TUGAS AKHIR - IF184802

# Pengenalan Aktivitas Manusia Secara Real Time Menggunakan Metode Convolutional Neural Network dan Deep Gated Recurrent Unit

**RASYID FAJAR**
**NRP 05111640000119**

Dosen Pembimbing I
**Dr. Eng. Nanik Suciati, S.Kom., M.Kom.**

Dosen Pembimbing II
**Dini Adni Navastara, S.Kom., M.Sc.**

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

**TUGAS AKHIR – IF184802**

# Pengenalan Aktivitas Manusia Secara Real Time Menggunakan Metode Convolutional Neural Network dan Deep Gated Recurrent Unit

**RASYID FAJAR**
**NRP 05111640000119**

**Dosen Pembimbing I**
**Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**

**Dosen Pembimbing II**
**Dini Adni Navastara, S.Kom., M.Sc.**

**Departemen Teknik Informatika**
**Fakultas Teknologi Elektro dan Informatika Cerdas**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2020**

*(Halaman ini sengaja dikosongkan)*


UNDERGRADUATE THESIS – IF184802

# Real Time Human Activity Recognition Based on Convolutional Neural Network and Deep Gated Recurrent Unit

**RASYID FAJAR**
**NRP 05111640000119**

**First Advisor**
**Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**

**Second Advisor**
**Dini Adni Navastara, S.Kom., M.Sc.**

**Department of Informatics**
**Faculty of Intelligent Electrical and Informatics Technology**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2020**

*(Halaman ini sengaja dikosongkan)*

# LEMBAR PENGESAHAN

## PENGENALAN AKTIVITAS MANUSIA SECARA REALTIME MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DAN DEEP GATED RECURRENT UNIT

## TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi Program Studi S-1 Departemen Teknik Informatika Fakultas Teknologi Elektro Dan Informatika Cerdas Institut Teknologi Sepuluh Nopember

Oleh:
**Rasyid Fajar**
**NRP: 05111640000119**

Disetujui oleh Pembimbing tugas akhir:

1. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
   (NIP. 19710428199422000) (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   (NIP. 198510172015042001) (Pembimbing 2)

**SURABAYA**
**JUNI, 2020**

*(Halaman ini sengaja dikosongkan)*

**PENGENALAN AKTIVITAS MANUSIA SECARA REALTIME MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DAN DEEP GATED RECURRENT UNIT**


**Nama : Rasyid Fajar**
**NRP : 05111640000119**
**Departemen : Teknik Informatika, FTEIC-ITS**
**Pembimbing I : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**
**Pembimbing II : Dini Adni Navastara, S.Kom., M.Sc.**


## ABSTRAK


*Recurrent neural network (RNN) telah mencapai kesuksesan dalam memproses data sekuensial dan menjadi state-of-the-art dalam speech recognition, pengenalan sinyal digital, pemrosesan video, dan analisa data teks. Pada penelitian ini, diimplementasikan metode pengenalan aktivitas manusia dengan memproses data video menggunakan convolutional neural network (CNN) dan deep gated recurrent unit (GRU). Pertama, akan dilakukan pemilihan frame dengan cara memilih frame dengan urutan kelipatan enam. Hal ini dilakukan untuk mengurangi kompleksitas dan mengurangi fitur yang redundan. Fitur dari sebuah frame akan diekstrak menggunakan CNN dengan arsitektur MobileNetV2 yang sudah dilatih pada dataset ImageNet. Kemudian, fitur yang sudah diekstrak menggunakan CNN akan dimasukkan ke GRU dengan tujuan untuk menganalisa fitur spatiotemporal. Hasil penelitian ini dapat mencapai F1 Score sebesar 92.01% pada dataset YouTube 11 Actions. Metode ini dapat mencapai kecepatan sebesar 65.43 FPS.*



**Kata kunci:** ***Human Activity Recognition, Video Analysis, Gated Recurrent Unit, Convolutional Neural Network***

*(Halaman ini sengaja dikosongkan)*

**REAL TIME HUMAH ACTIVITY RECOGNITION BASED ON CCONVOLUTIONAL NEURAL NETWORK AND DEEP GATED RECURRENT UNIT**

**Student's Name : Rasyid Fajar**
**Student's ID : 05111640000119**
**Department : Informatics, Faculty of ELECTICS-ITS**
**First Advisor : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**
**Second Advisor : Dini Adni Navastara, S.Kom., M.Sc.**

## ABSTRACT

*Recurrent neural network (RNN) have achieved great success in processing sequential data and yielded the state-of-the-art results in speech recognition, digital signal processing, video processing, and text data analysis. In this research, proposed a human action recognition method by processing the video data using convolutional neural network (CNN) and deep gated recurrent unit (GRU) network. First, frame selection will be carried out by selecting frames in multiples of six. This is done to reduce complexity and reduce redundant features. The features of a frame will be extracted using CNN with the MobileNetV2 architecture that has been trained in the ImageNet dataset. Then, features that have been extracted using CNN will be fed to the GRU to analyze spatiotemporal features. The results of this study achieved an F1 Score of 92.01% on the YouTube 11 Actions dataset. This method achieved speeds of 65.43 FPS.*

**Keywords:** ***Human Activity Recognition, Video Analysis, Gated Recurrent Unit, Convolutional Neural Network***

*(Halaman ini sengaja dikosongkan)*

# KATA PENGANTAR

Puji syukur saya sampaikan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya saya dapat melaksanakan Tugas Akhir yang berjudul:

"**PENGENALAN AKTIVITAS MANUSIA SECARA REALTIME MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK DAN DEEP GATED RECURRENT UNIT**"

Terselesaikannya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan banyak pihak, oleh karena itu melalui lembar ini penulis ingin mengucapkan terima kasih dan penghormatan kepada:

1. Orang tua dan keluarga penulis, yang telah memberikan dukungan doa, moral, dan material kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Dr.Eng. Nanik Suciati, S.Kom., M.Kom. dan Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing dan memberikan motivasi, nasihat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. selaku Ketua Departemen Teknik Informatika ITS dan seluruh dosen dan karyawan Departemen Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Teknik Informatika ITS.
4. Nuzha Musyafira yang telah membantu dalam penulisan buku Tugas Akhir ini.
5. Iftina I. U., Gigih bin Ganis, dan Raihan bin Haspin yang telah meminjamkan akun gmail untuk menjalankan Tugas Akhir ini.

6. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa laporan Tugas Akhir ini masih memiliki banyak kekurangan. Oleh karena itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan penulis kedepannya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum.

Surabaya, Juni 2020

# DAFTAR ISI

# DAFTAR TABEL

*(Halaman ini sengaja dikosongkan)*

# DAFTAR KODE SUMBER

*(Halaman ini sengaja dikosongkan)*

# DAFTAR GAMBAR

*(Halaman ini sengaja dikosongkan)*

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Pengenalan aktivitas pada rangkaian video adalah masalah yang menantang pada visi computer karena kemiripan dari konten visual [1], perubahan sudut pandang untuk aktivitas yang sama, gerakan kamera terhadap pelaku aktivitas, skala dan pose dari actor, dan perbedaan perubahan pencahayaan [2]. Aktivitas manusia sangat beragam, dari aktivitas sederhana melalui tangan dan kaki, hingga aktivitas yang kompleks dan terintegrasi antara tangan, kaki, dan tubuh. Contohnya, gerakan kaki untuk menendang bola adalah gerakan sederhana, sedangkan melompat untuk menyundul adalah gerakan kolektif antara tangan, kaki, kepala, dan tubuh [3]. Secara umum, aktivitas manusia adalah gerakan dari bagian tubuh dengan berinteraksi dengan objek pada lingkungan. Pada konteks video, aktivitas direpresentasikan menggunakan rangkaian frame, di mana manusia dapat memahami dengan mudah dengan menganalisa konten dari beberapa frame di suatu rangkaian frame. Pada penelitian ini, aktivitas manusia dikenali dengan cara yang mirip dengan pengamatan untuk aktivitas di dunia nyata. Penelitian ini menggunakan GRU untuk mempertimbangkan informasi dari frame sebelumnya untuk memahami aktivitas pada video

Salah satu motivasi yang menarik untuk melakukan riset di bidang pengenalan aktivitas adalah domainnya yang luas dari pengaplikasiannya dalam video pengawasan [4], robotik, interaksi manusia-komputer [5], analisa olahraga, permainan video untuk karakter pemain, dan manajemen video web [6]. Pengenalan aktivitas menggunakan analisa video membutuhkan komputasi yang besar karena video pendek dapat membutuhkan waktu yang lama karena frame rate yang tinggi. Setiap frame memiliki peran penting dalam sebuah video, menyimpan informasi dari rangkaian frame yang panjang menjadikan sistem lebih baik. Peneliti telah memberikan banyak solusi seperti gerakan, fitur space-time [7],

dan trajectory [8]. RNN adalah sebuah blok neuron yang terhubung dengan input unit, output unit, dan memiliki aktivasi pada waktu $t$, yang dapat memproses rangkaian data. GRU memproses satu elemen pada satu waktu sehingga dapat memodelkan output yang terdiri dari rangkaian elemen yang tidak independen [9]. Arsitektur RNN dapat mencari pola tersembunyi di dalam data *time-space* seperti video, audio, dan teks. RNN memproses data dalam sebuah rangkaian pada waktu $t$ mendapatkan input dari hidden state sebelumnya $s_{t-1}$ dan data baru $x_t$. Data dikalikan dengan weight dan ditambah bias dan dimasukkan ke dalam fungsi aktivasi. Karena jumlah komputasi yang besar, semakin mendapatkan rangkaian data lebih banyak atau setelah beberapa layer akan semakin mengabaikan efek dari input awal yang menyebabkan vanishing gradient problem. Solusi dari masalah ini adalah LSTM. Arsitektur LSTM memiliki sel memori, gerbang input, gerbang output, dan gerbang forget yang dapat menjaga state dari waktu ke waktu dan gerbang non linier yang meregularisasi aliran informasi yang masuk dan keluar dari sel [10]. Para peneliti telah menyajikan beberapa jenis LSTM seperti multi-layer LSTM dan bidirectional LSTM untuk memproses data sekuensial. Deep bidirectional LSTM (DB-LSTM) juga digunakan dalam pengenalan aktivitas manusia [11]. LSTM memiliki komputasi yang cukup besar, sehingga diperkenalkan metode GRU oleh Cho et al 2014 [12]. GRU memiliki kemiripan dengan LSTM yaitu menggunakan gerbang yang mengatur aliran informasi, tetapi GRU tidak memiliki sel memori sehingga GRU memiliki komputasi yang lebih efisien dibandingkan dengan LSTM [13].

Pada penelitian ini, fitur dari video dianalisa untuk pengenalan aktivitas. Untuk mengurangi fitur frame yang redundan dan mengurangi waktu kompuatasi, fitur dari setiap frame kelipatan enam akan diekstrak menggunakan MobileNetV2 yang sudah ditrain pada ImageNet [14]. Arsitektur selanjutnya adalah GRU untuk mempelajari rangkaian informasi dari frame pada video. Untuk mempelajari fitur yang lebih kompleks, digunakan dua layer GRU.

## 1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana proses pengenalan aktivitas manusia menggunakan CNN dan GRU?
2. Bagaimana performa CNN dan GRU pada pengenalan aktivitas?
3. Bagaimana kecepatan CNN dan GRU pada pengenalan aktivitas pada komputer tanpa GPU?

## 1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Dataset yang dipakai bersumber dari YouTube 11 Action
2. Menggunakan bahasa Python 3.

## 1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun sebuah sistem pengenalan aktivitas manusia dengan metode Convolutional Neural Network dan Gated Recurrent Unit yang dapat mengenali aktivitas pada data video secara real time.

## 1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu menambah kemampuan yang ada pada pengenalan aktivitas manusia sehingga dapat diimplementasikan pada sistem-sistem yang membutuhkan pengenalan aktivitas manusia secara otomatis seperti video pengawasan.

## 1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

- Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir yang berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri dari latar belakang diajukannya Tugas Akhir, rumusan masalah dan batasan masalah yang ditetapkan, serta manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

- Studi Literatur

Pada tahap ini dilakukan pencarian literatur berupa jurnal yang digunakan sebagai referensi untuk pengerjaan tugas akhir ini. Literatur yang dipelajari pada pengerjaan tugas akhir ini berasal dari jurnal ilmiah yang diambil dari berbagai sumber di internet, beserta berbagai literatur online tambahan terkait *Neural Network*, TensorFlow dan Keras.

- Implementasi Perangkat Lunak

Pada tahap ini akan dilaksanakan implementasi metode dan algoritma yang telah direncanakan. Implementasi sistem menggunakan Python 3 sebagai bahasa pemrograman, TensorFlow dan Keras sebagai *framework*, serta *library* pendukung lainya.

- Pengujian dan Evaluasi

Tahap pengujian dan evaluasi dilakukan menggunakan 10% data video yang diambil dari dataset Youtube 11 Action yang sisanya dijadikan data trainin. Evaluasi dilakukan dengan mengevaluasi model dari segi akurasi, *precision*, *recall*, serta F-measure untuk melihat performa model yang telah dibuat.

- Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang

digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

## 1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

**Bab I Pendahuluan**
Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

**Bab II Tinjauan Pustaka**
Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang *Neural Network* dan *library* yang digunakan.

**Bab III Perancangan Sistem**
Bab ini berisi pembahasan mengenai perancangan dari metode *Single Shot Detector* dan *Recurrent Neural Network* yang digunakan untuk pengenalan plat nomor kendaraan pada data video.

**Bab IV Implementasi**
Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode sumber yang digunakan untuk proses implementasi.

**Bab V Uji Coba Dan Evaluasi**
Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi terhadap kinerja dari sistem yang dibangun.

**Bab VI Kesimpulan dan Saran**
Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah

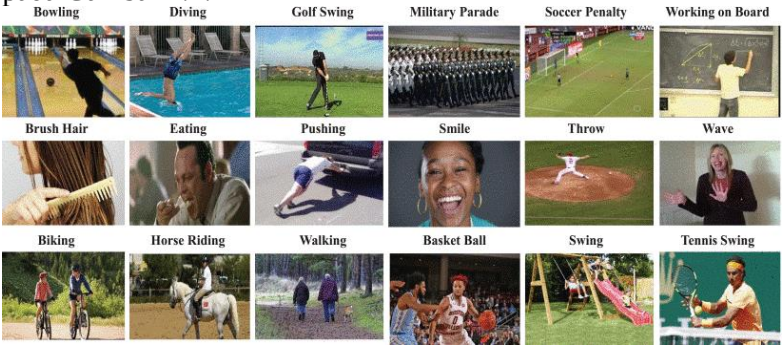
yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

# BAB II
# TINJAUAN PUSTAKA

Bab ini membahas mengenai teori-teori dasar yang digunakan dalam penelitian ini. Teori-teori tersebut adalah *Neural Network* dan beberapa teori lain yang mendukung penelitian. Penjelasan ini bertujuan untuk memberikan gambaran umum dan diharapkan dapat mendukung sistem yang dibangun.

## 2.1 Pengenalan Aktivitas Manusia

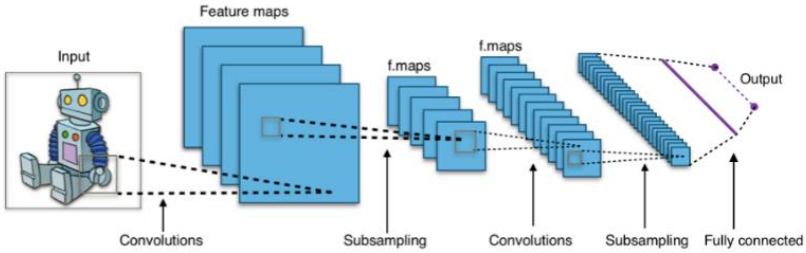
Pengenalan aktivitas manusia adalah salah satu permasalahan yang menantang di bidang kecerdasan buatan. Pengenalan aktivitas manusia berisi kecerdasan buatan yang dilatih untuk mengenali kelas kelas dari aktivitas manusia [15]. Aktivitas manusia berkisar dari aktivitas sederhana lengan atau kaki hingga aktivitas terintegrasi yang rumit antara tangan, kaki, dan tubuh. Misalnya, gerakan kaki untuk berjalan. Secara umum, tindakan manusia adalah gerakan bagian-bagian tubuh dengan berinteraksi dengan benda-benda di lingkungan. Dalam konteks video, suatu tindakan direpresentasikan menggunakan rangkaian frame, yang mudah dipahami manusia dengan menganalisis konten beberapa frame secara berurutan. Contoh aktivitas manusia dapat dilihat pada Gambar 2.1.




Gambar 2.1 Ilustrasi aktivitas manusia [11]

## 2.2 Convolutional Neural Network (CNN)

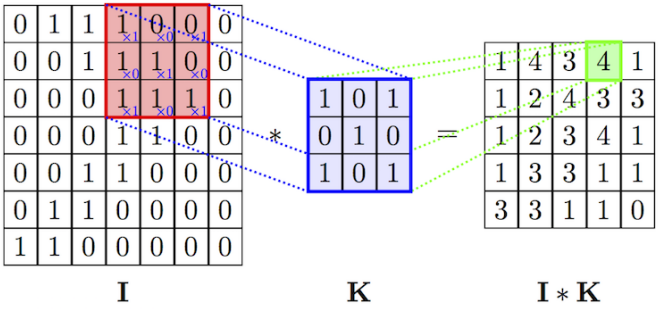
Di dalam ilmu deep learning, convolutional neural network (CNN, atau ConvNet) adalah kelas deep neural network yang paling sering digunakan untuk menganalisa gambar. CNN juga digunakan pada klasifikasi gambar, analisis gambar medis, dan natural language processing [16]. Contoh arsitektur CNN dapat dilihat pada Gambar 2.2.




Gambar 2.2 Contoh arsitektur CNN [17]

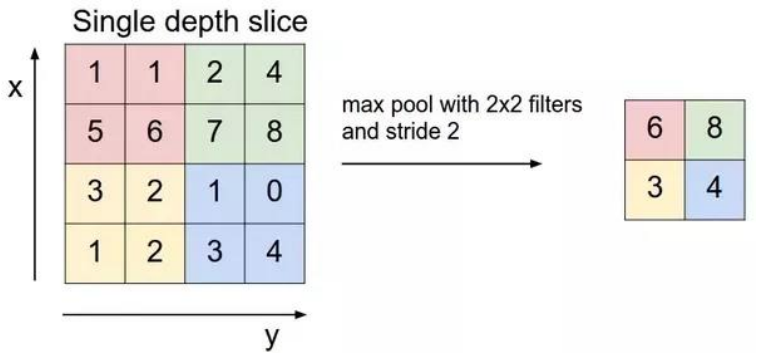
- Convolution Layer

*Convolution Layer* melakukan operasi konvolusi pada output dari lapisan sebelumnya. Konvolusi adalah istilah matematis yang artinya mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrak fitur dari citra masukan [18]. Ilustrasi cara kerja konvolusi bisa dilihat pada Gambar 2.3, dimana I adalah citra, K adalah *filter* atau kernel yang digunakan, I * K adalah hasil operasi konvolusi.




Gambar 2.3 Ilustrasi cara kerja konvolusi [19]

- Pooling Layer




Gambar 2.4 Ilustrasi cara kerja *Max Pooling* [20]

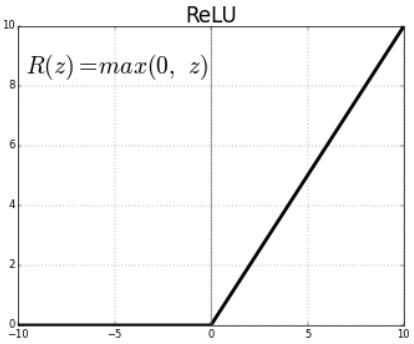
Fungsi dari *Pooling Layer* adalah mereduksi ukuran dari data. Terdapat beberapa tipe *Pooling Layer* diantaranya yaitu *max*, *average*, *sum* dan lainnya. Metode *Pooling* dalam *Convolutional Neural Network* (CNN) yang biasa digunakan adalah *Max Pooling* & *Average Pooling*. *Max Pooling* membagi *output* dari *Convolution Layer* menjadi beberapa matriks kecil lalu mengambil nilai maksimal dari tiap matriks untuk menyusun matriks citra yang telah direduksi, sedangkan *Average Pooling* akan memilih nilai rata-ratanya. Proses tersebut memastikan fitur yang didapatkan

akan sama meskipun obyek citra mengalami translasi. Ilustrasi cara kerja *Max Pooling* bisa dilihat pada Gambar 2.4.

- Fully Connected Layer

*Fully Connected Layer* dalam penerapannya sama dengan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. *Feature map* dari *Convolution Layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu yang disebut *feature vector* sebelum dapat dimasukkan ke dalam sebuah *Fully Connected Layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *Fully Connected Layer* diimplementasikan di akhir jaringan [21].

- ReLU Activation Function




Gambar 2.5 ReLU *Activation Function* [22]

Fungsi aktivasi befungsi untuk menentukan apakah *neuron* tersebut harus aktif atau tidak berdasarkan nilai masukan. Salah satu contoh fungsi aktivasi adalah ReLU (*Rectified Linear Unit*) dimana fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai masukan, dimana seluruh nilai yang kurang dari nol akan dijadikan nol, seperti pada Gambar 2.5.

- Fungsi Softmax

Fungsi *softmax* biasa digunakan dalam klasifikasi banyak kelas. *Softmax* memberikan nilai probabilitas untuk setiap label kelas, dimana jumlah seluruh probabilitas adalah 1. *Softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari nilai masukan sejumlah kelas pada model klasifikasi seperti pada Persamaan (2.1).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.1}$$

Dimana *y* adalah nilai masukan. Operasi akan menghasilkan nilai probabilitas. Label dari data masukan akan ditentukan berdasarkan kelas dengan nilai probabilitas tertinggi.

- Cross Entropy

*Loss function* merupakan fungsi yang menggambarkan kerugian yang dihasilkan oleh model. *Loss function* dikatakan baik, ketika menghasilkan *error* yang diharapkan paling rendah. Pada permasalahan klasifikasi banyak kelas, *cross entropy* adalah *loss function* yang biasa digunakan. *Cross entropy* akan menghitung *error* antara nilai prediksi *S* dengan nilai sebenarnya *T*, seperti pada Persamaan (2.2). Selanjutnya, nilai *error* akhir diambil dari rata-rata hasil *cross entropy*, seperti pada Persamaan (2.3).

$$D(S_i, T_i) = -\sum_j T_{ij} \log S_{ij} \tag{2.2}$$

$$J(W, b) = \frac{1}{n}\sum_i D(S_i, T_i) \tag{2.3}$$

Di mana $D(S_i, T_i)$ adalah cross entropy pada suatu kelas, dan $J(W, b)$ adalah rata-rata cross entropy dari seluruh kelas.

- Stochastic Gradient Descent

Ketika melatih sebuah model, dibutuhkan sebuah *loss function* yang dapat mengukur kualitas dari setiap bobot atau

parameter tertentu. *Stochastic Gradient Descent* (SGD) adalah algoritma pengoptimalan. Tujuan pengoptimalan adalah untuk menemukan parameter yang dapat meminimalkan nilai *error* dari *loss function.* SGD adalah algoritma yang digunakan untuk memperbarui nilai bobot dan bias pada neuron di *neural network*. Pada dasarnya operasi yang dilakukan hanya mengurangi bobot awal dengan sebagian nilai dari nilai gradien yang sudah kita dapat. Nilai sebagian disini diwakili oleh parameter bernama *learning rate*, seperti yang terlihat pada Persamaan (2.4) dan (2.5).

$$w_{j+1} = w_j - \alpha \frac{\partial}{\partial w_j} J(W, b) \tag{2.4}$$

$$b_{j+1} = b_j - \alpha \frac{\partial}{\partial b_j} J(W, b) \tag{2.5}$$

Di mana w dan b adalah bobot dan bias, $\alpha$ adalah *learning rate*, dan $\frac{\partial}{\partial w_j} J(W, b)$ adalah turunan dari fungsi loss terhadap bobot.

- RMSProp

RMSProp (*Root Mean Square Propagation*) adalah metode pengoptimalan berbasis *adaptive learning rate* yang diusulkan oleh Geoffrey Hinton [23]. RMSProp memodifikasi Adagrad dengan mengganti akumulasi gradien menjadi rata-rata bergerak gradien yang diberi bobot secara kuadratik, seperti yang dapat dilihat pada Persamaan (2.6) dan (2.7).

$$s_j = \beta . s_{j-1} + (1 - \beta)(g_j)^2 \tag{2.6}$$

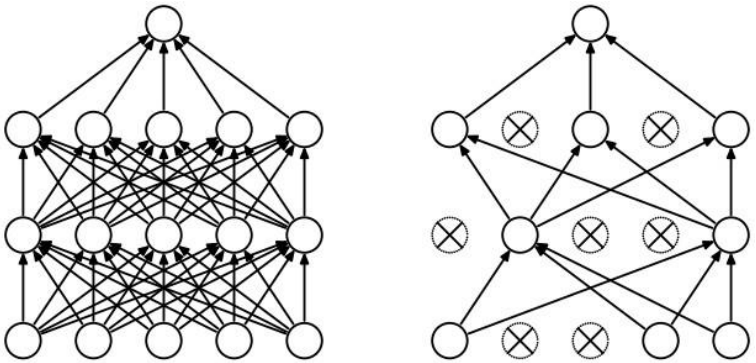
$$\theta_{j+1} = \theta_j - \frac{\alpha}{\sqrt{s_j + \varepsilon}} g_j \tag{2.7}$$

Di mana $s$ adalah *moving average* kuadrat dari gradien, $\beta$ adalah parameter *moving average*, $g$ adalah turunan dari fungsi loss terhadap bobot, $\theta$ adalah bobot, $\alpha$ adalah *learning rate*, dan $\varepsilon$ adalah angka konstan yang kecil untuk stabilitas numerik.

- Dropout

*Dropout* merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses learning. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden layer* mapun *visible layer* di dalam jaringan [24]. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Neuron yang akan dihilangkan akan dipilih secara acak.

Pada Gambar 2.6 (a) neuron tetap utuh pada *neural network* yang belum memakai *Dropout*, dan (b) *neural network* yang sebagian dari neuronnya tidak digunakan setelah diaplikasikan Dropout.




Gambar 2.6 Ilustrasi *neural network* mengaplikasikan *Dropout* [25]

- MobileNet-V2

MobileNet adalah arsitektur CNN yang efisien untuk perangkat mobile dan aplikasi embedded vision. MobileNet menggunakan depth-wise separable convolution untuk membangun deep neural network yang ringan. MobileNets juga efektif dalam banyak aplikasi seperti deteksi objek, klasifikasi gambar, atribut wajah, dan geo-lokalisasi skala besar [26].

MobileNetV2 adalah perkembangan dari MobileNet yang berhasil meningkatkan performa model pada beberapa masalah. Arsitektur MobileNetV2 berbasis pada struktur inverted residual di mana input dan output dari residual block adalah layer bottleneck yang tipis, berbeda dengan model residual tradisional di mana menggunakan expanded representation pada input. Seperti pendahulunya, MobileNetV2 menggunakan depth wise convolution untuk mengambil fitur. Arsitektur MobileNetV2 dapat dilihat pada Tabel 2.1.

Tabel 2.1 Arsitektur MobileNetV2

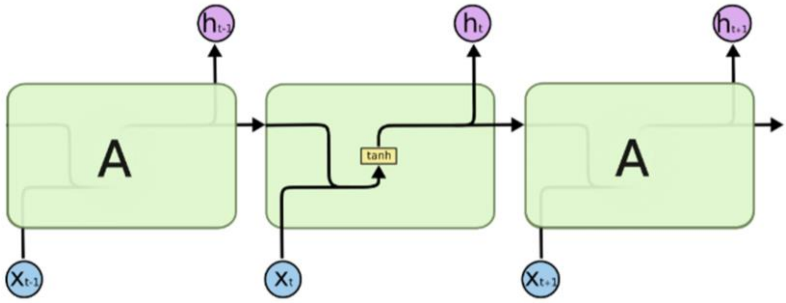
| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 32 | 1 | 2 |
| $112^2 \times 32$ | bottleneck | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | bottleneck | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | bottleneck | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | bottleneck | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | bottleneck | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | bottleneck | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | bottleneck | 6 | 320 | 1 | 1 |
| $7^2 \times 320$ | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2 \times 1280$ | avgpool 7x7 | - | - | 1 | - |
| $1 \times 1 \times 1280$ | conv2d 1x1 | - | k | - | |

Bottleneck layer adalah layer yang memiliki lebih sedikit neuron dari layer sebelumnya. Layer ini dapat mengurangi jumlah representasi fitur sehingga dapat mengurangi komputasi. Fitur dapat ditangkap dan dieksploitasi dengan layer bottleneck. Cara ini telah berhasil dilakukan pada MobileNetV1 untuk mendapatkan *trade off* terbaik antara komputasi dan akurasi [14].

## 2.3 Recurrent Neural Network

*Recurrent Neural Network* (RNN) adalah salah satu metode pembelajaran mesin yang berfokus pada prediksi dari hasil pola.

Secara umum, RNN mengingat masa lalu dan membuat keputusan yang dipengaruhi oleh apa yang telah dipelajari dari masa lalu [27]. Selagi RNN belajar dengan cara yang sama saat pelatihan, di samping itu, RNN mengingat hal-hal yang dipelajari dari masukan sebelumnya saat menghasilkan keluaran. RNN dapat mengambil satu atau lebih vektor masukan dan menghasilkan satu atau lebih vektor keluaran. Keluaran tidak hanya dipengaruhi oleh bobot yang diterapkan pada masukan seperti *Neural Network* biasa, tetapi juga oleh vektor status "tersembunyi" yang mewakili konteks berdasarkan masukan atau keluaran sebelumnya. Jadi, masukan yang sama dapat menghasilkan keluaran yang berbeda tergantung pada masukan sebelumnya dalam suatu seri. Ilustrasi RNN sederhana dapat dilihat pada Gambar 2.7.




Gambar 2.7 Contoh arsitektur jaringan RNN

## 2.4 Gated Recurrent Unit

Gated Recurrent Unit (GRU) adalah metode pengembangan dari RNN yang bekerja sangat baik dalam mengatasi masalah sekuensial [12]. GRU memiliki update gate dan reset gate untuk mengatasi vanishing gradient problem. GRU memiliki komputasi yang cukup ringan jika dibandingkan dengan LSTM. Formula dari GRU ke-j pada waktu t dapat dilihat pada Persamaan (2.8), (2.9), (2.10), dan (2.11).

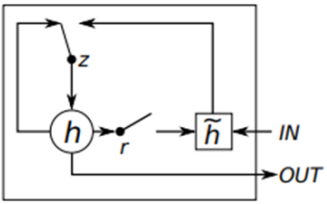
$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \tilde{h}_t^j \quad (2.8)$$

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \quad (2.9)$$

$$\tilde{h}_t^j = tanh(W x_t + U(r_t \odot h_{t-1}))^j \quad (2.10)$$

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \quad (2.11)$$

Di mana r adalah *reset gate*, z *update gate*, h adalah hasil aktivasi, dan h˜ adalah kandidat hasil aktivasi, W dan U adalah bobot, σ adalah fungsi sigmoid, dan Θ adalah Hadamard product [13]. Ilustrasi arsitektur GRU dapat dilihat pada Gambar 2.8.




Gambar 2.8 Ilustrasi GRU [13].

## 2.5 Confusion Matrix

*Confusion Matrix* adalah pengukuran kinerja untuk masalah klasifikasi pembelajaran mesin di mana keluaran bisa sebanyak dua atau lebih kelas [28]. Berikut adalah tabel dengan 4 kombinasi nilai prediksi dan aktual yang berbeda. Ilustrasi confusion matrix dapat dilihat pada Tabel 2.2.

Tabel 2.2 Ilustrasi Confusion Matrix.

| | | **Nilai Aktual** | |
|---|---|---|---|
| | | Positif | Negatif |
| **Nilai Prediksi** | Positif | TP | FP |
| | Negatif | FN | TN |

*TP* (*True Positive*) adalah ketika yang diprediksi positif dan kenyataannya benar. Contohnya, wanita yang diprediksi hamil dan sesungguhnya benar hamil. *TN* (*True Negative*) adalah ketika yang diprediksi negatif dan kenyataannya benar. Contohnya, lelaki yang diprediksi tidak hamil dan sesungguhnya benar tidak hamil. *FP* (*False Positive*) adalah ketika yang diprediksi positif namun kenyataannya salah. Contohnya, lelaki yang diprediksi hamil namun sesungguhnya tidak hamil. *FN* (*False Negative*) adalah ketika yang diprediksi negatif namun kenyataannya salah. Contohnya, wanita yang diprediksi tidak hamil namun sesungguhnya hamil.

Nilai-nilai di atas dapat digunakan untuk mengukur tingkat validasi data. Beberapa jenis teknik validasi yang umum digunakan antara lain:

$$Precision = \frac{TP}{TP+FP} \tag{2.12}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.13}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (2.14)$$

## 2.6 Python

Python adalah bahasa pemograman yang populer. Python sering dimanfaatkan dalam pengembangan web, perangkat lunak, penelitian, dan *system scripting*. Python dapat digunakan untuk menangani data besar dan melakukan operasi matematika yang kompleks. Python bekerja di berbagai *platform* seperti Windows, Mac, Linux, Raspberry Pi, dan lain-lain. Python dirancang untuk mudah dibaca, yaitu memiliki sintaks yang sederhana dan menggunakan bahasa inggris [29].

## 2.7 Library

*Library* merupakan sekumpulan program yang dapat digunakan pada program lain tanpa terikat satu dengan yang lainnya. Terdapat beberapa *library* yang digunakan dalam melakukan implementasi penelitian ini. *Library* yang digunakan antara lain:

- Keras

Keras adalah *high-level neural networks* API, yang ditulis dalam bahasa pemograman Python dan mampu berjalan di atas TensorFlow dan Theano. Keras dikembangkan dalam rangka memungkinkan eksperiman dilakukan dengan cepat. Keras dapat berjalan baik di CPU dan GPU. Keras berisi banyak implementasi *neural network* yang umum digunakan, fungsi aktivasi, *optimizer*, dan *tool* lain yang memudahkan dalam pengolahan citra dan data teks [30].

- TensorFlow

TensorFlow adalah *library open source* untuk pembuatan program yang membutuhkan komputasi numerik berkinerja tinggi. TensorFlow dikembangakan oleh tim Google Brain. TensorFlow

menyediakan fungsi-fungsi *machine learning* dan *deep learning*, dan dapat dijalankan dalam CPU atau GPU [31].

- OpenCV

OpenCV (*Open Source Computer Vision*) adalah *library* yang dimanfaatkan dalam pengolahan citra dinamis secara *real-time*. OpenCV dapat digunakan dalam berbagai bahasa pemograman seperti Python, C++, Java, atau MATLAB. OpenCV memiliki fitur seperti *Feature & Object Detection, Motion Analysis and Object Tracking, Image Filtering, Image Processing*, dan lain-lain [32].

- Numpy

Numpy adalah *library Python* yang mendukung pengolahan data pada *array* dan matriks multidimensi yang besar. Numpy menyediakan kumpulan fungsi matematika, seperti aljabar linear, transformasi Fourier, pembuatan angka acak, dan lain-lain. *Numpy* bersifat *open source* sehingga banyak dimanfaatkan dalam pengolahan data peneilitian [33].

- Matplotlib

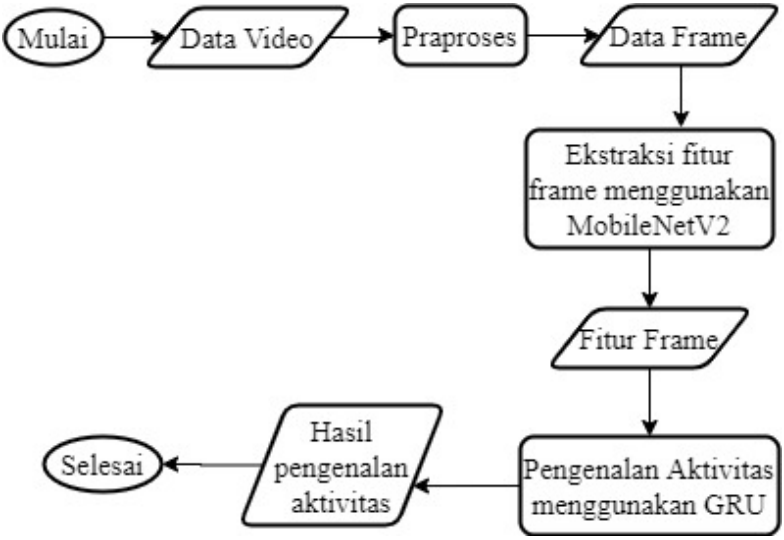
Matplotlib adalah *library Python* yang mendukung pembuatan grafik dua dimensi dalam berbagai format dan dari berbagai jenis data. Matplotlib bersifat *open source* dan banyak digunakan untuk pengolahan data dalam penelitian. Matplotlib dapat membuat plot, histogram, spektrum daya, diagram batang, diagram kesalahan, plot pencar, dan lain-lain [34].

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

# BAB III
# PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan data dan sistem pengenalan aktivitas manusia menggunakan MobileNetV2 dan Gated Recurrent Unit (GRU). Bab ini juga akan menjelaskan gambaran umum sistem dalam bentuk diagram alir.

## 3.1 Desain Umum Sistem




Gambar 3.1 Diagram alir sistem yang dibangun

Sistem pengenalan aktivitas manusia yang dibangun memiliki proses utama di antaranya praprosesing, ekstraksi fitur frame, dan pengenalan aktivitas. Diagram alir dari sistem ditunjukkan pada Gambar 3.1. Arsitektur model yang dibangun dapat dilihat pada Tabel 3.1.

Tabel 3.1 Arsitektur model yang dibangun

| Layer (type) | Output Shape | Param # |
|---|---|---|
| time_distributed_1 (TimeDist | (1, None, 7, 7, 1280) | 2257984 |
| time_distributed_2 (TimeDist | (1, None, 7, 7, 1280) | 0 |
| time_distributed_3 (TimeDist | (1, None, 62720) | 0 |
| gru_1 (GRU) | (1, None, 500) | 94831500 |
| dropout_2 (Dropout) | (1, None, 500) | 0 |
| gru_2 (GRU) | (1, 500) | 1501500 |
| dropout_3 (Dropout) | (1, 500) | 0 |
| dense_1 (Dense) | (1, 11) | 5511 |

## 3.2 Deskripsi Dataset

Dataset yang digunakan adalah YouTube11 Action. Data yang digunakan adalah data video yang berisi satu aktivitas tertentu. Data dibagi menjadi 11 kelas yang terdiri dari bermain bola basket, bersepeda, lompat indah, bermain golf, naik kuda, sepak bola, bermain ayunan, bermain tenis, melompat, bermain voli, dan berjalan bersama anjing. Dataset terdiri dari 25 subjek yang berbeda dengan masing masing subjek memiliki lebih dari 4 video klip. Video dalam satu subjek memiliki kemiripan fitur seperti aktor yang sama, background yang sama, dan sudut pandang yang sama. Dataset memiliki variasi yang besar pada pergerakan kamera, penampakan objek, pose, skala objek, sudut pandang, background yang berantakan, dan kondisi pencahayaan yang membuat dataset ini lebih menantang. Contoh dataset dapat dilihat pada Gambar 3.2.


Gambar 3.2 Contoh dataset yang digunakan [35]

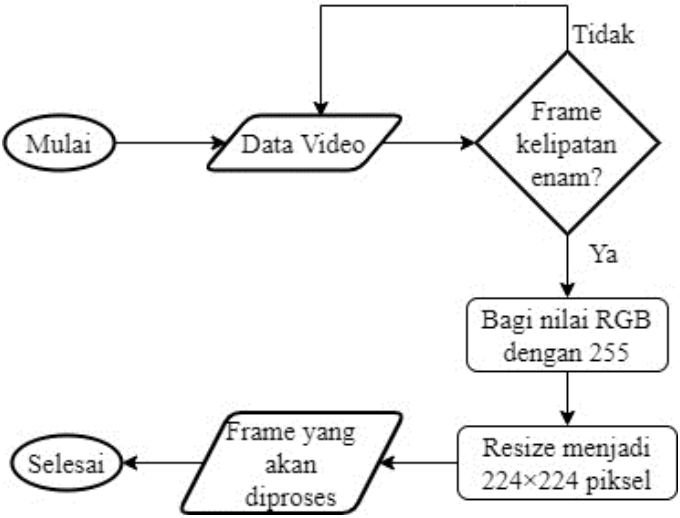
## 3.3 Tahap Praproses

Tahap preproses dibagi menjadi tiga bagian yaitu memilih frame, resize, dan normalisasi. Sebuah video dengan frame rate yang tinggi memiliki banyak fitur yang redundan dan memiliki kompleksitas yang besar, maka dari itu tidak diproses seluruh frame pada video. Frame yang dipilih hanya frame pada kelipatan 6. Kemudian frame akan diresize menjadi ukuran 224x224 pixel agar sesuai dengan ukuran input untuk MobileNetV2. Setelah itu dilakukan normalisasi dengan cara membagi nilai RGB dengan 255 agar setiap nilainya di antara 0 dan 1. Diagram alir tahap praproses dapat dilihat pada Gambar 3.3.

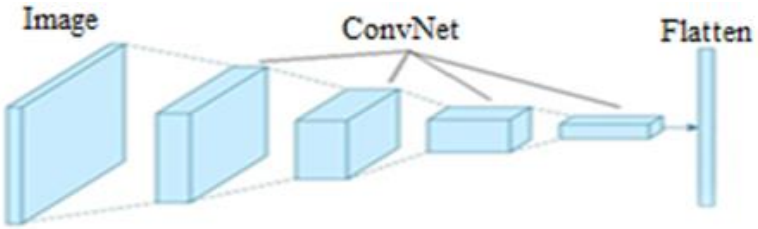
## 3.4 Tahap Ekstraksi Fitur Frame

Pada tahap ini, akan dilakukan ekstraksi fitur menggunakan MobileNetV2 untuk mengambil informasi yang berguna dari sebuah frame untuk pengenalan aktivitas. CNN adalah sumber fitur yang dominan dari representasi dan klasifikasi gambar. Melatih model deep learning untuk mendapatkan representasi gambar yang baik membutukan pelatihan pada ribuan bahkan jutaan gambar dan

juga membutuhkan GPU dengan kekuatan tinggi untuk melatih CNN. Masalah ini dapat diselesaikan dengan menggunakan model yang sudah dilatih terlebih dahulu [36].




Gambar 3.3 Diagram alir tahap praproses




Gambar 3.4 Ilustrasi ekstraksi fitur

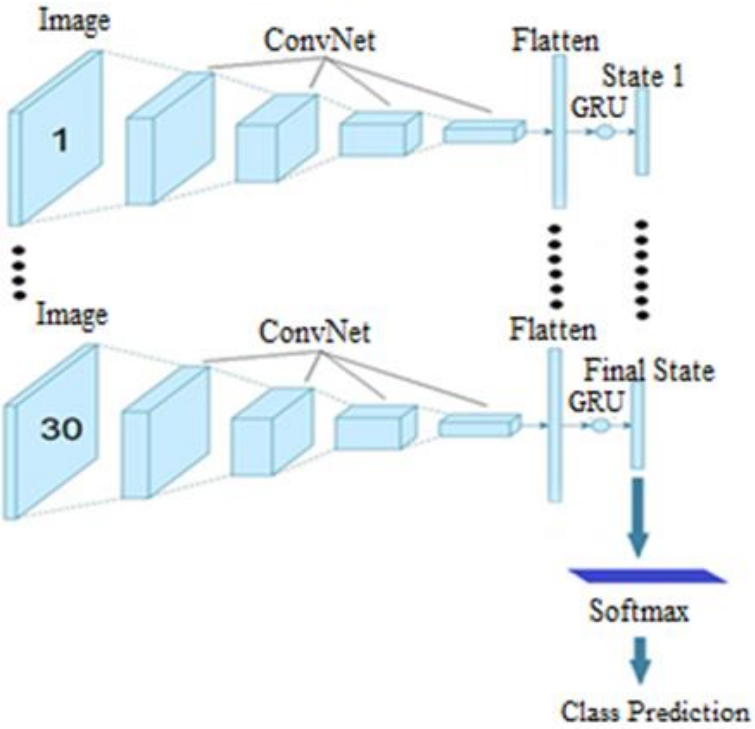
MobileNetV2 sudah dilatih pada dataset ImageNet agar dapat mengekstrak fitur dengan baik tanpa harus dilatih lebih lanjut pada dataset yang besar. Jumlah fitur yang dihasilkan dari MobileNetV2 adalah konvolusi yang dijadikan vector sepanjang 62,720 elemen. Jika tanpa CNN, maka model tidak bisa mengenali

apapun yang ada di dalam gambar. Ilustrasi ekstraksi fitur dapat dilihat pada Gambar 3.4.

## 3.5 Tahap Pengenalan Aktivitas

Pada tahap pengenalan aktivitas, fitur dari frame yang telah diekstrak oleh MobileNetV2 akan dimasukkan ke GRU. Tahap ini bertujuan untuk menganalisa pola perubahan antar frame. Setiap fitur dari frame akan digunakan untuk mengupdate state dari GRU, kemudian state terakhir akan digunakan untuk menganalisa hasil akhir dari aktivitas pada video. Jika tidak menggunakan GRU, model akan kehilangan fitur *spatiotemporal* sehingga model tidak dapat menganalisa perubahan frame. Ilustrasi tahap ini dapat dilihat pada Gambar 3.5.

Tahap ini menggunakan dua layer GRU dengan masing-masing layer terdapat 500 hidden unit. Dua layer digunakan agar model dapat menganalisa fitur dengan level lebih tinggi [11]. State dari GRU akan menjadi input dari Multi Layer Perceptron (MLP) dengan jumlah unit sebanyak jumlah kelas yaitu sebelas. MLP menggunakan fungsi aktivasi softmax dan perhitungan error menggunakan categorical cross entropy.

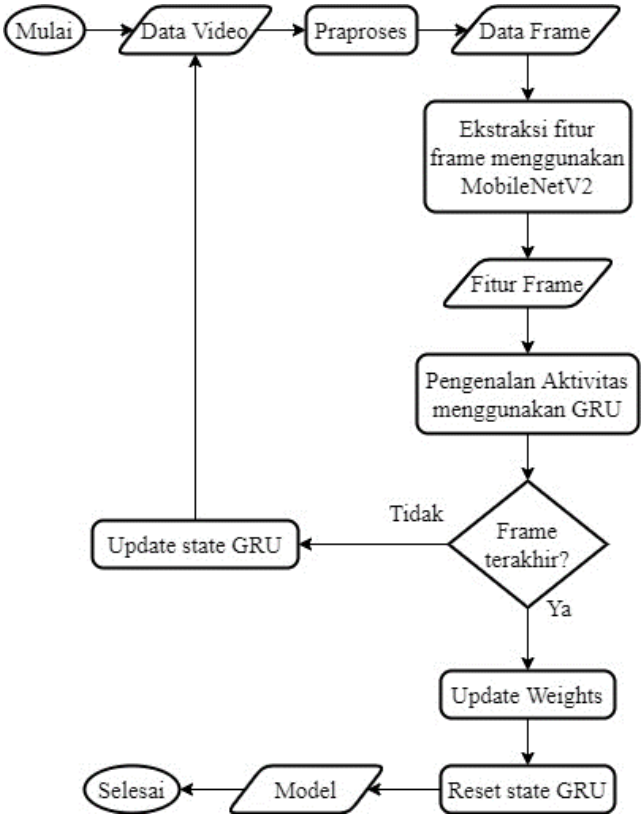



Gambar 3.5 Ilustrasi pengenalan aktivitas

## 3.6 Pelatihan dan Pengujian

Proses pelatihan dimulai dengan sistem menerima input video kemudian dilakukan praproses pada video tersebut sehingga didapatkan frame yang akan diproses lebih lanjut. Fitur pada frame tersebut diekstrak menggunakan MobileNetV2. Hasil keluaran dari MobileNetV2 akan dimasukkan ke GRU dengan tujuan menganalisa fitur *spatiotemporal* pada rangkaian frame. Proses ini diulang lagi sampai frame terakhir dari video. Proses update weight dilakukan setelah frame terakhir diproses dan didapatkan hasil outputnya. Kemudian state GRU direset agar analisa video selanjutnya tidak dipengaruhi oleh video sebelumnya. Setiap satu epoch dilakukan validasi menggunakan data validasi untuk

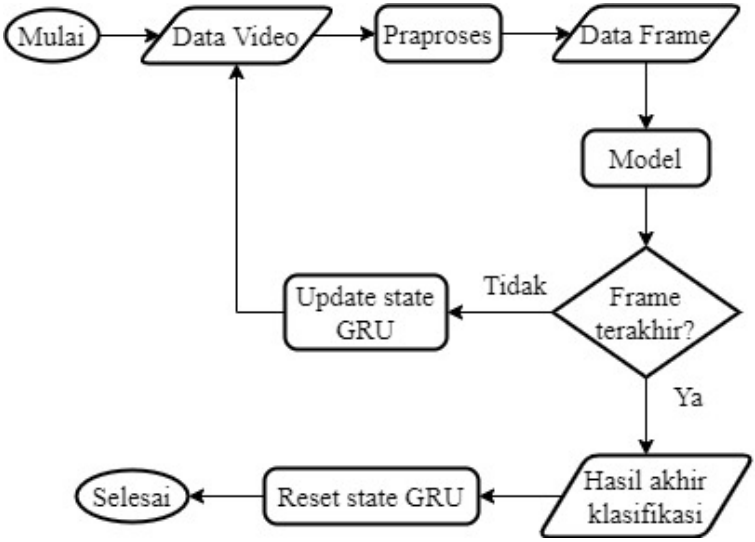
mengukur performa model saat pelatihan. Ilustrasi pelatihan dapat dilihat pada Gambar 3.6.




Gambar 3.6 Diagram alir pelatihan model.

Pengujian dimulai dengan sistem menerima input video kemudian dilakukan praproses melakukan praproses pada data video input. cara mengukur performa model pada data sehingga didapatkan frame yang akan diproses lebih lanjut. Frame akan dianalisa menggunakan model yang sudah didapatkan melalui proses pelatihan. Setelah mendapatkan output dari pengujian, dilakukan pengukuran performa dengan metrik akurasi, precision,

recall, dan f1 measure. Diagram alir proses pengujian dapat dilihat pada Gambar 3.7.




Gambar 3.7 Diagram alir pengujian model.

# BAB IV
# IMPLEMENTASI

Bab ini menjelaskan mengenai implementasi perangkat lunak dari rancangan sistem yang telah dibahas pada Bab 3 meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

## 4.1 Lingkungan Implementasi

Dalam mengimplementasikan aplikasi pengenalan ekspresi manusia diperlukan beberapa perangkat pendukung sebagai berikut.

- Perangkat Keras

Implementasi penelitian ini menggunakan Google Colaboratory. Sistem operasi yang digunakan adalah Ubuntu 18.0.4. PC yang digunakan memiliki spesifikasi Intel Xeon dengan kecepatan 2.3 GHz single core, 2 thread per core, *Random Access Memory* (RAM) sebesar 13 GB, dan mempunyai *Graphics Processing Unit* (GPU) yaitu NVIDIA Tesla K80 sebesar 12 GB.

- Perangkat Lunak

Perangkat lunak memiliki spesifikasi antara lain menggunakan bahasa pemograman Python 3.6.9, dilengkapi dengan *library* antara lain OpenCV, Tensorflow-GPU, Keras-GPU, Numpy, dan Matplotlib

## 4.2 Implementasi Praproses

Tahap praproses ditulis pada Kode Sumber 4.1. Parameter X pada baris pertama adalah path dari suatu video. Baris kedua adalah membuka video menggunakan library OpenCV. Baris ke tujuh adalah pembacaan frame dari suatu video. Baris ke delapan adalah pengecekan kondisi apakah video sudah habis atau belum. Baris ke sembilan adalah pengecekan apakah frame tersebut kelipatan enam

atau bukan. Jika iya maka baris ke sepuluh dijalankan. Baris ke sepuluh meresize frame menjadi ukuran 224*224 dan membagi nilai RGB dengan 255 kemudian disimpan pada suatu list. Baris ke dua belas mengubah list menjadi numpy array kemudian mereturn rangkaian model untuk dianalisa.

```
1.  def preprocess(X):
2.      cap = VideoCapture(X) # load video
3.      frameIndex = 0
4.      j = 0
5.      frames = []
6.      while True:
7.          ret, frame = cap.read() # mengambil frame
8.          if ret is False: break # jika video selesai
9.          if frameIndex % 6 == 0:
10.             frames.append(resize(frame/255, (224,
    224))) # jika kelipatan 6 maka resize ke bentuk
    224*224 dan membagi nilai rgb dengan 255
11.         frameIndex += 1
12.     return frames = np.array(frames) # cast ke
    numpy array
```

Kode Sumber 4.1 Implementasi praproses

## 4.3 Implementasi Pembangunan Arsitektur

Pada bagian ini akan dijabarkan implementasi fungsi-fungsi pada tahap pembangunan model. Implementasi arsitektur tercantum pada Kode Sumber 4.2. Baris pertama adalah kode untuk memulai membangun graf komputasi. Baris kedua adalah kode untuk memuat arsitektur MobileNetV2 beserta weightnya yang sudah ditrain pada ImageNet. Baris ketiga adalah kode untuk menambahkan MobileNetV2 ke graf komputasi dengan input berupa data timeseries. Input yang dimasukkan memiliki bentuk (timesteps, tinggi_gambar, lebar_gambar, kanal_warna) dengan batch 1. Karena video memiliki durasi yang berbeda, sehingga ukuran batch harus 1. None pada timesteps menunjukan jumlah timestep yang dinamis. Baris ke empat, tujuh, dan sembilan adalah dropout layer dengan batas 0.1. Baris ke lima adalah untuk

mengonversi fitur dari CNN yang berbentuk konvolusi menjadi vector. Baris ke 6 dan 7 adalah layer GRU dengan aktivasi relu. Jika stateful pada layer GRU adalah True, maka state pada layer tersebut tidak direset secara otomatis. Hal ini berguna untuk membagi data sekuensial menjadi rangkaian yang lebih kecil. Baris ke sepuluh adalah layer output yang berupa Dense dengan aktivasi softmax. Baris ke sebelas adalah untuk mengatur apakah layer tersbut diupdate weightnya saat pelatihan atau tidak. Bagian ini diatur sesuai skenario yang dijalankan. Baris ke 12 adalah optimizer yang digunakan. Baris ke 13 adalah untuk mengompile model agar dapat ditraining.

```
1.  model = Sequential()
2.  cnn = MobileNetV2(include_top=False,
    weights='imagenet') # load mobilenetv2
3.  model.add(TimeDistributed(cnn, input_shape = (None,
    224, 224,3), batch_size=1)) # menambahkan
    mobilenetv2 sebagai layer pertama pada model
4.  model.add(TimeDistributed(Dropout(0.1))) #
    menambahkan layer dropout
5.  model.add(TimeDistributed(Flatten())) # mengubah
    fitur menjadi sebuah vektor
6.  model.add(GRU(500, activation='relu',
    return_sequences=True, stateful=True)) # menambah
    layer gru
7.  model.add(Dropout(0.1)) # menambah layer dropout
8.  model.add(GRU(500, activation='relu',
    stateful=True)) # menambah layer gru
9.  model.add(Dropout(0.1)) # menambah layer dropout
10. model.add(Dense(11, activation='softmax')) #
    menambah klasifier menggunakan softmax
11. model.layers[0].trainable = False # membekukan
    bobot pada layer pertama
12. opt =  RMSprop(lr=lr) # menggunakan optimizer
    rmsprop
13. model.compile(optimizer=opt,
    loss='categorical_crossentropy',
    metrics=['accuracy']) # mengompile model
```

Kode Sumber 4.2 Implementasi arsitektur sistem

## 4.4 Implementasi Pelatihan

Setelah pembangunan arsitektur, dilakukan pembuatan fungsi pelatihan dan tes. Implementasi fungsi pelatihan dan tes dapat dilihat pada Kode Sumber 4.3. Parameter frame, Y, _model, dan train pada baris pertama adalah rangkaian frame hasil output dari fungsi preprocess(X), label kelas video, graf komputasi, dan boolean untuk menandai apakah training atau tidak. Baris ke lima dan delapan adalah mengubah bentuk frame agar sesuai dengan ukuran timestep dan batch yang sudah ditentukan. Pada kasus ini digunakan satu timestep dan satu batch. Baris ke enam adalah untuk mengupdate state GRU sampai frame sebelum terakhir. Baris ke sembilan adalah mengubah bentuk label agar sesuai dengan ukuran batch. Baris ke sepuluh adalah mengupdate weight berdasarkan state terakhir. Baris ke sepuluh dijalankan jika ingin melakukan training. Baris ke sebelas melakukan prediksi berdasarkan state terakhir. Baris ke sebelas dijalankan jika tidak ingin melakukan training. Baris ke dua belas adalah mereset state dari GRU agar video saat ini tidak dipertimbangkan pada video selanjutnya. Baris ke tiga belas adalah mereturn hasil dari training atau testing.

```
1.  def run(frames, Y, _model, train):
2.      l = len(frames) # mengambil panjang frame
3.      j = 0
4.      while j<l-1:
5.          x    =    np.reshape(frames[j],    (1,    1,
    frames[j].shape[0],              frames[j].shape[1],
    frames[j].shape[2])) # reshape data
6.          model.predict_on_batch(x) # update state gru
7.          j+=1
8.      x     =     np.reshape(frames[j],     (1,     1,
    frames[j].shape[0],              frames[j].shape[1],
    frames[j].shape[2])) # reshape data
9.      y  =  np.reshape(Y,  (1,  Y.shape[0]))  #  reshape
    label
10.     if train: ret = _model.train_on_batch(x=x, y=y)
    # update weight
```

```
11.     else:  ret  =  _model.test_on_batch(x=x,  y=y),
    np.argmax(_model.predict_on_batch(x)) # prediksi
12.     model.reset_states() # reset state gru
13.     return ret
```

Kode Sumber 4.3 Implementasi fungsi pelatihan dan tes

Setelah membuat fungsi untuk pelatihan dan tes, dilakukan proses pelatihan dan pembuatan model. Implementasi proses pelatihan dapat dilihat pada Kode Sumber 4.4. Baris ke tujuh adalah batas jumlah epoch yang sudah didefinisikan. Baris ke delapan dan sebelas adalah mendapatkan angka acak untuk mengubah urutan dataset. Baris ke sembilan, sepuluh, dua belas, dan tiga belas adalah mengubah urutan dataset berdasarkan angka acak yang sudah didapatkan. Baris ke tujuh belas adalah batas perulangan satu epoch, yaitu sebanyak jumlah data train karena ukuran batch hanya satu. Baris ke delapan belas adalah menjalankan fungsi run dengan parameter (rangkaian_frame, label, model, train). Train diatur True karena proses ini adalah proses pelatihan sehingga dilakukan update weight. Baris ke sembilan belas dan dua puluh adalan untuk menyimpan sementara nilai loss dan akurasi dari hasil training. Baris ke 24 adalah menyiapkan kontainer untuk menyimpan confusion matarix pada suatu epoch. Baris ke 26 adalah batas perulangan validasi. Baris ke 27 adalah proses menjalankan prediksi dengan data validasi. Parameter train pada baris ke 27 diatur False agar weight tidak dipengaruhi oleh data validasi. Baris ke 28 sampai tiga puluh adalah proses menyimpan sementara nilai loss, akurasi, dan confusion matrix pada validasi.

```
1.  trainLosses = []
2.  trainAcc = []
3.  valLosses = []
4.  valAcc = []
5.  confusionMatricesVal = []
6.  e = 0
7.  while e<epoch:
```

```
8.      r = np.array(sample(range(len(trainX)),
    len(trainX))) # mendapatkan angka acak
9.      trainX = trainX[r] # mengubah urutan data
    dengan angka acak yang sudah didapatkan
10.     trainY = trainY[r] # mengubah urutan label
    dengan angka acak yang sudah didapatkan
11.     r = np.array(sample(range(len(valX)),
    len(valX)))
12.     valX = valX[r]
13.     valY = valY[r]
14.     i = 0
15.     l = len(trainX)
16.     # train
17.     while i<l:
18.         ret = run(preprocess(trainX[i]), trainY[i],
    model, True) # menjalankan proses training
19.         trainLosses.append(ret[0]) # menyimpan loss
    training
20.         trainAcc.append(ret[1]) # menyimpan akurasi
    training
21.         i+=1
22.     i = 0
23.     l = len(valX)
24.     confusionMatricesVal.append(np.zeros((numCLass,
    numCLass), dtype=np.int32))
25.     # validation
26.     while i<l:
27.         ret = run(preprocess(valX[i]), valY[i],
    model, False) # menjalankan proses validasi
28.         valLosses.append(ret[0][0]) # menyimpan
    loss validasi
29.         valAcc.append(ret[0][1]) # menyimpan
    akurasi validasi
30.         confusionMatricesVal[e][np.argmax(valY[i]),
    ret[1]] += 1 # membuat confusion matrix validasi
31.         i+=1
```

Kode Sumber 4.4 Implementasi proses pelatihan

## 4.5 Implementasi Pengujian

Pada tahap pengujian, data yang digunakan sebagai data uji adalah data tes yang telah dibagi sebelumnya. Sebelum diuji, setiap video akan melalui praproses yang telah dijelaskan pada tahap sebelumnya terlebih dahulu. Setelah itu, dilakukan pengujian yang implementasinya dapat dilihat pada Kode Sumber 4.5. Proses pengujian secara umum sama dengan proses validasi. Pada baris ke tiga belas menghitung f1 score, presisi, dan recall berdasarkan confusion matrix hasil tes. Perhitungan metrik dapat dilihat pada Kode Sumber 4.6.

```
1.  i = 0
2.  l = len(testX)
3.  testLosses = []
4.  testAcc = []
5.  confusionMatrixTest = np.zeros((11, 11),
    dtype=np.int32)
6.  while i<l:
7.      ret = run(preprocess(testX[i]), testY[i],
    model, False) # menjalankan proses pengujian
8.      testLosses.append(ret[0][0]) # menyimpan loss
    pengujian
9.      testAcc.append(ret[0][1]) # menyimpan akurasi
    pengujian
10.     confusionMatrixTest[label[-1], ret[1]] += 1 #
    membuat confusion matrix pengujian
11.     i += 1
12. acc = np.sum(testAcc)/len(testX) # menghitung
    akurasi pengujian
13. macroF1, precision, recall =
    metric(confusionMatrixTest) # menghitung metriks
```

Kode Sumber 4.5 Implementasi proses pengujian

```
1.  def metric(cm):
2.      precision = []
3.      recall = []
4.      f1score = []
5.      for i in range(len(cm)):
```

```
6.          precision.append(cm[i, i]/np.sum(cm[:, i]))
    # true positive / (true positive + false positive)
7.          recall.append(cm[i, i]/np.sum(cm[i, :])) #
    true positive / (true positive + false negative)
8.          f1score.append(2*precision[-1]*recall[-
    1]/(precision[-1]+recall[-1])) # 2 * precision *
    recall / (precision / recall)
9.      return np.mean(f1score), np.mean(precision),
    np.mean(recall) # menghitung rata rata
```

Kode Sumber 4.6 Implementasi perhitungan metrik

# BAB V
# UJI COBA DAN EVALUASI

Bab ini akan membahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

## 5.1 Lingkungan Uji Coba

Lingkungan uji coba pada penelitian ini adalah Google Colaboratory. Sistem operasi yang digunakan adalah Ubuntu 18.0.4 dengan spesifikasi perangkat keras Intel Xeon dengan kecepatan 2.3 GHz single core, 2 thread per core, *Random Access Memory* (RAM) sebesar 13 GB. Pada sisi perangkat lunak, uji coba pada penelitian ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, OpenCV, Numpy, Matplotlib.

## 5.2 Deskripsi Dataset

Dataset dibagi menjadi 3 bagian yaitu data training, data validasi, dan data tes dengan jumlah 968, 316, dan 316 video. Spesifikasi lengkap dataset skenario dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi dataset Youtube 11 Action.

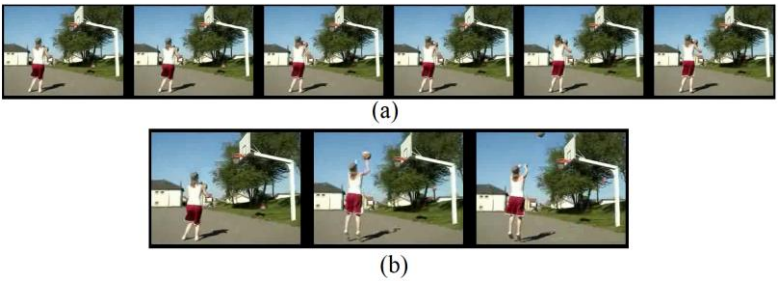
| **Keterangan** | **Spesifikasi** |
|---|---|
| Tinggi gambar asli | 144 – 262 px |
| Lebar gambar asli | 176 – 320 px |
| Frame rate | 9 – 60 fps |
| Jumlah frame | 11 – 536 |
| Durasi | 0.44 – 30 detik |
| Ekstensi | .avi |
| Jumlah video | 1600 |
| Jumlah kelas | 11 kelas |
| Kanal warna | 3 (RGB) |

## 5.3 Hasil Praproses

Sebelum memasuki proses pengenalan aktivitas yang telah dirancang sebelumnya, akan dilakukan praproses.

- Praproses Pemilihan Frame

Pada tahap praproses untuk pemilihan frame, suatu video akan diambil frame kelipatan enam untuk mengurangi fitur yang redundan dan menunurkan kompleksitas. Citra hasil dapat dilihat pada Gambar 5.1.




Gambar 5.1 (a) Rangkaian frame asli; (b) Rangkaian frame kelipatan enam.

- Praproses Resize

Pada tahap praproses untuk resize, gambar akan dijadikan ukuran $224 \times 224$ *pixel* sesuai dengan ukuran input MobileNetV2. Citra hasil dapat dilihat pada Gambar 5.2.

(a) (b)

Gambar 5.2 (a) Frame sebelum diresize; (b) Hasil frame setelah diresize.

## 5.4 Skenario Uji Coba

Proses uji coba berguna untuk menemukan parameter-parameter yang menghasilkan performa model yang paling optimal. Pelatihan dilakukan sebanyak 50 epoch dan setiap skenario diulang tiga kali untuk diambil rata ratanya. Parameter yang tepat akan memberikan hasil yang lebih baik pada saat proses uji coba.

Hasil terbaik dari suatu skenario validasi akan digunakan untuk skenario uji coba. Percobaan dilakukan tanpa menggunakan GPU. Pada subbab ini, skenario uji coba dibagi menjadi 3 yaitu:

1. Skenario Uji Coba pada pembekuan weight MobileNetV2
2. Skenario Uji Coba pada dropout
3. Skenario Uji Coba pada learning rate
4. Skenario Uji Coba pada jumlah hidden unit GRU

Parameter default yaitu weight MobileNetV2 dibekukan, dropout 0,1, learning rate 0,00001, dan jumlah hidden unit GRU 500.

### 5.4.1 Skenario Pembekuan Weight MobileNetV2

Pengujian scenario ini bertujuan untuk mengetahui apakah fitur yang diekstrak langsung dari MobileNetV2 tanpa dilatih Kembali menghasilkan performa lebih baik dibandingkan dengan MobileNetV2 yang dilatih kembali dengan dataset yang ada. Hasil percobaan dapat dilihat pada Tabel 5.2.

Tabel 5.2 Hasil percobaan skenario 1.

| **Informasi** | **Nilai** | |
|---|---|---|
| MobileNetV2 Weights | Dibekukan | Tidak Dibekukan |
| Akurasi | 92,30% | 91,24% |
| F1Score | 92,01% | 90,84% |
| Presisi | 92,25% | 91,51% |
| Recall | 92,16% | 90,94% |
| FPS | 65,43 | 69,21 |

Skenario ini menunjukkan bahwa weight MobileNetV2 yang dibekukan memiliki hasil F1Score yang lebih baik sehingga skenario selanjutnya menggunakan weight MobileNetV2 yang dibekukan.

### 5.4.2 Skenario Dropout

Skenario ini dilakukan untuk melihat apakah dropout dapat mengatasi overfitting pada model. Hasil percobaan dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil percobaan skenario 2.

| **Informasi** | **Nilai** | |
|---|---|---|
| Dropout | 0,1 | 0 |
| Akurasi | 92,30% | 90,82% |

| | | |
|---|---|---|
| F1Score | 92,01% | 90,36% |
| Presisi | 92,25% | 90,77% |
| Recall | 92,16% | 90,37% |
| FPS | 65,43 | 71,38 |

Skenario ini membuktikan bahwa menambahkan layer dropout dapat meningkatkan performa model. Skenario selanjutkan akan menggunakan layer dropout.

### 5.4.3 Skenario Learning Rate

Pada skenario ini dilakukan percobaan learning rate untuk mendapatkan hasil yang terbaik. Hasil percobaan dapat dilihat pada Tabel 5.4

Tabel 5.4 Hasil percobaan skenario 3.

| **Informasi** | **Nilai** | |
|---|---|---|
| Learning rate | 1E-05 | 1E-06 |
| Akurasi | 92,30% | 90,61% |
| F1Score | 92,01% | 90,17% |
| Presisi | 92,25% | 90,54% |
| Recall | 92,16% | 90,16% |
| FPS | 65,43 | 68,58 |

Perubahan learning rate antara 0.00001 menjadi 0.000001 dapat menurunkan performa model.

### 5.4.4 Skenario Jumlah Hidden Unit GRU

Pada skenario ini dilakukan percobaan untuk meningkatkan kecepatan model dengan cara mengurangi hidden layer pada GRU. Hasil percobaan dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil percobaan skenario 4.

| Informasi | Nilai | |
|---|---|---|
| Jumlah Hidden Unit GRU | 500 | 250 |
| Akurasi | 92,30% | 91,14% |
| F1Score | 92,01% | 90,85% |
| Presisi | 92,25% | 91,25% |
| Recall | 92,16% | 90,77% |
| FPS | 65,43 | 85,88 |

Jumlah hidden unit yang dikurangi dapat meningkatkan FPS tetapi menurunkan akurasi dan F1Score pada model.

## 5.5 Hasil dan Evaluasi

Pada uji coba penggantian parameter, diperoleh hasil F1 score yang paling baik pada MobileNetV2 yang dibekukan dengan dropout 0,1, learning rate 0,00001, dan jumlah hidden unit sebanyak 500 dengan akurasi sebesar 92,3% dan F1 score 92,01%. MobileNetV2 yang dibekukan memberikan hasil yang lebih baik karena banyaknya weight yang diupdate menyebabkan overfitting pada model. Dropout juga bekerja dengan baik pada kasus ini untuk mengurangi overfitting sehingga menghasilkan performa yang lebih baik. Confusion matrix pada dataset ini dapat dilihat pada Tabel 5.6.

Tabel 5.6 Confusion Matrix.

| Kelas | Basket | Berse peda | Lompat indah | Golf | Naik Kuda | Sepak bola | Ayun an | Tennis | Melo mpat | Volle y | Berjal an |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Basket | 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 6 | 0 |
| Bersepeda | 0 | 28 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Lompat indah | 0 | 0 | 28 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| Golf | 1 | 0 | 0 | 25 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Naik kuda | 0 | 0 | 0 | 0 | 38 | 0 | 0 | 0 | 0 | 0 | 1 |
| Sepak bola | 0 | 0 | 0 | 1 | 0 | 29 | 0 | 0 | 0 | 0 | 1 |
| Ayunan | 0 | 0 | 0 | 0 | 0 | 1 | 25 | 0 | 0 | 0 | 0 |
| Tennis | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 32 | 0 | 0 | 0 |
| Melompat | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 22 | 0 | 0 |
| Volley | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 22 | 0 |
| Berjalan | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 22 |

Confusion matrix menunjukkan bahwa kelas basket memiliki akurasi paling rendah. Basket sering diprediksi sebagai voli karena kedua kelas tersebut memiliki kemiripan yaitu sama-sama mengangkat tangan kemudian melontarkan bola. Selain itu background dari kedua kelas juga banyak yang sama-sama berada di lapangan dalam ruangan. Contoh kedua kelas tersebut dapat dilihat pada Gambar 5.3 (a) kelas basket dan 5.3 (b) kelas voli.

(a)

(b)

Gambar 5.3 (a) Ilustrasi kelas basket; (b) Ilustrasi kelas voli.

# BAB VI
# KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang didasari oleh hasil uji coba yang telah dilakukan pada bab sebelumnya. Kesimpulan nantinya sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut di masa depan.

## 6.1 Kesimpulan

Dalam pengerjaan penelitian ini setelah melalui tahap perancangan aplikasi, implementasi metode, serta uji coba, diperoleh kesimpulan sebagai berikut:

1. Pengenalan aktivitas dapat dilakukan dengan mengekstrak fitur dari sebuah frame menggunakan MobileNetV2 kemudian fitur tersebut dimasukkan ke GRU untuk menganalisa fitur *spatiotemporal* pada video tersebut.
2. Sistem yang memiliki performa paling baik adalah dengan membekukan weight MobileNetV2, dropout 0.1, learning rate 1e-5, dan 500 hidden unit GRU. Performa yang dihasilkan adalah F1 score 92.01%.
3. Berdasarkan uji coba tersebut, model yang dibangun menghasilkan rata-rata kecepatan 65 fps.
4. Mengurangi jumlah hidden layer pada GRU dapat menurunkan F1 Score sebesar 1.16% tetapi dapat menaikkan kecepatan hampir 20 fps.

## 6.2 Saran

Saran yang diberikan untuk pengembangan sistem pengenalan aktivitas manusia, yaitu:

1. Menambah variasi dataset berdasarkan kondisi sesungguhnya untuk menangani kondisi-kondisi pada realita.

2. Menambahkan deteksi manusia agar dapat mengenali aktivitas pada orang tertentu pada suatu video.
3. Melakukan eksplorasi arsitektur CNN selain MobileNetV2.
4. Melakukan eksplorasi dengan bottleneck layer agar model dapat menjadi lebih ringan dan diharapkan dapat dijalankan pada smart phone.

# DAFTAR PUSTAKA


[1] A. Nanda, D. S. Chauhan, S. P. K. dan S. Bakshi, “"Illumination and scale invariant relevant visual features with hypergraph-based learning for multi-shot person re-identification,” *Multimedia Tools Appl*, pp. 1-26, 2017.

[2] K. Soomro, A. R. Zamir dan M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” 2012.

[3] S. Herath, M. Harandi dan F. Porikli, “Going deeper into action recognition: A survey,” *Image Vis. Comput.,* vol. 60, pp. 4-21, 2017.

[4] A. Nanda, P. K. Sa, S. K. Choudhury, S. Baksh dan B. Majhi, “A neuromorphic person re-identification framework for video surveillance,” *IEEE Access,* vol. 5, pp. 6471-6482, 2017.

[5] S. A. Aly, T. A. Alghamdi, M. Salim dan A. A. Gutub, “Data dissemination and collection algorithms for collaborative sensor devices using dynamic cluster heads,” *Trends Appl. Sci. Res.,* vol. 8, pp. 55-72, 2013.

[6] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, V. O., R. Monga dan T. G., “Beyond short snippets: Deep networks for video classification,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* pp. 4694-4702, 2015.

[7] C. Schuldt, I. Laptev dan B. Caputo, “Recognizing human actions: A local SVM approach,” *Proc. 17th Int. Conf. Pattern Recognit. (ICPR),* pp. 32-36, 2004.

[8] O. V. R. Murthy dan R. Goecke, “Ordered trajectories for human action recognition with large number of classes,” *Image Vis. Comput.,* vol. 42, pp. 22-34, 2015.

[9] Z. C. Lipton, J. Berkowitz dan C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv,* 2015.

[10 K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrin dan J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 28, pp. 2222-2232, 2017.

[11 A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad dan S. W. Baik, "Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features," *IEEE Access,* vol. 6, pp. 1155 - 1166, 2017.

[12 K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk dan Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv,* 2014.

[13 J. Chung, C. Gulcehr, K. Cho dan Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv,* 2014.

[14 M. Sandler, A. Howard, M. Zhu, A. Zhmoginov dan L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018.

[15 F. Angelini, Z. Fu, Y. Long, L. Shao dan S. Naqvi, "ActionXPose: A Novel 2D Multi-view Pose-based Algorithm for Real-time Human Action Recognition," 2008.

[16 R. Colloert dan J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," *Princeton,* 2008.

[17 Aphex34, "Convolutional Neural Network," Wikipedia, 16 12 2015. [Online]. Available: https://en.wikipedia.org/wiki/File:Typical_cnn.png. [Diakses 14 12 2019].

[18 S. Fadillah, "Penerapan Pengolahan Citra menggunakan Metode Deep Learning untuk Mendeteksi Kecacatan Permukaan Buah Manggis," Yogyakarta, 2017.

[19 "Deep learning for complete beginners: convolutional neural networks with keras," Cambridgespark, 20 March 2017. [Online]. Available: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html. [Diakses 29 November 2018].

[20 "An Intuitive Explanation of Convolutional Neural Networks," Ujjwalkarn, 11 August 2016. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. [Diakses 29 November 2018].

[21 I. W. Suartika, A. Y. Wijaya dan R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional pada Caltech 101," *JURNAL TEKNIK ITS*, vol. 5, 2016.

[22 S. Sena, "Pengenalan Deep Learning Neural Network," 28 October 207. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac. [Diakses 30 November 2018].

[23 G. Hinton, *Neural Networks for Machine Learning*.

[24 A. Budhiraja, "Dropout in (Deep) Machine Learning," [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. [Diakses 11 12 2018].

[25 A. Karpathy, "Convolutional Neural Networks for Visual Recognition," Stanford University, [Online]. Available: http://cs231n.github.io/. [Diakses 30 November 2018].

[26 A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto dan H. Adam, "MobileNets:

Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv,* 2017.

[27 M. Venkatachalam, "Recurrent Neural Networks," [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce. [Diakses 13 June 2019].

[28 S. Narkhede, "Understanding Confusion Matrix," [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Diakses 29 May 2019].

[29 "About Python," Python, [Online]. Available: https://www.python.org/about/. [Diakses 30 November 2018].

[30 "Keras: The Python Deep Learning library," Keras, [Online]. Available: https://keras.io/. [Diakses 30 November 2018].

[31 "TensorFlow," TensorFlow, [Online]. Available: https://www.tensorflow.org/. [Diakses 30 November 2018].

[32 "OpenCV," [Online]. Available: https://opencv.org/. [Diakses 30 November 2018].

[33 "NumPy," NumPy, [Online]. Available: http://www.numpy.org/. [Diakses 30 November 2018].

[34 "Matplotlib," Matplotlib, [Online]. Available: https://matplotlib.org/index.html. [Diakses 30 November 2018].

[35 UCF, "www.crcv.ucf.edu," University of Central Florida, 31 October 2011. [Online]. Available: https://www.crcv.ucf.edu/data/UCF_YouTube_Action.php. [Diakses 24 June 2020].

[36 A. S. Razavian, H. Azizpour, J. Sullivan dan S. Carlsson, "CNN features off-the-shelf: An astounding baseline for

recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, pp. 806-813, 2014.

[37 "Scikit-learn," Scikit-learn, [Online]. Available: http://scikit-learn.org/stable/index.html. [Diakses 30 November 2018].

[38 S. Ruder, "Ruder.io," 19 January 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/index.html#rmsprop. [Diakses 23 December 2018].

[39 N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 9(1), pp. 62-66, 1979.

[40 s.-i. d. team, scikit-image, [Online]. Available: https://scikit-image.org. [Diakses 8 11 2019].

[41 "Sci-Py.org," Sci-Py.org, [Online]. Available: https://www.scipy.org/about.html. [Diakses 8 11 2019].

[42 "Binary Image Analysis," dalam *Computer Vision*, 2000, pp. 63-75.

[43 "Elman Networks," Mnemosyne Studio, [Online]. Available: http://mnemstudio.org/neural-networks-elman.htm. [Diakses 8 11 2019].

[44 A. Winarto, "Variations of SSD—Understanding Deconvolutional Single-Shot Detectors," Medium, 25 8 2018. [Online]. Available: https://medium.com/@amadeusw6/variations-of-ssd-understanding-deconvolutional-single-shot-detectors-c0afb8686d03. [Diakses 8 11 2019].

[45 L. Zheng, X. He, B. Samali dan L. T. Yang, "An algorithm for accuracy enhancement," *Journal of Computer and System Sciences*, pp. 245-255, 2013.

[46 S. C. Pau, "SEGMENTATION-FREE LICENSE PLATE RECOGNITION USING DEEP LEARNING," *A project report submitted in partial fulfilment of the*

*requirements for the award of Bachelor of Science (Hons.) Software Engineering* , 2017.

[47 F. Oladeji, "Developing a License Plate Recognition System with Machine Learning in Python," [Online]. Available: https://blog.devcenter.co/developing-a-license-plate-recognition-system-with-machine-learning-in-python-787833569ccd. [Diakses 27 May 2019].

[48 W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu dan A. C. Berg, "SSD: Single Shot Multibox Detector," vol. 5, 2016.

[49 E. Forson, "Understanding SSD MultiBox—Real-Time Object Detection In Deep Learning," [Online]. Available: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab. [Diakses 12 June 2019].

[50 P. Baskara, Pengenalan Nomor Polisi Kendaraan pada Data Video menggunakan Convolutional Neural Network, Surabaya: Institut Teknologi Sepuluh Nopember, 2019.

[51 K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk dan Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder," vol. 3, 2014.

[52 J. Chung, C. Gulcehre, K. Cho dan Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014.

[53 J. Chung, C. Gulcehre, K. Cho dan Y. Bengio, "123," 123.

# LAMPIRAN

## L.1 Hasil Uji Coba Pertama

| MobileNet Weights | Freezed | Unfreezed | Freezed | Freezed | Freezed |
|---|---|---|---|---|---|
| Dropout | 0,1 | 0,1 | 0 | 0,1 | 0,1 |
| Learning Rate | 0,00001 | 0,00001 | 0,00001 | 0,000001 | 0,00001 |
| Jumlah GRU Hidden Units | 500 | 500 | 500 | 500 | 250 |
| Accuracy | 0,9209 | 0,9019 | 0,9051 | 0,9209 | 0,9177 |
| MacroF1 | 0,9164 | 0,8971 | 0,8997 | 0,9191 | 0,9144 |
| Precision | 0,9206 | 0,9045 | 0,9068 | 0,9242 | 0,9208 |
| Recall | 0,9192 | 0,8992 | 0,9012 | 0,918 | 0,9127 |
| FPS | 65,38 | 65,87 | 71,18 | 69,71 | 86,99 |

## L.2 Hasil Uji Coba Kedua

| MobileNet Weights | Freezed | Unfreezed | Freezed | Freezed | Freezed |
|---|---|---|---|---|---|
| Dropout | 0,1 | 0,1 | 0 | 0,1 | 0,1 |
| Learning Rate | 0,00001 | 0,00001 | 0,00001 | 0,000001 | 0,00001 |
| Jumlah GRU Hidden Units | 500 | 500 | 500 | 500 | 250 |
| Accuracy | 0,9209 | 0,9082 | 0,8987 | 0,9051 | 0,9146 |

| | | | | | |
|---|---|---|---|---|---|
| MacroF1 | 0,9187 | 0,9066 | 0,895 | 0,9001 | 0,9129 |
| Precision | 0,9202 | 0,9126 | 0,8984 | 0,9018 | 0,9157 |
| Recall | 0,918 | 0,9076 | 0,8936 | 0,9008 | 0,9127 |
| FPS | 69,12 | 71,1 | 71,41 | 67,22 | 85,37 |

## L.3 Hasil Uji Coba Ketiga

| MobileNet Weights | Freezed | Unfreezed | Freezed | Freezed | Freezed |
|---|---|---|---|---|---|
| Dropout | 0,1 | 0,1 | 0 | 0,1 | 0,1 |
| Learning Rate | 0,00001 | 0,000001 | 0,00001 | 0,000001 | 0,00001 |
| Jumlah GRU Hidden Units | 500 | 500 | 500 | 500 | 250 |
| Accuracy | 0,9272 | 0,9272 | 0,9209 | 0,8924 | 0,9019 |
| MacroF1 | 0,9253 | 0,9215 | 0,9161 | 0,8859 | 0,8982 |
| Precision | 0,9268 | 0,9283 | 0,918 | 0,8905 | 0,901 |
| Recall | 0,9275 | 0,9213 | 0,9165 | 0,8861 | 0,8977 |
| FPS | 64,79 | 70,65 | 71,54 | 68,81 | 85,29 |

# BIODATA PENULIS

Rasyid Fajar, Lahir di Yogyakarta, 14 Agustus 1998. Penulis menempuh pendidikan mulai dari TK Al-Amin (2002 – 2004), SD IT Masjid Syuhada Yogyakarta (2004 – 2010), SMP Muhammadiyah 3 Yogyakarta (2010 – 2013), SMA Muhammadiyah 1 Yogyakarta (2013 – 2016), dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Penulis aktif dalam UKM Robotika ITS. Penulis juga aktif dalam mengikuti lomba robot kapal nasional maupun internasional bersama Barunastra ITS Roboboat Team. Prestasi penulis Bersama Barunastra ITS adalah menjuarai International Roboboat Competition di Florida, USA pada tahun 2018 dan 2019. Komunikasi dengan penulis dapat melalui *email*: **rasyidfajar01@gmail.com**.