ULIL ALBAB : Jurnal Ilmiah Multidisiplin
Vol.2, No.2, Januari 2023

Pengembangan Pengenalan Aktivitas Manusia Secara Real Time Menggunakan Metode ConvolutionalNeural Network dan Deep Gated Recurrent Unit

Feren Ade Verilia[1], Rangga Firdaus[2], Hery Dian Septama[3]
[1,2,3]Universitas Lampung
E-mail: feren.ade5014@students.unila.ac.id[1], ranggafirdaus@fkip.unila.ac.id[2], Hery@eng.unila.ac.id[3]

Article History:
Received: 10 Januari 2023
Revised: 30 Januari 2023
Accepted: 31 Januari 2023

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

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 thesis, proposed a human action recognition method by processing the video data using convolutional neural network (CNN) and deep gated recurrent unit (GRU) network. First, features are extracted from frame every multiple of six in the videos to helps reduce the redundancy and complexity. Next, the sequential information among frame features is learnt using deep GRU network, where multiple layers are stacked together to increase its depth. The result of this study achieved 92.01% F1 Score on YouTube 11 Actions dataset. This method achieve 65.43 FPS.*

PENDAHULUAN

Pengenalan aktivitas pada rangkaian video adalah masalah yang menantang pada visi komputer karena kemiripan dari konten visual (Nanda, Chauhan, K., & Bakshi, 2017), perubahan sudut pandang untuk aktivitas yang sama, gerakan kamera terhadap pelaku aktivitas, skala dan pose dari *actor*, dan perbedaan perubahan pencahayaan (Soomro, Zamir, & Shah, 2012). 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 (Herath, Harandi, & Porikli, 2017). 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 (Nanda, Sa, Choudhury, Baksh, & Majhi, 2017), robotik, interaksi manusia-komputer (A. Aly, Alghamdi, Salim, & Gutub, 2013), analisa olahraga, permainan video untuk karakter pemain, dan manajemen video web (Ng, Hausknecht, Vijayanarasimhan, O., Monga, & G., 2015). 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* (Schuldt, Laptev, & Caputo, 2004), dan *trajectory* (Murthy & Goecke, 2015). 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 *independent* (Lipton, Berkowitz, & Elkan, 2015). 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 St-1 dan data baru xt. 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 (Greff, Srivastava, Koutník, Steunebrin, & Schmidhuber, 2017). 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 (Ullah, Ahmad, Muhammad, Sajjad, & Baik, 2017). LSTM memiliki komputasi yang cukup besar, sehingga diperkenalkan metode GRU oleh Cho et al 2014 (Cho, et al., 2014). 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 (Chung, Gulcehr, Cho, & Bengio, 2014).

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 *MobileNet-V2* yang sudah ditrainig pada *ImageNet* (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018). Arsitektur selanjutnya adalah GRU untuk mempelajari rangkaian informasi dari *frame* pada video. Untuk mempelajari fitur yang lebih kompleks, digunakan dua *layer* GRU. Pada penelitian ini ini, diusulkan metode pengenalan aktivitas manusia dengan memproses data video menggunakan *Convolutional Neural Network* (CNN) dan *Deep Gated Recurrent Unit* (GRU).

LANDASAN TEORI

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 (Angelini, Fu, Long, Shao, & Naqvi, 2008). 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

MobileNetV2

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 (Howard, et al., 2017). *MobileNetV2* adalah perkembangan dari *MobileNet* yang berhasil meningkatkan performa model pada beberapa masalah. Alasan menggunakan *MobileNetV2* selain memang *score* akurasinya cukup tinggi, juga yang menjadi keunggulan utamanya ialah jumlah *training parameters* yang kecil dibandingkan dengan arsitektur CNN lainnya, sehingga kebutuhan akan komputasinya lebih ringan. Selain itu *model size MobileNetV2* juga kecil, hanya 14MB saja, namun tentunya dengan performansi yang baik. Sehingga kedepannya jika model tersebut akan di *deploy* kedalam sebuah *real app*, misal aplikasi android ataupun aplikasi berbasis *website* akan ringan dan berukuran kecil. Arsitektur *MobileNetV2* berbasis pada struktur i*nverted 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 1.

Table 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	-	

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 (Venkatachalam). 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.

Gated Recurrent Unit

Gated Recurrent Unit (GRU) adalah metode pengembangan dari RNN yang bekerja sangat baik dalam mengatasi masalah sekuensial (Cho, et al., 2014). 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 adalah:

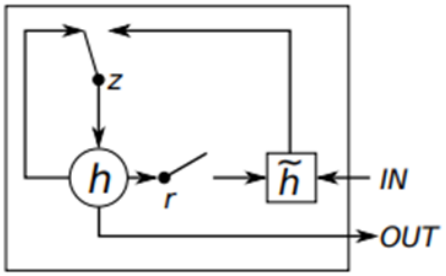
$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j \quad (2.1)$$

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \quad (2.2)$$

$$\tilde{h}_t^j = tanh(W x_t + U(r_t \odot h_{t-1}))^j \quad (2.3)$$

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \quad (2.4)$$

W adalah bobot, x_t adalah input pada unit, σ adalah fungsi sigmoid, dan Θ adalah Hadamard product (Chung, Gulcehr, Cho, & Bengio, 2014). Ilustrasi arsitektur GRU dapat dilihat pada gambar 1.



Gambar 1. Ilustrasi GRU. r dan z adalah reset gate dan update gate, dan h dan h˜ adalah aktivasi dan kandidat aktivasi

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. Contoh dataset dapat dilihat pada gambar 2.

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. Dataset dibagi menjadi 3 bagian yaitu data training, data validasi, dan data tes dengan jumlah 968, 316, dan 316 video. Spesifikasi dataset dapat dilihat pada table 1



Gambar 2. Contoh dataset yang digunakan (UCF, 2011)

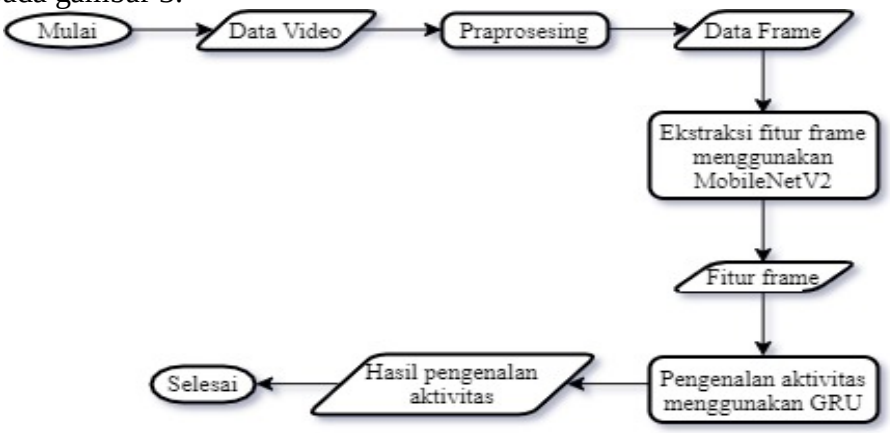
Tabel 2. Spesifikasi data latih uji coba SSD

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 (hlm 21)
Kanal warna	3 (RGB)

METODE PENELITIAN

Desain Umum Sistem

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.

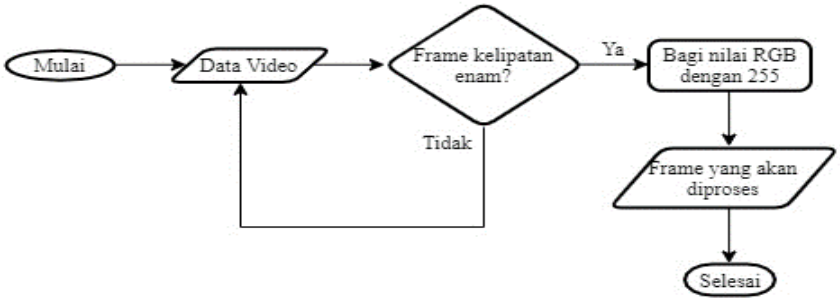


Gambar 3. Diagram alir sistem yang dibangun

Tahap Praproses

Tahap preprosesing dibagi menjadi tiga bagian yaitu memilih frame, resize, dan normalisasi. Sebuah video dengan frame rate yang tinggi memiliki banyak fitur yang redundan

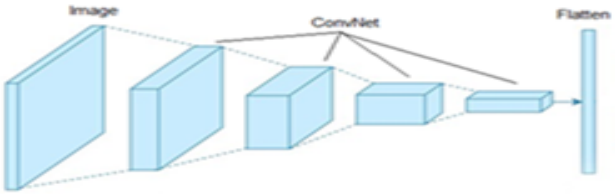
dan memiliki kompleksitas yang besar, maka dari itu tidak diproses seluruh frame pada video. Frame yang dipilih pada penelitian saya hanya frame yang kelipatan enam tujuannya untuk mengurangi komputasi dan mengurangi fitur-fitur yang redundan sehingga perubahan frame lebih signifikan hal ini didasari dari penelitian yang dilakukan oleh Ullah, Ahmad, Muhammad, Sajjad, dan Baik pada tahun 2018. Kemudian frame akan diresize menjadi ukuran 224x224 pixel hal ini dilakukan untuk menyesuaikan dengan input pada arsitektur CNN yaitu MobileNetV2. Setelah itu dilakukan normalisasi yaitu proses membuat beberapa variabel memiliki rentang nilai yang sama dengan cara membagi nilai RGB dengan 255 agar setiap nilainya di antara 0 dan 1. Diagram alir tahap praproses dapat dilihat pada gambar 4.



Gambar 4. Diagram alir sistem yang dibangun

Desain Tahap Ekstraksi Fitur Frame

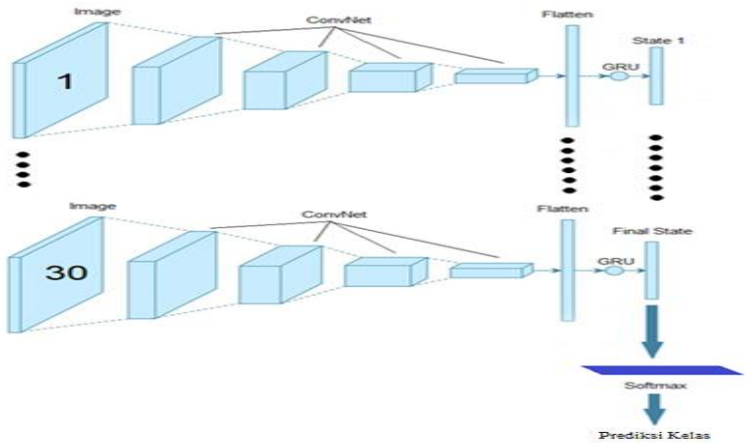
Pada tahap ini, akan dilakukan ekstraksi fitur menggunakan *MobileNetV2* untuk mengambil informasi yang berguna dari sebuah *frame* untuk pengenalan aktivitas. CNN adalah sumber fitur yang dominan dari representasi dan klasifikasi gambar. Melatih model *deep learning* untuk mendapatkan representasi gambar 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 (Razavian, Azizpour, Sullivan, & Carlsson, 2014). *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. Ilustrasi ekstraksi fitur dapat dilihat pada gambar 5.



Gambar 5. Ilustrasi ekstraksi fitur

Desain 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. Ilustrasi tahap ini dapat dilihat pada gambar 6.



Gambar 6. Ilustrasi pengenalan aktivitas

HASIL DAN PEMBAHASAN

Hasil

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 di*update* 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 table 2.

Tabel 3. Confusion Matrix

		Hasil Prediksi									
	Kelas	**Basket**	**Bersepeda**	**Lompat Indah**	**Golf**	**Naik Kuda**	**Sepak Bola**	**Ayunan**	**Tennis**	**Melompat**	**Volley**	**Berjalan**
Kelas Asli	**Basket**	26	0	0	0	0	1	0	0	0	1	0
	Bersepeda	0	28	0	0	0	0	0	0	0	0	1
	Lompat Indah	0	0	31	0	0	0	0	0	0	0	0
	Golf	0	0	0	27	0	1	0	0	0	0	0
	Naik Kuda	0	1	0	0	38	0	0	0	0	0	0
	Sepak Bola	0	0	0	1	1	28	0	1	0	0	0
	Ayunan	0	0	0	0	0	0	27	0	0	0	0
	Tennis	0	0	0	0	0	0	0	33	0	0	0
	Melompat	0	3	0	0	1	0	0	0	19	0	0
	Volley	0	0	1	0	1	0	0	0	0	21	0
	Berjalan	0	2	0	0	1	0	0	0	1	0	20

Tabel *Confusion matrix* terdiri dari kelas asli dan hasil prediksi. Kelas asli bersumber dari dataset YouTube 11 Action yang termasuk klasifikasi multiclass dan hasil prediksi yang diproleh dari data sistem yang telah diimplementasi pada bahasa pemrograman python yang dijalankan di Google Colaboratory. Setiap data dari masing-masing kelas dalam tabel *confusion matix* menunjukan jumlah prediksi yang dibuat guna untuk mengkalsifikasikan kelas yang benar atau salah. Misalkan pada kelas asli ayunan diperoleh 27 video dan menghasilkan 27 video pada kelas prediksi sehingga prediksinya dikatakan benar.

Pembahasan

Lingkungan uji coba pada tugas akhir 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 tugas akhir ini dilakukan dengan menggunakan bahasa pemograman Python 3.6 dilengkapi dengan *library* antara lain Keras, Tensorflow, OpenCV, Numpy, Matplotlib.

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 4 yaitu:

1. Skenario Uji Coba pada pembekuan *weight MobileNetV2* dan tidak pembekuan *weightMobileNetV2*

   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.

Table 4. Skenario uji coba pada pembekuan *weight MobileNetV2* dan tidak pembekuan *weight MobileNetV2*

MobileNetV2 Weights	Dropout	Learning Rate	Jumlah Hidden Unit GRU	Akurasi	F1Score	Presisi	*Recall*	FPS
Dibekukan	0.1	1.00E-5	500	92.30%	**92.01%**	92.25%	92.16%	65.43
Tidak Dibekukan	0.1	1.00E-5	500	91.24%	90.84%	91.51%	90.94%	69.21

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

2. Skenario Uji Coba pada *dropout* 0 dan *dropout* 0,1

Table 5. Skenario Uji Coba pada *dropout* 0 dan *dropout* 0,1

MobileNetV2 Weights	Dropout	Learning Rate	Jumlah Hidden Unit GRU	Akurasi	F1Score	Presisi	*Recall*	FPS
Dibekukan	**0.1**	1.00E-5	500	**92.30%**	**92.01%**	**92.25%**	**92.16%**	65.43
Dibekukan	0	1.00E-5	500	90.82%	90.36%	90.77%	90.37%	71.38

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

3. Skenario Uji Coba pada *learning rate* 1.00E-5 dan *learning rate* 1.00E-6

Table 6. Skenario Uji Coba pada *learning rate* 1.00E-5 dan *learning rate* 1.00E-6

MobileNetV2 Weights	Dropout	Learning Rate	Jumlah Hidden Unit GRU	Akurasi	F1Score	Presisi	*Recall*	FPS
Dibekukan	0.1	1.00E-5	500	**92.30%**	**92.01%**	**92.25%**	**92.16%**	65.43
Dibekukan	0.1	1.00E-6	500	90.61%	90.17%	90.54%	90.16%	68.58

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

4. Skenario Uji Coba pada jumlah *hidden* unit GRU 500 dan 250.

   Jumlah hidden unit yang dikurangi dapat meningkatkan FPS tetapi menurunkan akurasi dan F1*Score* pada model.

Table 7. Skenario Uji Coba pada jumlah *hidden* unit GRU 500 dan 250.

MobileNetV2 Weights	Dropout	Learning Rate	Jumlah Hidden Unit GRU	Akurasi	F1Score	Presisi	*Recall*	FPS
Dibekukan	0.1	1.00E-5	**500**	**92.30%**	**92.01%**	**92.25%**	**92.16%**	65.43
Dibekukan	0.1	1.00E-5	250	91.14%	90.85%	91.25%	90.77%	85.88

KESIMPULAN

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

1. 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%.
2. Berdasarkan 4 uji coba tersebut, model yang dibangun menghasilkan rata-rata kecepatan 65 FPS (Frames Per Second).
3. Mengurangi jumlah *hidden layer* 500 menjadi 250 pada GRU dapat menurunkan *F1 Score* sebesar 1.16% tetapi dapat menaikkan kecepatan sebesar 20.45 fps.

DAFTAR REFERENSI

A. Aly, S., Alghamdi, T. A., Salim, M., & Gutub, A. A. (2013). Data dissemination and collection algorithms for collaborative sensor devices using dynamic cluster heads. *Trends Appl. Sci. Res., 8*, 55-72.

About Python. (n.d.). (Python) Retrieved November 30, 2022, from https://www.python.org/about/

An Intuitive Explanation of Convolutional Neural Networks. (2016, August 11). (Ujjwalkarn) Retrieved November 29, 2022, from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets

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

Aphex34. (2015, 12 16). *Convolutional Neural Network*. (Wikipedia) Retrieved 11 14, 2022, from https://en.wikipedia.org/wiki/File:Typical_cnn.png

Budhiraja, A. (n.d.). *Dropout in (Deep) Machine Learning*. Retrieved 11 11, 2022, from https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Cho, K., Merrienboer, B. v., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv*.

Chung, J., Gulcehr, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*.

Colloert, R., & Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Princeton*.

Deep learning for complete beginners: convolutional neural networks with keras. (2017, March 20). (Cambridgespark) Retrieved November 29, 2022, from https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html

Fadillah, S. (2017). *Penerapan Pengolahan Citra menggunakan Metode Deep Learning untuk Mendeteksi Kecacatan Permukaan Buah Manggis.* Penelitian, Universitas Muhammadiyah Yogyakarta, Teknik Informatika, Yogyakarta.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrin, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst., 28*, 2222-2232.

Herath, S., Harandi, M., & Porikli, F. (2017). Going deeper into action recognition: A survey. *Image Vis. Comput., 60*, 4-21.

Hinton, G. (n.d.). Neural Networks for Machine Learning.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv*.

Karpathy, A. (n.d.). *Convolutional Neural Networks for Visual Recognition*. (Stanford University) Retrieved November 30, 2022, from http://cs231n.github.io/

Keras: The Python Deep Learning library. (n.d.). (Keras) Retrieved November 30, 2022, from https://keras.io/

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv*.

Matplotlib. (n.d.). (Matplotlib) Retrieved November 30, 2022, from https://matplotlib.org/index.html

Murthy, O. V., & Goecke, R. (2015). Ordered trajectories for human action recognition with large number of classes. *Image Vis. Comput., 42*, 22-34.

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

Nanda, A., Sa, P. K., Choudhury, S. K., Baksh, S., & Majhi, B. (2017). A neuromorphic person re-identification framework for video surveillance. *IEEE Access, 5*, 6471-6482.

Narkhede, S. (n.d.). *Understanding Confusion Matrix*. Retrieved May 29, 2022, from https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., O., V., Monga, R., & G., T. (2015). Beyond short snippets: Deep networks for video classification. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 4694-4702.

NumPy. (n.d.). (NumPy) Retrieved November 30, 2022, from http://www.numpy.org/

OpenCV. (n.d.). Retrieved November 30, 2022, from https://opencv.org/

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 806-813.

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

Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: A local SVM

approach. *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, 32-36.

Sena, S. (2017, October 28). *Pengenalan Deep Learning Neural Network*. Retrieved November 30, 2022, from https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac

Soomro, K., Zamir, A. R., & Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild.

Suartika, I. W., Wijaya, A. Y., & Soelaiman, R. (2016). Klasifikasi Citra Menggunakan Convolutional pada Caltech 101. *JURNAL TEKNIK ITS, 5*.

TensorFlow. (n.d.). (TensorFlow) Retrieved November 30, 2022, from https://www.tensorflow.org/

UCF. (2011, October 31). *www.crcv.ucf.edu*. (University of Central Florida) Retrieved June 24, 2021, from https://www.crcv.ucf.edu/data/UCF_YouTube_Action.php

Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features. *IEEE Access, 6*, 1155 - 1166.

Venkatachalam, M. (n.d.). *Recurrent Neural Networks*. Retrieved June 13, 2022, from https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce