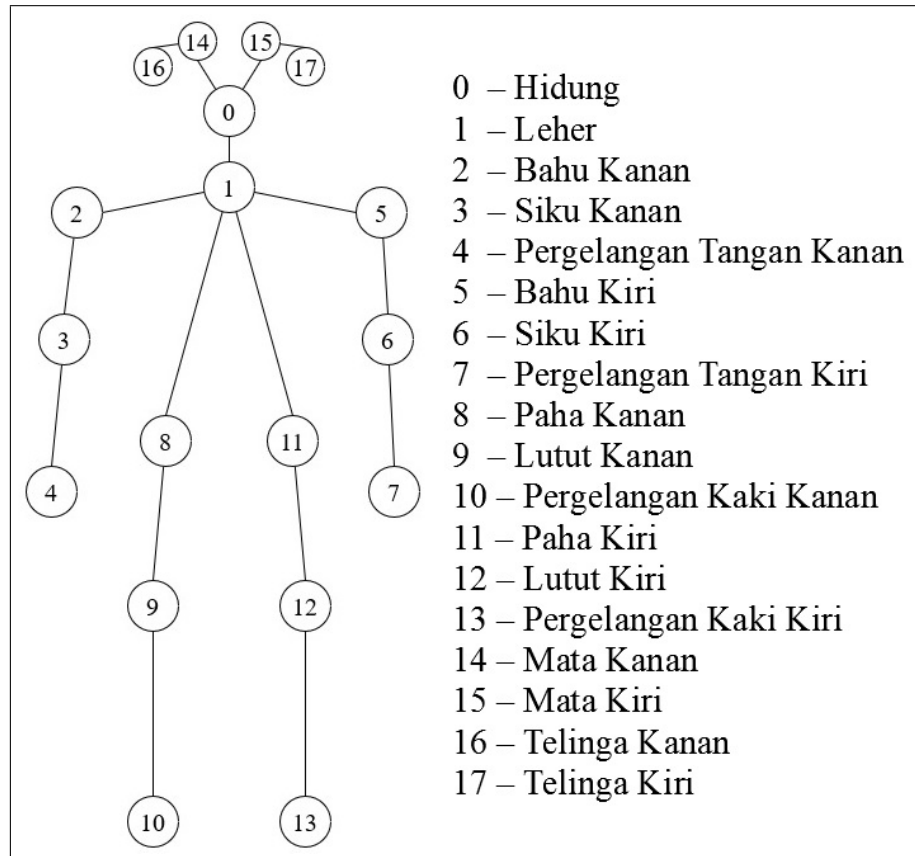
Proses inferensi pose dua dimensi menjadi lebih cepat ketika gambar dua dimensi yang dioperasikan memiliki resolusi yang kecil. Tingkat resolusi yang digunakan saat merekam pose adalah 640 x 360. Memperkecil ukuran resolusi juga harus dilihat dari segi kualitas gambar yang dihasilkan. Gambar juga memiliki banyak informasi statis seperti pada area piksel berwarna hitam yang berada di kiri dan kanan gambar. Resolusi 290 x 290 dengan titik tengah berada pada titik kunci pinggang dianggap tepat karena mampu menjangkau semua pose badan dan kualitas gambar yang masih bagus. Inferensi yang lebih efisien dapat dilakukan dengan memperkecil resolusi dan area piksel mati seperti pada gambar 3.11.



Gambar 3.11: Resolusi Gambar Data Inferensi

3.5.2 OpenPose

OpenPose merupakan aplikasi estimasi pose tubuh manusia dua dimensi. OpenPose menerima input gambar kemudian mencari titik kunci pose dua dimensi. Titik kunci pose dua dimensi berada pada koordinat lokal sesuai dengan bidang gambar. OpenPose menghasilkan berkas "json" yang berisi hierarki pose menurut spesifikasi COCO-MS. Anotasi COCO-MS berisi delapan belas titik kunci tubuh manusia dengan urutan tertentu seperti pada gambar 3.12. Visualisasi estimasi pose dua dimensi menggunakan OpenPose dapat dilihat pada gambar 3.13.



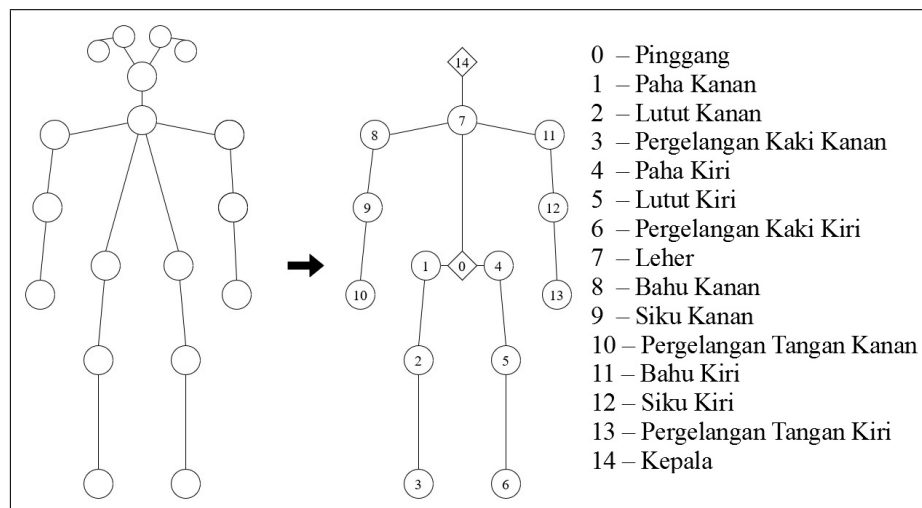
Gambar 3.12: Spesifikasi Titik Kunci COCO-MS



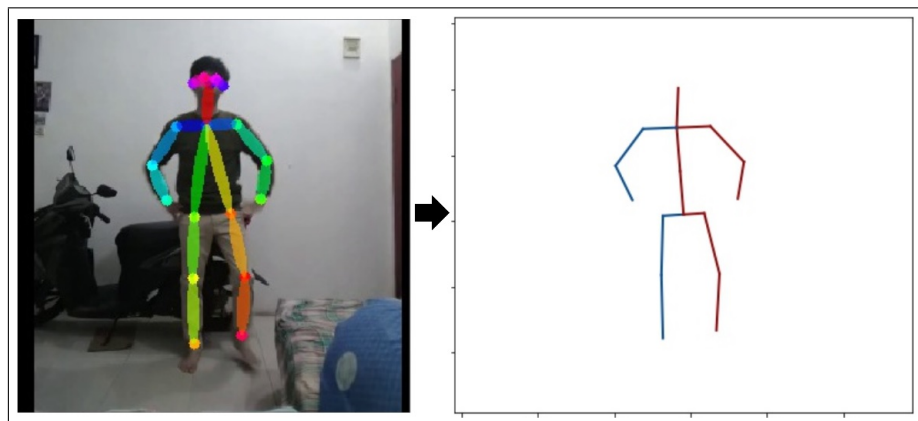
Gambar 3.13: Inferensi Pose Dua Dimensi OpenPose

3.5.3 Inferensi Model

Rangkaian titik kunci yang dihasilkan oleh OpenPose memiliki spesifikasi COCO-MS yang berbeda dengan spesifikasi *dataset* pembuatan model. Spesifikasi COCO-MS yang memiliki delapan belas titik kunci dikonversi menjadi lima belas titik kunci dengan menyatukan titik kunci kedua mata dan kedua telinga serta membuat titik kunci pinggang berdasarkan rata-rata kedua kaki bagian atas seperti pada gambar 3.14. Konversi ini dilakukan supaya model dapat melakukan inferensi terhadap titik kunci yang mewakili pose tersebut. Visualisasi konversi titik kunci COCO-MS menggunakan warna biru mewakili anggota badan sebelah kanan bersamaan dengan warna merah yang mewakili anggota badan tengah dan kiri. seperti pada gambar 3.15.

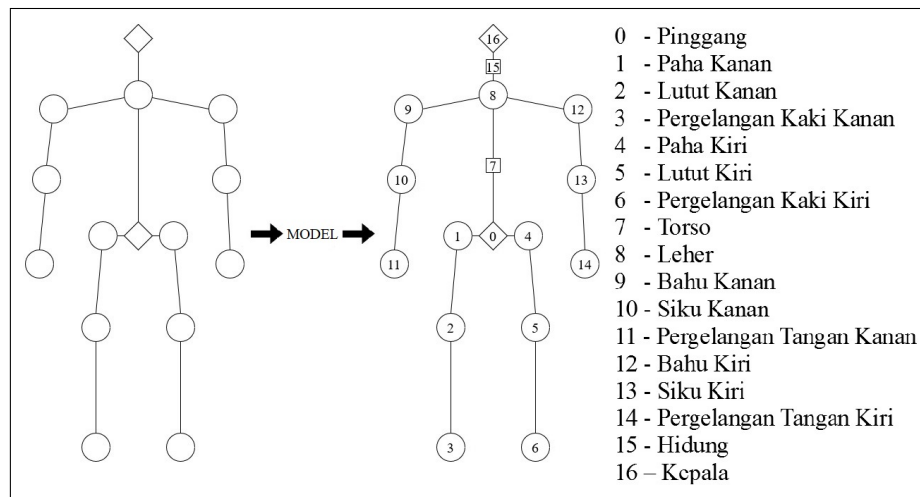


Gambar 3.14: Konversi Titik Kunci Dua Dimensi

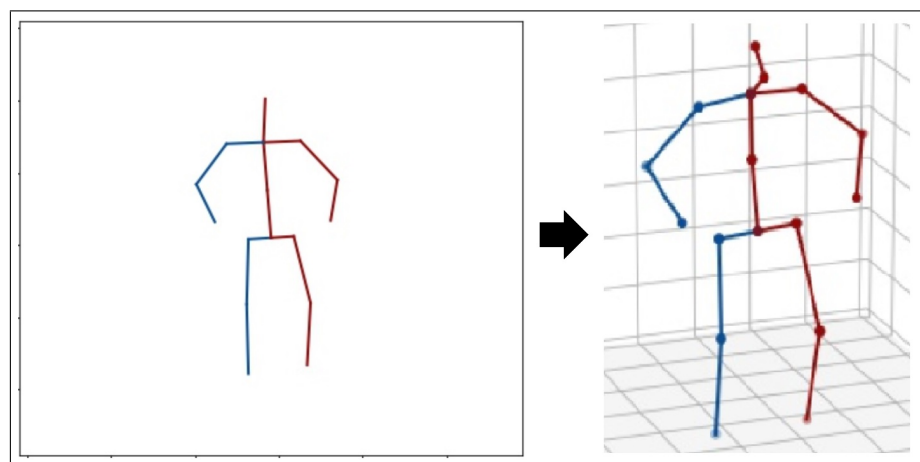


Gambar 3.15: Visualisasi Titik Kunci Dua Dimensi

Inferensi pada model *deep neural network* menerima titik kunci pose dua dimensi yang telah dikonversi sebagai *input* kemudian menghasilkan titik kunci pose tiga dimensi sebagai *output*. Inferensi model menghasilkan tujuh belas titik yang berada dalam bidang tiga dimensi. Titik kunci pose tiga dimensi memiliki dua titik baru yang meliputi torso dan hidung. Titik torso mewakili lengkungan badan pada pose tiga dimensi sehingga pose terlihat akurat. Titik hidung mewakili arah hadapan kepala. Kedua titik tambahan ini memperjelas orientasi pose tiga dimensi. Proses inferensi model untuk mendapatkan pose tiga dimensi dapat dilihat pada gambar 3.16. Visualisasi inferensi model dapat dilihat pada gambar 3.17.



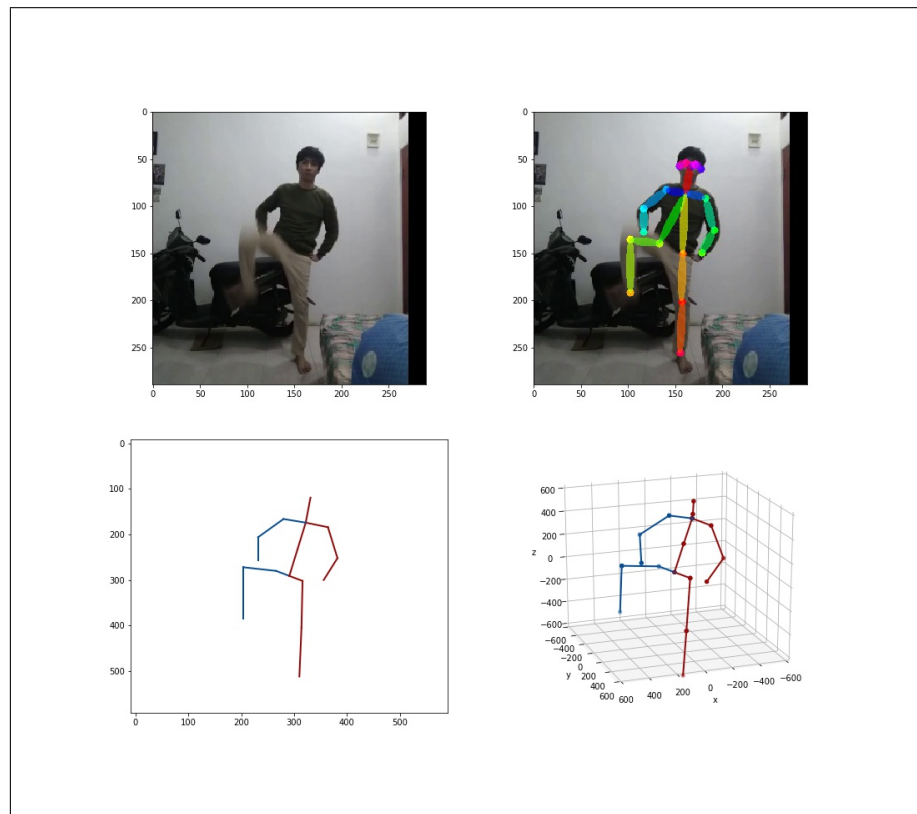
Gambar 3.16: Inferensi Model



Gambar 3.17: Visualisasi Titik Kunci Tiga Dimensi

3.5.4 Visualisasi

Visualisasi mencakup penggunaan aplikasi dalam satu bingkai secara keseluruhan. Terdapat empat buah figur yang masing-masing mewakili langkah-langkah pada tahapan uji coba. Figur pertama (kiri atas) berisi prapemrosesan data inferensi dimana resolusi gambar diubah menjadi 290 x 290. Figur kedua (kanan atas) menggambarkan pose dua dimensi yang didapatkan oleh OpenPose. Figur ketiga (kiri bawah) menggambarkan konversi titik kunci OpenPose dengan spesifikasi COCO-MS menjadi titik kunci yang cocok dengan model. Figur keempat (kanan bawah) menggambarkan pose tiga dimensi yang dihasilkan oleh model kedalam sebuah sistem koordinat tiga dimensi. Contoh visualisasi aplikasi dapat dilihat pada gambar 3.18.



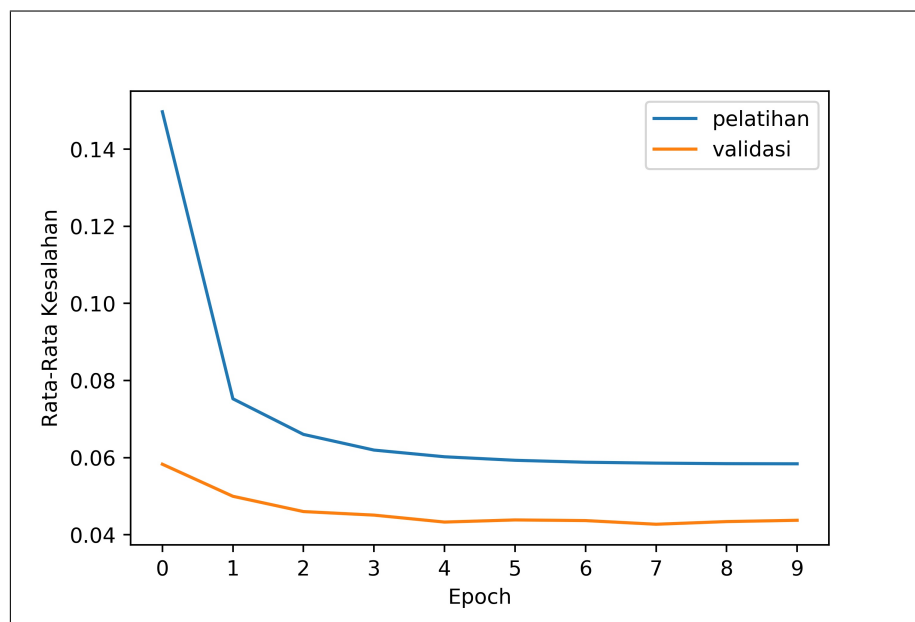
Gambar 3.18: Visualisasi Aplikasi

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Pemelajaran Model

Pemelajaran model yang dilakukan selama sepuluh *epochs* menunjukkan bahwa kesalahan model dalam mengestimasi pose tiga dimensi berkurang dalam setiap *epoch*. *Learning rate* yang dibagi dua dalam setiap *epoch* mempengaruhi pemelajaran model dimana penyesuaian model semakin teliti. Adaptasi bobot model terjadi secara drastis pada *epoch* 0 sampai dengan *epoch* 3. *Epoch* 4 dan seterusnya menggunakan *learning rate* yang semakin kecil sehingga model semakin teliti dalam melakukan adaptasi. Grafik pemelajaran model dapat dilihat pada gambar 4.1.



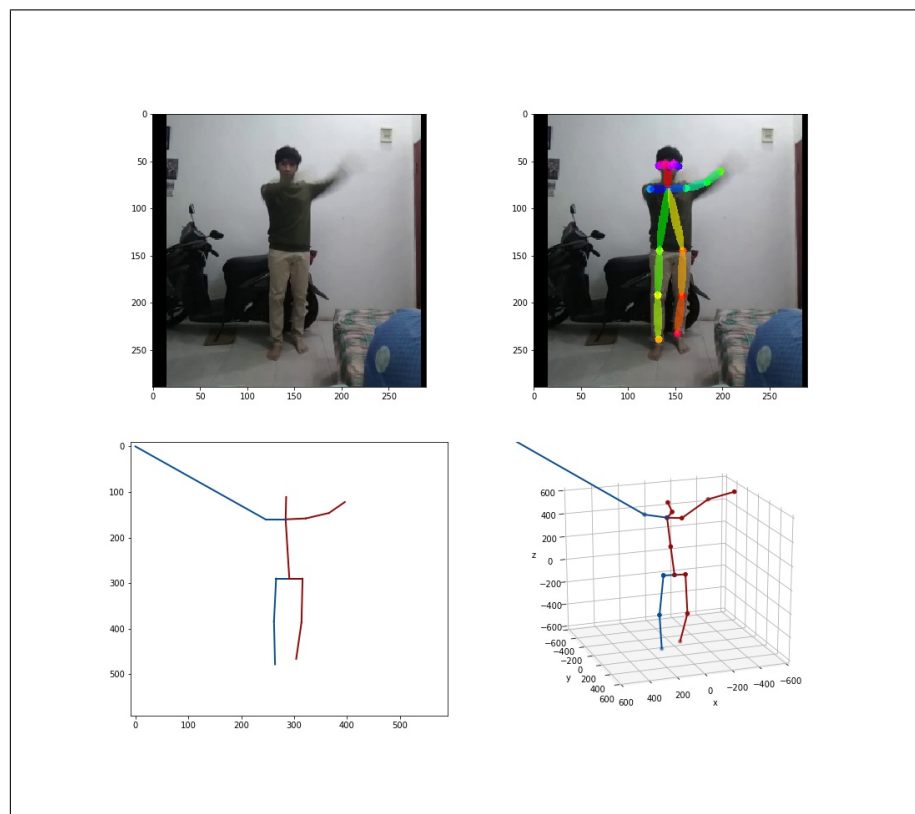
Gambar 4.1: Grafik Pemelajaran Model

4.2 Kekurangan Aplikasi

Inferensi yang bagus akan terjadi apabila langkah-langkah pada tahapan uji coba tidak mengalami kesalahan. Kualitas gambar dan pose yang tidak cacat juga mempengaruhi proses dari awal hingga akhir. Prapemrosesan gambar pada data inferensi yang tepat memudahkan OpenPose dalam mencari titik kunci pose dua

dimensi secara lengkap. Titik kunci OpenPose yang lengkap kemudian memenuhi syarat untuk dikonversi menjadi spesifikasi yang diinginkan. Informasi tersebut kemudian diteruskan ke model untuk mendapatkan titik kunci pose tiga dimensi.

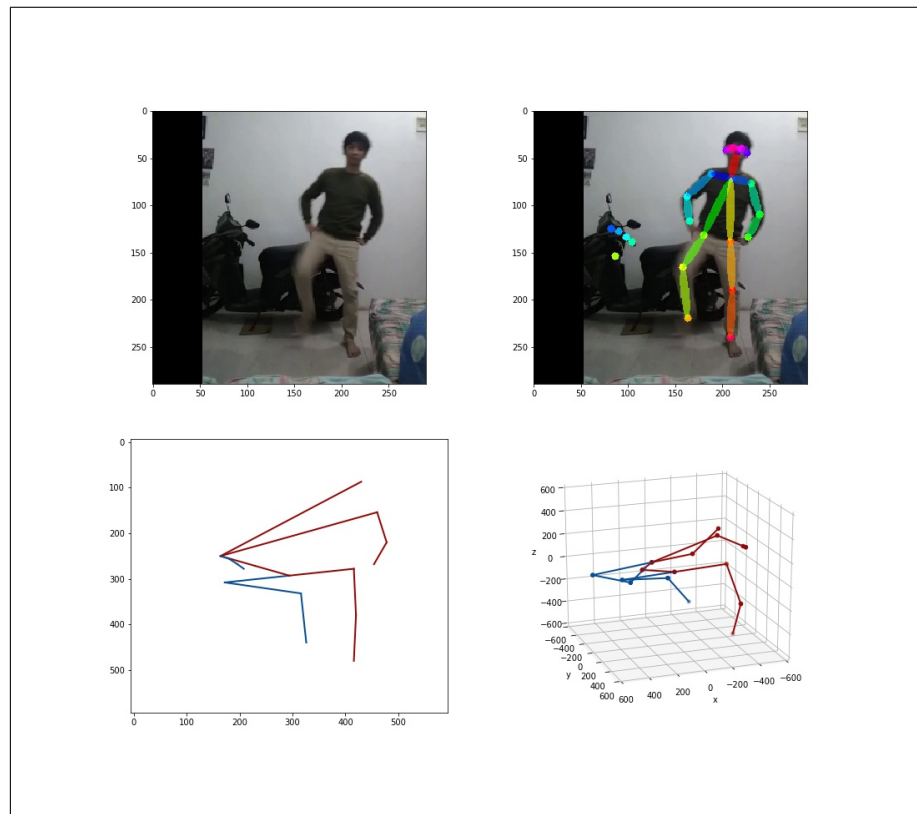
Kualitas pose yang cacat menghasilkan estimasi pose tiga dimensi yang cacat. Oklusi pose pada gambar dua dimensi dapat menghilangkan suatu bagian tubuh. Hilangnya bagian ini dari gambar menyebabkan kesalahan pada langkah-langkah selanjutnya. Titik kunci lengan kanan hilang ketika pose lengan mengarah lurus ke lensa kamera sehingga terjadi oklusi. Hal ini menyebabkan OpenPose tidak dapat menemukan titik kunci lengan kanan dan memberi nilai nol pada titik kunci tersebut. Proses konversi dan inferensi titik kunci tiga dimensi juga menghasilkan pose yang tidak realistis. Inferensi pose hilang yang diakibatkan oklusi dapat dilihat pada gambar 4.2.



Gambar 4.2: Inferensi Pose Hilang

Kesalahan juga dapat terjadi pada proses inferensi titik kunci. Apabila OpenPose mengeluarkan *output* yang ambigu dimana terdapat titik kunci yang

dianggap sebagai bagian dari tubuh manusia. OpenPose menghasilkan titik kunci ganda yang tidak sesuai dengan spesifikasi yang diperlukan meskipun berhasil mendeteksi pose secara lengkap. Hasil keluaran yang tidak sesuai dengan spesifikasi model mengakibatkan estimasi pose tiga dimensi yang rusak. Inferensi pose yang mengalami kesalahan deteksi dapat dilihat pada gambar 4.3.



Gambar 4.3: Kesalahan Deteksi

BAB V

PENUTUP

5.1 Kesimpulan

Aplikasi estimasi pose tiga dimensi menggunakan model *deep neural network* berhasil dilatih. Model ini melakukan pembelajaran secara mandiri menggunakan informasi pose dua dimensi sebagai *input* dan pose tiga dimensi sebagai *output*. Model tersebut diuji untuk menemukan pose tiga dimensi pada data inferensi. Model melakukan pembelajaran selama sepuluh *epochs* dengan hasil rata-rata kesalahan akhir bernilai 0.0584 pada data pelatihan dan 0.0437 pada data validasi. Nilai kesalahan data pelatihan masih lebih besar daripada nilai data validasi. Hal ini menandakan model masih berada pada kondisi *underfitting* dimana selisih kedua nilai tersebut relatif besar. Model yang lebih baik dapat didapatkan dengan melatih model dalam jumlah *epochs* yang lebih banyak dan berhenti saat mulai terjadi *overfitting*.

5.2 Saran

Pengembangan model *deep neural network* ini masih menggunakan arsitektur minimalis, data dengan satu domain, dan memiliki tahapan yang tidak efisien. Pengembangan selanjutnya disarankan menggunakan arsitektur yang lebih efisien. Arsitektur *residual network* merupakan arsitektur yang paling bagus pada saat penulisan ini dilakukan. Algoritma pembelajaran yang lebih efisien juga disarankan pada penelitian mendatang. Data dengan domain yang lebih luas juga merupakan hal yang penting seperti estimasi pose pada hewan tertentu. Dengan demikian penelitian ini dapat bermanfaat dan dapat dikembangkan menjadi jauh lebih baik lagi pada masa mendatang.

DAFTAR PUSTAKA

- [1] Abiodun, O., Jantan, A., Omolara, O., Dada, K., Umar, A., Linus, O., Arshad, H., Aminu Kazaure, A., Gana, U., dan Kiru, M. (2019). Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, PP:1–1.
- [2] Agostinelli, F., Hoffman, M., Sadowski, P., dan Baldi, P. (2014). Learning Activation Functions to Improve Deep Neural Networks. *arXiv e-prints*, page arXiv:1412.6830.
- [3] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., dan Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [4] Chen, T., Fang, C., Shen, X., Zhu, Y., Chen, Z., dan Luo, J. (2020). Anatomy-aware 3D Human Pose Estimation in Videos. *arXiv e-prints*, page arXiv:2002.10322.
- [5] Felzenszwalb, P. dan Huttenlocher, D. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79.
- [6] Gkioxari, G., Hariharan, B., Girshick, R., dan Malik, J. (2014). Using k-poselets for detecting people and localizing their keypoints. pages 3582–3589.
- [7] Guliyev, N. J. dan Ismailov, V. E. (2015). A single hidden layer feedforward network with only one neuron in the hidden layer can approximate any univariate function. *arXiv e-prints*, page arXiv:1601.00013.
- [8] He, K., Zhang, X., Ren, S., dan Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385.
- [9] Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 3:31.

- [10] Hinton, G. E. Rectified linear units improve restricted boltzmann machines
vinod nair.
- [11] Ionescu, C., Papava, D., Olaru, V., dan Sminchisescu, C. (2014). Human3.6m:
Large scale datasets and predictive methods for 3d human sensing in natural
environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,
36(7):1325–1339.
- [12] Kingma, D. P. dan Ba, J. (2014). Adam: A Method for Stochastic
Optimization. *arXiv e-prints*, page arXiv:1412.6980.
- [13] Kratsios, A. (2019). The Universal Approximation Property:
Characterizations, Existence, and a Canonical Topology for Deep-Learning.
arXiv e-prints, page arXiv:1910.03344.
- [14] Martinez, J., Hossain, R., Romero, J., dan Little, J. J. (2017). A simple yet
effective baseline for 3d human pose estimation. In *ICCV*.
- [15] Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Elgharib, M., Fua, P., Seidel,
H.-P., Rhodin, H., Pons-Moll, G., dan Theobalt, C. (2019). XNect: Real-time
Multi-Person 3D Motion Capture with a Single RGB Camera. *arXiv e-prints*,
page arXiv:1907.00837.
- [16] Nwankpa, C., Ijomah, W., Gachagan, A., dan Marshall, S. (2018). Activation
Functions: Comparison of trends in Practice and Research for Deep Learning.
arXiv e-prints, page arXiv:1811.03378.
- [17] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen,
T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E.,
DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai,
J., dan Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance
Deep Learning Library. *arXiv e-prints*, page arXiv:1912.01703.
- [18] Ruder, S. (2016). An overview of gradient descent optimization algorithms.
arXiv e-prints, page arXiv:1609.04747.

- [19] Rumelhart, D. E., Hinton, G. E., dan Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536.
- [20] Sharma, V., Rai, S., dan Dev, A. (2012). A comprehensive study of artificial neural networks.
- [21] Torabi, M. dan Rao, J. (2013). Estimation of mean squared error of model-based estimators of small area means under a nested error linear regression model. *Journal of Multivariate Analysis*, 117:76 – 87.
- [22] You, K., Long, M., Wang, J., dan Jordan, M. I. (2019). How Does Learning Rate Decay Help Modern Neural Networks? *arXiv e-prints*, page arXiv:1908.01878.
- [23] Zhang, J. (2019). Basic Neural Units of the Brain: Neurons, Synapses and Action Potential. *arXiv e-prints*, page arXiv:1906.01703.
- [24] Zhou, X., Huang, Q., Sun, X., Xue, X., dan Wei, Y. (2017). Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach. *arXiv e-prints*, page arXiv:1704.02447.

LAMPIRAN

Lampiran 1: Kelas Dataset

```
class Human36Dataset(Dataset):
    def __init__(self, actions,
                  data_path, is_train=True):
        self.actions, self.data_path,
        self.is_train =
            actions, data_path,
            is_train
        self.inp_list, self.out_list,
        self.key_list = [],
        [], []

    if self.is_train:
        self.data_2d = torch.
            load(data_path/'
            train_2d.pt')
        self.data_3d = torch.
            load(data_path/'
            train_3d.pt')
    else:
        self.data_2d = torch.
            load(data_path/'
            test_2d.pt')
        self.data_3d = torch.
            load(data_path/'
            test_3d.pt')

    for key in self.data_2d.keys():
        assert self.data_2d[key]
            .shape[0] == self.
            data_3d[key].shape
            [0]
        num_file = self.data_2d[
            key].shape[0]
        for i in range(num_file):
            self.inp_list.append
                (self.data_2d[
                    key][i])
            self.out_list.append
                (self.data_3d[
                    key][i])
            self.key_list.append
                (key)

    def __getitem__(self, idx):
        inp = torch.from_numpy(self.
            inp_list[idx]).float()
        out = torch.from_numpy(self.
            out_list[idx]).float()
        return inp, out

    def get_key(self, idx):
        return self.key_list[idx]

    def __len__(self):
        return len(self.inp_list)
```

Lampiran 2: Kelas ResLinear

```
class ResLinear(nn.Module):
    def __init__(self, size, pd=0.5)
```

```
:
    super().__init__()
    self.size = size
    self.relu = nn.ReLU(inplace=
        True)
    self.drop = nn.Dropout(pd)
    # learnable
    self.ln1 = nn.Linear(self.
        size, self.size)
    self.bn2 = nn.BatchNorm1d(
        self.size)
    self.ln3 = nn.Linear(self.
        size, self.size)
    self.bn4 = nn.BatchNorm1d(
        self.size)

    def forward(self, x):
        y = self.drop(self.relu(self.
            .bn2(self.ln1(x))))
        y = self.drop(self.relu(self.
            .bn4(self.ln3(y))))
        return x + y
```

Lampiran 3: Arsitektur Model

```
class Model(nn.Module):
    def __init__(self, size=1024,
                  num_res_lyr=2, pd=0.5):
        super().__init__()
        self.size, self.num_res_lyr,
        self.pd = size,
        num_res_lyr, pd
        self.input_size, self.
        output_size = 32, 48
        self.relu = nn.ReLU(inplace=
            True)
        self.drop = nn.Dropout(self.
            pd)

        # input size
        self.ln_in = nn.Linear(self.
            input_size, self.size)
        self.bn_in = nn.BatchNorm1d(
            self.size)

        # res layers
        self.lins = []
        for i in range(num_res_lyr):
            self.lins.append(
                ResLinear(self.size,
                    self.pd))
        self.lins = nn.ModuleList(
            self.lins)

        # output size
        self.ln_out = nn.Linear(self.
            size, self.output_size)

    def forward(self, x):
        y = self.drop(self.relu(self.
            .bn_in(self.ln_in(x))))
        for i in range(self.
            num_res_lyr):
            y = self.lins[i](y)
        y = self.ln_out(y)
```

```

        return y

# Lampiran 4: Options
class Options():
    def __init__(self):
        # paths
        self.data_path = Path('data')
        self.model_path = Path('model')

        # train options
        self.actions = 'All'
        self.attempt_id = '01'
        self.attempt_path = Path('model')/self.attempt_id

        self.load_ckpt = False

        # train hyper-params
        self.bs = 128
        self.epochs = 10
        self.lr = 1e-3

        # model hyper-params
        self.size = 1024
        self.stages = 2
        self.dropout = 0.5

# Lampiran 5: Pemuatan Data
stat_3d = torch.load(data_path/'stat_3d.pt')
stat_2d = torch.load(data_path/'stat_2d.pt')
rcams = torch.load(data_path/'rcams.pt')

mean_2d = stat_2d['mean']
std_2d = stat_2d['std']
dim_use_2d = stat_2d['dim_use']
dim_ignore_2d = stat_2d['dim_ignore']

mean_3d = stat_3d['mean']
std_3d = stat_3d['std']
dim_use_3d = stat_3d['dim_use']
dim_ignore_3d = stat_3d['dim_ignore']

# Lampiran 6: Instansiasi Dataset dan DataLoader
train_ds = Human36Dataset(
    get_actions(options.actions),
    options.data_path, is_train=True)
train_dl = DataLoader(train_ds,
    batch_size=options.bs, shuffle=True)
test_ds = Human36Dataset(get_actions(
    options.actions), options.data_path, is_train=False)
test_dl = DataLoader(test_ds,
    batch_size=options.bs, shuffle=False)

# Lampiran 7: Algoritma Pelatihan
def train(train_dl, model, criterion
, optimizer, options, mb):
    model.train()
    loss_list = []
    skel_loss_list = []
    for xb, yb in progress_bar(
        train_dl, parent=mb):
        xb, yb = xb.cuda(), yb.cuda()
        yhat = model(xb)
        optimizer.zero_grad()
        loss_skel = criterion(yhat,
            yb)
        loss = loss_skel.mean()
        loss.backward()
        nn.utils.clip_grad_norm_(
            model.parameters(),
            max_norm=1)
        optimizer.step()

        loss_list.append(loss.item())
        skel_loss_list.append(
            loss_skel.data.cpu().
            numpy())

        mb.child.comment = f'train_
            loss:{loss.item()}'
    return loss_list, skel_loss_list

# Lampiran 8: Algoritma Validasi
def test(test_dl, model, criterion,
    options, mb):
    model.eval()
    loss_list = []
    skel_loss_list = []
    for xb, yb in progress_bar(
        test_dl, parent=mb):
        xb, yb = xb.cuda(), yb.cuda()
        with torch.no_grad():
            yhat = model(xb)
            loss_skel = criterion(
                yhat, yb)
            loss = loss_skel.mean()
            loss_list.append(loss.item())
        skel_loss_list.append(
            loss_skel.data.cpu().
            numpy())
        mb.child.comment = f'test_
            loss:{loss.item()}'
    return loss_list, skel_loss_list

# Lampiran 9: Instansiasi Model
model = Model()
model = model.cuda()
model.apply(init_kaiming)
print(f'total_params:{sum(p.numel()
    for p in model.parameters())}')

criterion = nn.MSELoss(reduction='
    none').cuda()
optimizer = optim.Adam(model.
    parameters(), lr=options.lr)

if options.load_ckpt:
    options = torch.load('model/01/

```

```

        options.pt')
    model_state = torch.load(options
        .attempt_path/'last_model.pt
        ')
    optimizer_state = torch.load(
        options.attempt_path/'
        last_optimizer.pt')
    model.load_state_dict(
        model_state)
    optimizer.load_state_dict(
        optimizer_state)

# Lampiran 10: Visualisasi Titik
    Kunci
    key = train_key_list[184]
    plt.figure(figsize=(16,6))
    gs2 = GridSpec(1,2)
    ax1 = plt.subplot(gs2[0])
    ax2 = plt.subplot(gs2[1], projection
        ='3d')

    idx = 200

    ts_2d = utils.unnormalize_data(
        train_set_2d[key][idx], mean_2d,
        std_2d, dim_ignore_2d)[0]

    ts_3d = utils.unnormalize_data(
        train_set_3d[key][idx], mean_3d,
        std_3d, dim_ignore_3d)[0]
    ts_3d = utils.cam_to_world_centered(
        ts_3d, key, rcams)

    utils.show_2d_pose(ts_2d, ax1)
    ax1.invert_yaxis()

    utils.show_3d_pose(ts_3d, ax2)

    plt.show()

# Lampiran 11: Visualisasi Inferensi
## %matplotlib qt
fig = plt.figure(figsize=(15,15))
gs = GridSpec(2, 2)
ax0 = plt.subplot(gs[0])
ax1 = plt.subplot(gs[1])
ax2 = plt.subplot(gs[2])
ax3 = plt.subplot(gs[3], projection=
    '3d')
ax3.view_init(elev=20, azim=70)

img0 = None
img1 = None
for i in range(len(img_lists)):
    # for i in range(3):
        if img0 is None:
            img0 = ax0.imshow(img_ls[i])

        else:
            img0.set_data(img_ls[i])

        if img1 is None:
            img1 = ax1.imshow(out_ls[i])
        else:
            img1.set_data(out_ls[i])

    ax2.clear()
    show_2d_pose(kp_ls[i], ax2)
    ax2.invert_yaxis()

    ax3.clear()
    show_3d_pose(kp3d_ls[i], ax3)

    plt.pause(1e-25)
    plt.draw()
    plt.savefig(f'imgs/asd/bro{i}.
        jpg')

# Lampiran 12: Plot Grafik
train_loss_lists = []
train_mean = []
train_max = []
train_min = []
for i in range(options.epochs):
    tll = torch.load(options.
        attempt_path/f'
        train_loss_list_e{i}.pt')
    train_mean.append(np.mean(tll))
    train_max.append(np.max(tll))
    train_min.append(np.min(tll))
    train_loss_lists.append(tll)

test_loss_lists = []
test_mean = []
test_max = []
test_min = []
for i in range(options.epochs):
    tll = torch.load(options.
        attempt_path/f'
        test_loss_list_e{i}.pt')
    test_mean.append(np.mean(tll))
    test_max.append(np.max(tll))
    test_min.append(np.min(tll))
    test_loss_lists.append(tll)

plt.ylabel("Rata-Rata_Kesalahan")
plt.xlabel("Epoch")

plt.plot(train_mean, label="
    pelatihan", linewidth=1)
plt.plot(test_mean, label="validasi"
    , linewidth=1)
plt.xticks(range(0, 10))

plt.legend()
plt.savefig("brrr.jpg", dpi=500)
plt.show()

```