Komputika: Jurnal Sistem Komputer
Volume 11, Nomor 1, April 2022, hlm. 19 - 28
Terakreditasi Peringkat 3, SK No. 28/E/KPT/2019
DOI: 10.34010/komputika.v11i1.5176

ISSN: 2252-9039 (print)
ISSN: 2655-3198 (online)

Pengenalan Gestur Angka Pada Tangan Menggunakan Arsitektur *AlexNet* Dan *LeNet* Pada Metode *Convolutional Neural Network*

Muhammad Ezar Al Rivan[1*], Alvin Setiawan[2]

[1,2)] Program Studi Informatika, Fakultas Ilmu Komputer dan Rekayasa, Universitas Multi Data Palembang
Jl. Rajawali No. 14, Palembang, Indonesia 30113
Email: meedzhar@mdp.ac.id

(Naskah masuk: *07 Juli 2021*; diterima untuk diterbitkan: *27 Agustus 2021*)

ABSTRAK *– Gestur merupakan salah satu jenis komunikasi dengan membentuk suatu objek seperti huruf atau angka pada tangan untuk menyampaikan pesan ataupun sebuah informasi, salah satunya gestur angka pada tangan yang memiliki banyak jenisnya dengan pola yang berbeda untuk setiap pergerakan angka yang terbentuk. Komputer memiliki kendala dalam mengenali gestur angka pada tangan karena perlu metode pengenalan. Salah satu solusi yang dapat dilakukan untuk melakukan pengenalan gestur angka pada tangan kepada komputer adalah menggunakan metode Convolutional Neural Network (CNN) dengan arsitektur AlexNet maupun LeNet. Penelitian ini menggunakan dataset citra gestur angka yang sebelumnya dilakukan tahap pre-processing yang terdiri dari threshold dan resize. Penelitian dilakukan menggunakan 2 pooling layer, yaitu Average Pooling dan Max Pooling kemudian menggunakan optimizer, yaitu SGD, RMSprop, dan Adam. Berdasarkan hasil pengujian yang didapatkan pada penelitian ini, yaitu penggunaan arsitektur AlexNet dengan Average Pooling dan optimizer RMSprop menghasilkan akuras dan f1-score keseluruhan 99,45% serta penggunaan arsitektur LeNet dengan Average Pooling dan optimizer RMSprop menghasilkan akurasi dan f1-score keseluruhan 99,49%. Secara keseluruhan penggunaan Average Pooling dengan optimizer RMSprop mendapatkan tingkat akurasi yang paling baik dibandingkan dengan pengujian yang lainnya.*

Kata Kunci *– Gestur; Convolutional Neurral Network; Optimizer; AlexNet; LeNet; Pooling Layer.*

Recognition Of Number Gesture On Hand Using Architecture AlexNet And LeNet On Convolutional Neural Network

ABSTRACT *– Gesture is one type of communication by forming an object such as letters or numbers on the hand to convey a message or information, one of which is the number gesture on the hand which has many types with a different patterns for each movement of the numbers formed. Computer have problems recognizing numeric hand gestures because they needs recognition method. One solution that can be done to perform numeric gesture recognition on a computer is to use the Convolutional Neural Network (CNN) method with AlexNet and LeNet architectures. This study uses a numeric gesture image dataset that was previously carried out in the pre-processing stage consisting of threshold and resize. The research was conducted using 2 pooling layers, namely Average Pooling and Max Pooling and then using an optimizer, namely SGD, RMSprop, and Adam. Based on the test results obtained in this study, the use of the AlexNet architecture with Average Pooling and the RMSprop optimizer resulted in an overall accuracy and f1-score of 99.45% and the use of the LeNet architecture with Average Pooling and the RMSprop optimizer resulted in an overall accuracy and f1-score of 99.49%. Overall the use of Average Pooling with the RMSprop optimizer gets the best level of accuracy compared to other tests.*

Keywords *– Gesture; Convolutional Neurral Network; Optimizer; AlexNet; LeNet; Pooling Layer.*

1. PENDAHULUAN

Gestur merupakan bentuk komunikasi non-verbal dengan aksi pada tubuh yang terlihat untuk menyampaikan pesan-pesan tertentu dan dapat digunakan sebagai pengganti komunikasi secara verbal. Gestur juga sering diartikan sebagai gerakan fisik yang mengikuti pergerakan dari jari-jari, tangan, lengan, wajah atau bagian tubuh yang lainnya [1]. Seiring berkembangnya zaman penggunaan gestur tidak hanya dilakukan untuk komunikasi dan pembelajaran dalam kehidupan sehari-hari, penggunaan gestur terutama gestur angka dapat langsung dimengerti oleh setiap manusia yang menggunakan gestur sebagai metode komunikasi atau pembelajaran. Pengenalan gestur memiliki tujuan untuk dapat mengenali makna dari ekspresi gerakan manusia, termasuk didalamnya tangan, lengan, wajah, kepala atau bagian tubuh yang lainnya [2].

Namun kondisinya sedikit berbeda, apabila gestur dikenalkan kepada komputer. Dalam hal ini komputer tidak dapat langsung mengenali sebuah gestur yang ditampilkan, komputer memerlukan sebuah kecerdasan buatan untuk mengenali suatu objek atau gambar. Gestur memiliki banyak jenisnya salah satunya adalah gestur angka. Hal ini menjadi salah satu kesulitan dalam mengenalkan sebuah gestur tersebut kepada komputer.

Beberapa penelitian terkait dengan pengenalan gestur huruf maupun angka yang telah dilakukan dengan berbagai macam metode seperti, penelitian [3] dengan hasil penelitian menunjukan bahwa metode CNN mencapai tingkat pengenal 6 angka sebesar 100%, lalu penelitian [4] dengan hasil pengujian yang memiliki akurasi sebesar 98,89%, dan penelitian [5] dengan menghasilkan tingkat pengenalan sebesar 99%. Kemudian penelitian [6] dengan menghasilkan tingkat akurasi pengenalan sebesar 100%, lalu penelitian [7] menghasilkan tingkat akurasi pengenalan sebesar 99%.

Dalam melakukan pengenalan gestur terdapat berbagai macam metode, salah satunya adalah Convolutional Neural Network (CNN). CNN merupakan klasifikasi objek citra yang banyak digunakan pada penelitian terdahulu dikarenakan menghasilkan tingkat akurasi yang signifikan dalam melakukan pengenalan objek citra [8]. Metode CNN merupakan metode *deep learning* yang mampu mengenali bentuk objek khususnya citra. CNN sering digunakan untuk mengatasi permasalahan seperti *object detection*, *image classification* dan *image segmentation*. Adapun penelitian terkait image *classification* dan *image segmentation* [9] menghasilkan tingkat akurasi sebesar 96,47, dan penelitian [10] yang menghasilkan tingkat akurasi sebesar 95%. Sehingga dapat disimpulkan bahwa pengenal gestur dapat dikenali dengan baik dengan menggunakan sebuah kecerdasan buatan dengan metode CNN.

2. METODE PENELITIAN

2.1. Identifikasi Masalah

Pada tahap ini dilakukan identifikasi masalah penelitian mengenai bagaimana melakukan pengenalan dalam mengklasifikasikan gestur angka 0 s.d. 5 pada tangan kanan dan tangan kiri berbentuk citra menggunakan arsitektur *AlexNet* dan *LeNet* pada metode CNN.

2.2. Studi Literatur

Tahapan ini melakukaan pencarian, pengumpulan dan mempelajari beberapa jurnal dan buku yang berkaitan dengan pengenalan gestur angka pada tangan dengan berbagai macam metode serta pengggunaan metode CNN dengan arsitektur *AlexNet* dan arsitektur *LeNet* dalam pengenalan atau klasifikasi citra.

2.3. Pengumpulan Data

Data yang digunakan dalam penelitian merupakan dataset *public* yang berjumlah 21.600 data gambar dengan 12 jenis gestur angka pada tangan kanan dan tangan kiri, yaitu gestur angka 0, 1, 2, 3, 4, 5. Dataset yang berjumlah 21.600 data gambar, untuk setiap dataset gambar citra berukuran asli 128x128. Dataset yang digunakan bersumber dari *website Kaggle* [11]

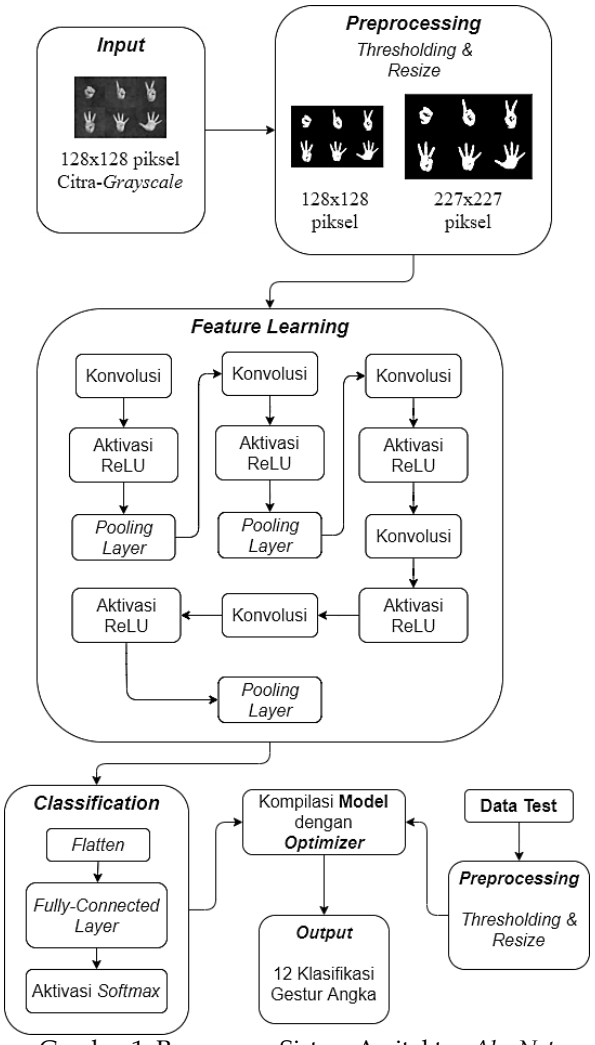
2.4. Perancangan Sistem

Tahap ini dataset yang digunakan akan dibagi menjadi dua fase pelatihan yaitu *feature learning* atau ekstraksi fitur dan *classfication*, dengan penggunaan data *training* sebanyak 75%, data *validation* sebanyak 15%, dan data *testing* sebanyak 10% dari total dataset yang digunakan sebanyak 21.600 data gambar. Proses pengenalan dalam mengklasifikasikan citra gestur angka pada tangan dibagi menjadi 2 tahapan besar yaitu, perancangan sistem dengan arsitektur *AlexNet* dan perancangan sistem dengan arsitektur *LeNet*. Tahapan perancangan sistem dengan arsitektur *AlexNet* dapat dilihat pada Gambar 1.

Pada Gambar 1 merupakan proses dari CNN dengan rancangan arsitektur *AlexNet*, proses tersebut memiliki spesifikasi tersendiri. Secara keseluruhan lapisan yang dimiliki arsitektur *AlexNet* telah diringkas untuk mempermudah proses penelitian yang dapat dilihat pada Tabel 1.

Tabel 1. Ringkasan Arsitektur *AlexNet*

Layer	*Feature Maps*	*Size*	*Kernel Size*	*Stride*	*Activation*
Input	3	227x227	-	-	-
Conv	16	55x55	11x11	4	ReLU
Pooling	16	27x27	3x3	2	-
Conv	32	23x23	5x5	1	ReLU
Pooling	32	11x11	3x3	2	-
Conv	64	9x9	3x3	1	ReLU
Conv	64	7x7	3x3	1	ReLU
Conv	32	5x5	3x3	1	ReLU
Pooling	32	2x2	3x3	2	-
FC	-	32	-	-	ReLU
FC	-	32	-	-	ReLU
FC-Output	-	12	-	-	*Softmax*



Gambar 1. Rancangan Sistem Arsitektur *AlexNet*

Hasil dimensi keluaran atau *size* dari masing-masing lapisan didapatkan dengan menggunakan Persamaan 1 dan 2 [12].

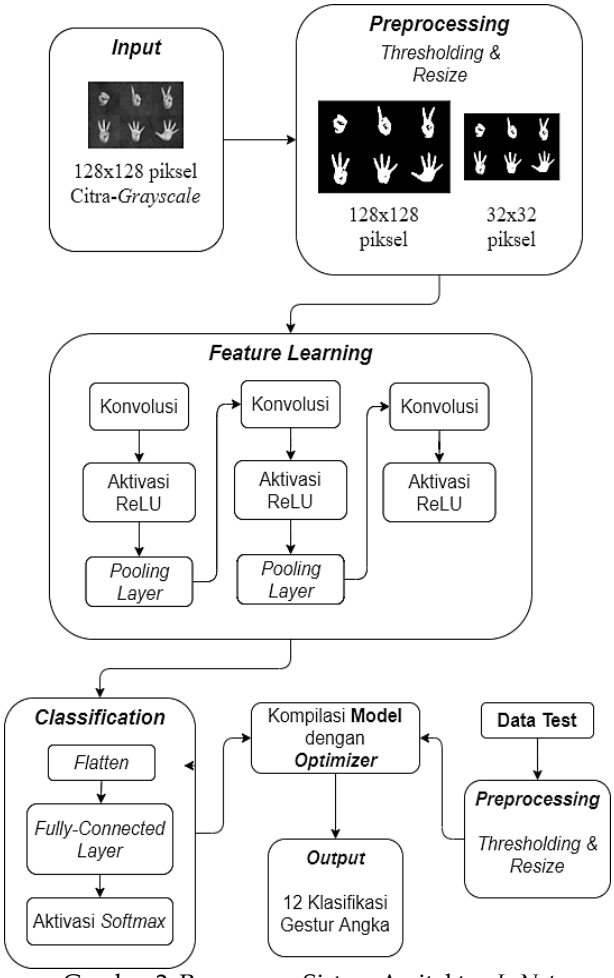
$$Size\ Weight = \left(\frac{W-F_w+2P}{S_w}\right)+1 \quad (1)$$

$$Size\ Height = \left(\frac{H-F_h+2P}{S_h}\right)+1 \quad (2)$$

Dimana $W\ (width)\ dan\ H(Height)$ adalah ukuran dari sebuah citra *input*, $F_w\ (width)\ dan\ F_h(Height)$ adalah filter atau *kernel*, P adalah *padding* suatu citra dengan ukuran *default* yang bernilai 0 (nol), dan $S_w\ (width)\ dan\ S_h(Height)$ adalah ukuran dari *stride* atau pergeseran suatu *kernel size* [12].

Tahapan perancangan sistem dengan arsitektur *LeNet* dapat dilihat pada Gambar 2. Pada Gambar 2 merupakan proses dari CNN dengan rancangan arsitektur *LeNet*, proses tersebut memiliki spesifikasi tersendiri.

Secara keseluruhan lapisan yang dimiliki arsitektur *LeNet* telah diringkas untuk mempermudah proses penelitian yang dapat dilihat pada Tabel 2.



Gambar 2. Rancangan Sistem Arsitektur *LeNet*

Tabel 2. Ringkasan Arsitektur *LeNet*

Layer	*Feature Maps*	*Size*	*Kernel Size*	*Stride*	*Activation*
Input	3	32x32	-	-	-
Conv	6	28x28	5x5	1	ReLU
Pooling	6	14x14	2x2	2	-
Conv	16	10x10	5x5	1	ReLU
Pooling	16	5x5	2x2	2	-
Conv	120	1x1	5x5	1	ReLU
FC	-	32	-	-	ReLU
FC-Output	-	12	-	-	*Softmax*

Hasil dimensi keluaran atau *size* dari masing-masing lapisan didapatkan dengan menggunakan Persamaan 1 dan 2 [12].

2.5. Implementasi

Tahap implementasi ini akan menerapkan rancangan CNN dengan arsitektur *AlexNet* dan *LeNet* terhadap citra gestur angka pada tangan. Tahap ini akan diimplementasikan kedalam bentuk bahasa pemrogramman yaitu *Matlab* dan *Python*. Untuk tahap preprocessing yaitu *threshold* menggunakan *tools Matlab* dilanjutkan dengan *preprocessing* kedua yaitu *resize* dan proses keseluruhan CNN dengan arsitektur *AlexNet* dan *LeNet* menggunakan *plugin Jupyter Notebook* pada *tools Visual Studio Code*.

2.6. Pengujian

Proses pengujian dilakukan dengan melakukan *tunning parameter* yang kemudian akan digunakan untuk melakukan prediksi terhadap data *testing* dengan tujuan untuk menghasilkan tingkat akurasi yang terbaik dalam mengklasifikasikan citra gestur angka pada tangan. Rincian skenario pengujian dapat dilihat pada Tabel 3.

Tabel 3. Rincian Skenario Pengujian

Skenario	Arsitektur	*Pooling Layer*	*Optimizer*
1	*AlexNet*	*Average Pooling*	*SGD*
2	*AlexNet*	*Average Pooling*	*RMSprop*
3	*AlexNet*	*Average Pooling*	*Adam*
4	*AlexNet*	*Max Pooling*	*SGD*
5	*AlexNet*	*Max Pooling*	*RMSprop*
6	*AlexNet*	*Max Pooling*	*Adam*
7	*LeNet*	*Average Pooling*	*SGD*
8	*LeNet*	*Average Pooling*	*RMSprop*
9	*LeNet*	*Average Pooling*	*Adam*
10	*LeNet*	*Max Pooling*	*SGD*
11	*LeNet*	*Max Pooling*	*RMSprop*
12	*LeNet*	*Max Pooling*	*Adam*

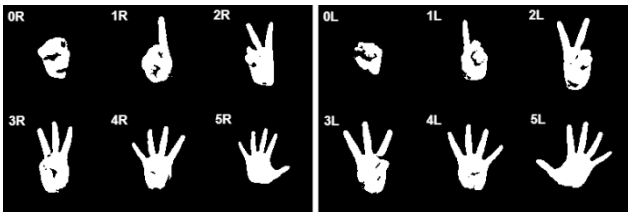
2.7. Evaluasi

Pada tahap ini dilakukan evaluasi dari tahapan sebelumnya yaitu pengujian. Evaluasi yang dilakukan untuk data per kelas sebanyak 12 kelas dengan menggunakan metode *Confusion Matrix* yang terdiri dari *Precision*, *Recall*, *Accuracy* dan *F1-Score*.

3. HASIL DAN PEMBAHASAN

3.1. Hasil *Thresholding* terhadap Citra Gestur Angka Pada Tangan

Pada tahap ini menjelaskan proses dari tahap *preprocessing* yaitu metode *thresholding* dengan mengubah citra *grayscale* menjadi citra hitam putih dengan tujuan mendapatkan bentuk dari citra yang dihasilkan. Citra hasil metode *thresholding* dapat dilihat pada Gambar 3, dengan jenis gestur angka yang terdiri dari 12 jenis kelas, yaitu 0R, 1R, 2R, 3R, 4R, 5R, 0L, 1L, 2L, 3L, 4L, 5L.



Gambar 3. Citra Hasil *Thresholding*

3.2. Implementasi Metode *Convolutional Neural Network* (CNN)

Ringkasan dari metode CNN dengan arsitektur *AlexNet* dan *LeNet* akan diimplementasikan untuk melakukan *training* model pada dataset sebanyak 25 kali atau 25 *epoch* dengan menggunakan 2 jenis *pooling layer*, 3 jenis *optimizer* dengan *learning rate* sebesar 1.0E-5 atau 0.00001. Kemudian proses penyimpanan model menggunakan sebuah *checkpoint* (*Checkpoint Callback*) untuk menyimpan model dengan tingkat akurasi yang lebih baik dari *epoch* sebelumnya.

Dataset yang digunakan terdiri dari *folder* dataset *training* dan *validation* dengan rasio dataset *trainining* sebanyak 75% dengan jumlah per kelas (1.350 gambar) dan daset *validation* sebanyak 15% dengan jumlah per kelas (270 gambar) dan *folder* dataset *testing* sebanyak 10% dengan jumlah per kelas (180 gambar) dari total dataset yang digunakan sebanyak 21.600 data gambar dengan total kelas data sebanyak 12 kelas.

3.3. Implementasi Metode CNN dengan Arsitektur *AlexNet*

Proses *training* dataset pada arsitektur *AlexNet* dapat dilihat pada Gambar 4.

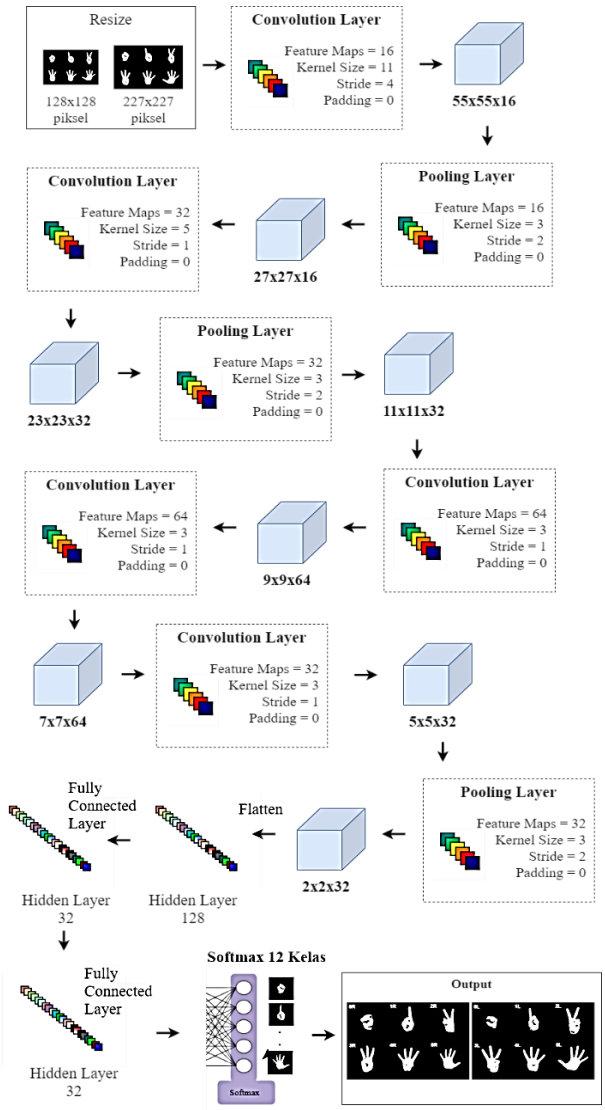
Params yang dihasilkan pada lapisan *convolution* (*Conv2D*) dan lapisan *fully-connected* (*Dense*) diperoleh dengan menggunakan Persamaan 3 dan 4 [13].

$$Conv = (N \times M \times L + bias) \times K \quad (3)$$

Dimana N adalah *kernel size width*, M adalah *kernel size height*, L adalah *feature maps input*, dan K adalah *feature maps output*.

$$Dense = (N \times bias) \times M \quad (4)$$

Dimana N merupakan *node input* dan M adalah *node output*. Nilai bias dalam kasus ini bernilai 1.



Gambar 4. Proses *Training* pada Arsitektur *AlexNet*

Ringkasan hasil model dari hasil implementasi arsitektur *AlexNet* dengan menggunakan *average* dan *max pooling* dapat dilihat pada Tabel 4 dan Tabel 5.

Output shape yang dihasilkan pada lapisan *convolution* dan *fully-connected* didapatkan menggunakan Persamaan 1 dan 2. Kemudian *output shape* pada lapisan *flatten* didapatkan dengan menggunakan Persamaan 5 [14].

$$flatten = Output_{witdh} \times Output_{height} \times f \quad (5)$$

Dimana f adalah *feature map* dan $Output$ yang dimaksud adalah hasil keluaran dari lapisan sebelum *flatten*. Pada lapisan *pooling layer* tidak menghasilkan nilai *param* dikarenakan lapisan ini hanya digunakan untuk memperkecil ukuran dimensi citra [15]. Semua tahap dari proses lapisan *convolution* dan lapisan *fully-connected* menggunakan aktivasi ReLU dengan tujuan menghasilkan *output* piksel tanpa adanya nilai negatif [16], dan lapisan akhir menggunakan aktivasi *softmax* untuk menghitung probabilitas dari masing-masing kelas.

Tabel 4. Ringkasan Model Hasil Implementasi Arsitektur *AlexNet* dengan *Average Pooling*

Layer (type)	***Output Shape***	***Param #***
conv2d (Conv2D)	*(None, 55, 55,16)*	5824
average_pooling2d (AveragePooling2D)	*(None, 27. 27, 16)*	0
conv2d_1 (Conv2D)	*(None, 23, 23, 32)*	12832
average_pooling2d_1 (AveragePooling2D)	*(None, 11, 11, 32)*	0
conv2d_2 (Conv2D)	*(None, 9, 9, 64)*	18496
conv2d_3 (Conv2D)	*(None, 7, 7, 64)*	36928
conv2d_4 (Conv2D)	*(None, 5, 5, 32)*	18464
average_pooling2d_2 (AveragePooling2D)	*(None, 2, 2, 32)*	0
flatten (Flatten)	*(None, 128)*	0
dense (Dense)	*(None, 32)*	4128
dense_1 (Dense)	*(None, 32)*	1056
dense_2 (Dense)	*(None, 12)*	396
Total Param : 98124		
Trainale params : 98124		
Non-trainable params : 0		

Tabel 5. Ringkasan Model Hasil Implementasi Arsitektur *AlexNet* dengan *Max Pooling*

Layer (type)	***Output Shape***	***Param #***
conv2d (Conv2D)	*(None, 55, 55,16)*	5824
average_pooling2d (AveragePooling2D)	*(None, 27. 27, 16)*	0
conv2d_1 (Conv2D)	*(None, 23, 23, 32)*	12832
average_pooling2d_1 (AveragePooling2D)	*(None, 11, 11, 32)*	0
conv2d_2 (Conv2D)	*(None, 9, 9, 64)*	18496
conv2d_3 (Conv2D)	*(None, 7, 7, 64)*	36928
conv2d_4 (Conv2D)	*(None, 5, 5, 32)*	18464
average_pooling2d_2 (AveragePooling2D)	*(None, 2, 2, 32)*	0
flatten (Flatten)	*(None, 128)*	0
dense (Dense)	*(None, 32)*	4128
dense_1 (Dense)	*(None, 32)*	1056
dense_2 (Dense)	*(None, 12)*	396
Total Param : 98124		
Trainale params : 98124		
Non-trainable params : 0		

3.4. Implementasi Metode CNN dengan Arsitektur *LeNet*

Ringkasan hasil model dari hasil implementasi arsitektur *AlexNet* dengan menggunakan *average pooling* dan *max pooling* dapat dilihat pada Tabel 6 dan Tabel 7.

Tabel 6. Ringkasan Model Hasil Implementasi Arsitektur *LeNet* dengan *Average Pooling*

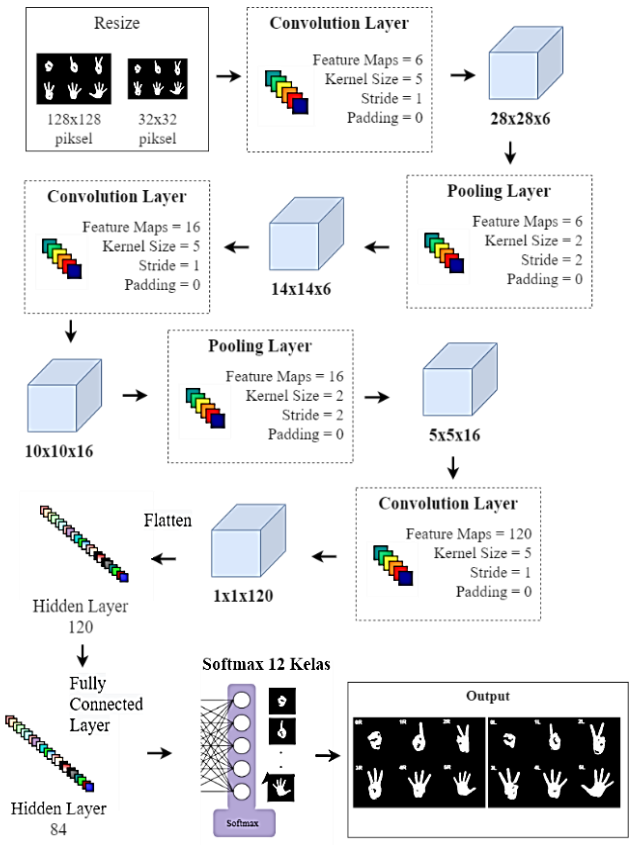
Layer (type)	***Output Shape***	***Param #***
conv2d (Conv2D)	*(None, 28, 28,6)*	456
average_pooling2d (AveragePooling2D)	*(None, 14. 14, 6)*	0
conv2d_1 (Conv2D)	*(None, 10, 10, 16)*	2416

average_pooling2d_1 (AveragePooling2D)	*(None, 5, 5, 16)*	0
conv2d_2 (Conv2D)	*(None, 1, 1, 120)*	48120
flatten (Flatten)	*(None, 120)*	0
dense (Dense)	*(None, 84)*	10164
dense_1 (Dense)	*(None, 12)*	1020
Total Param : 62176		
Trainale params : 62176		
Non-trainable params : 0		

Tabel 7. Ringkasan Model Hasil Implementasi Arsitektur *LeNet* dengan *Max Pooling*

Layer (type)	***Output Shape***	***Param #***
conv2d (Conv2D)	*(None, 28, 28,6)*	456
average_pooling2d (AveragePooling2D)	*(None, 14. 14, 6)*	0
conv2d_1 (Conv2D)	*(None, 10, 10, 16)*	2416
average_pooling2d_1 (AveragePooling2D)	*(None, 5, 5, 16)*	0
conv2d_2 (Conv2D)	*(None, 1, 1, 120)*	48120
flatten (Flatten)	*(None, 120)*	0
dense (Dense)	*(None, 84)*	10164
dense_1 (Dense)	*(None, 12)*	1020
Total Param : 62176		
Trainale params : 62176		
Non-trainable params : 0		

Proses *training* dataset pada arsitektur *LeNet* dapat dilihat pada Gambar 5.



Gambar 5. Proses *Training* pada Arsitektur *LeNet*

3.5. Implementasi Skenario Pengujian

Pada tahap ini menjelaskan hasil implementasi dengan pengujian per-kelas dan 12 skenario pengujian yang telah dilakukan dengan menggunakan dataset *training* dan *validation* yang belum dilakukan proses pelatihan model pada arsitektur *AlexNet* dan *LeNet*.

A. Arsitektur *AlexNet* menggunakan *Average Pooling* dengan *Optimizer* SGD

Hasil yang diperoleh pada skenario pertama dapat dlihat pada Tabel 8.

Tabel 8. Hasil Pengujian Skenario ke-1

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	47.78	87.76	47.78	61.87
0R	86.67	56.93	86.67	68.72
1L	48.89	30.45	48.89	37.53
1R	53.89	67.63	53.89	59.88
2L	3.89	35	3.89	7
2R	32.78	37.58	32.78	35.01
3L	56.11	60.12	56.11	58.05
3R	42.78	48.43	42.78	45.43
4L	35	42.28	35	38.3
4R	78.33	66.51	78.33	71.94
5L	88.89	66.12	88.89	75.83
5R	87.22	63.31	87.22	73.36
Overall	**55.19**	**55.15**	**55.19**	**52.74**
Total Waktu *Training*			**1345.64 detik**	

B. Arsitektur *AlexNet* menggunakan *Average Pooling* dengan *Optimizer* RMSprop

Hasil yang diperoleh pada skenario kedua dapat dlihat pada Tabel 9.

Tabel 9. Hasil Pengujian Skenario ke-2

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	100	100	100	100
0R	100	100	100	100
1L	97.78	100	97.78	98.88
1R	100	97.83	100	98.9
2L	98.89	100	98.89	99.44
2R	98.89	98.89	98.89	98.89
3L	100	97.83	100	98.9
3R	98.89	98.89	98.89	98.89
4L	100	100	100	100
4R	98.89	100	98.89	99.44
5L	100	100	100	100
5R	100	100	100	100
Overall	**99.44**	**99.45**	**99.44**	**99.44**
Total Waktu *Training*			**1509.17 detik**	

C. Arsitektur *AlexNet* menggunakan *Average Pooling* dengan *Optimizer* Adam

Hasil yang diperoleh pada skenario ketiga dapat dlihat pada Tabel 10.

Tabel 10. Hasil Pengujian Skenario ke-3

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	100	99.45	100	99.72
0R	100	57.69	100	73.17
1L	42.78	100	42.78	59.92
1R	77.78	99.29	77.78	87.23
2L	98.89	94.68	98.89	96.74
2R	92.22	94.32	92.22	93.26
3L	92.22	93.79	92.22	93
3R	91.11	96.47	91.11	93.71
4L	100	100	100	100
4R	100	99.45	100	99.72
5L	100	99.45	100	99.72
5R	100	91.84	100	95.74
Overall	**91.25**	**93.87**	**91.25**	**91**
Total Waktu *Training*			**1622.69 detik**	

D. Arsitektur *AlexNet* menggunakan *Max Pooling* dengan *Optimizer* SGD

Hasil yang diperoleh pada skenario keempaat dapat dlihat pada Tabel 11.

Tabel 11. Hasil Pengujian Skenario ke-4

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	80	90	80	87.71
0R	88.33	80.71	88.33	84.35
1L	85.56	77.78	85.56	81.48
1R	54.44	67.59	54.44	60.31
2L	53.89	56.07	53.89	54.96
2R	45	46.29	45	45.63
3L	52.22	59.12	52.22	55.46
3R	56.67	56.35	56.67	56.51
4L	90.56	84.9	90.56	87.63
4R	87.22	83.96	87.22	85.56
5L	96.11	92.02	96.11	94.02
5R	96.67	84.88	96.67	90.39
Overall	**73.89**	**73.3**	**73.89**	**73.42**
Total Waktu *Training*			**1328.46 detik**	

E. Arsitektur *AlexNet* menggunakan *Max Pooling* dengan *Optimizer* RMSprop

Hasil yang diperoleh pada skenario kelima dapat dlihat pada Tabel 12.

Tabel 12. Hasil Pengujian Skenario ke-5

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	100	97.83	100	98.9
0R	100	100	100	100
1L	98.89	97.27	98.89	98.07
1R	97.78	96.7	97.78	97.24
2L	96.67	98.31	96.67	97.48
2R	97.78	98.32	97.78	98.05
3L	96.11	98.3	96.11	97.19
3R	98.89	98.34	98.89	98.61
4L	97.22	98.87	97.22	98.04
4R	99.44	99.44	99.44	99.44
5L	100	98.9	100	99.45
5R	99.44	100	99.44	99.72
Overall	**98.52**	**98.52**	**98.52**	**98.52**
Total Waktu *Training*			**1595.35 detik**	

F. Arsitektur *AlexNet* menggunakan *Max Pooling* dengan *Optimizer* Adam

Hasil yang diperoleh pada skenario keenam dapat dlihat pada Tabel 13.

Tabel 13. Hasil Pengujian Skenario ke-6

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	70	100	70	82.32
0R	99.44	99.44	99.44	99.44
1L	88.33	77.18	88.33	82.38
1R	93.89	81.25	93.89	87.11
2L	91.67	93.75	91.67	92.7
2R	92.22	93.79	92.22	93
3L	93.89	96.06	93.89	94.94
3R	95.56	91.01	95.56	93.22
4L	98.33	96.2	98.33	97.25
4R	99.44	94371	99.44	97.02
5L	98.33	98.88	98.33	98.61
5R	94.44	100	94.44	97.14
Overall	**92.96**	**93.52**	**92.96**	**92.93**
Total Waktu *Training*			**1700.39 detik**	

G. Arsitektur *LeNet* menggunakan *Average Pooling* dengan *Optimizer* SGD

Hasil yang diperoleh pada skenario ketujuh dapat dlihat pada Tabel 14.

Tabel 14. Hasil Pengujian Skenario ke-7

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	83.89	78.24	83.89	80.97
0R	79.44	82.66	79.44	81.02
1L	74.44	70.16	74.44	72.24
1R	75	74.59	75	74.79
2L	73.33	58.93	73.33	65.35
2R	72.78	76.16	72.78	74.43
3L	57.22	72.54	57.22	63.98
3R	63.89	64.25	63.89	64.07
4L	93.33	93.85	93.33	93.59
4R	94.44	91.89	94.44	93.15
5L	92.22	98.22	92.22	95.13
5R	94.44	98.84	94.44	96.59
Overall	**79.54**	**80.03**	**79.54**	**79.61**
Total Waktu *Training*			**193.16 detik**	

H. Arsitektur *LeNet* menggunakan *Average Pooling* dengan *Optimizer* RMSprop

Hasil yang diperoleh pada skenario kedelapan dapat dlihat pada Tabel 15.

Tabel 15. Hasil Pengujian Skenario ke-8

Kelas	*Confusion Matrix* (%)			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	100	100	100	100
0R	100	100	100	100
1L	99.44	97.81	99.44	98.62
1R	98.33	99.44	98.33	98.88
2L	98.89	98.34	98.89	98.61
2R	100	99.45	100	99.72
3L	98.33	99.44	98.33	98.88
3R	98.89	99.44	98.89	99.16
4L	100	100	100	100

4R	100	100	100	100
5L	100	100	100	100
5R	100	100	100	100
Overall	**99.49**	**99.49**	**99.49**	**99.49**
Total Waktu *Training*			**183.94 detik**	

I. Arsitektur *LeNet* menggunakan *Average Pooling* dengan *Optimizer* Adam

Hasil yang diperoleh pada skenario kesembilan dapat dlihat pada Tabel 16.

Tabel 16. Hasil Pengujian Skenario ke-9

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	99.44	98.35	99.44	98.9
0R	99.44	98.44	99.44	98.44
1L	95.56	92.97	95.56	94.25
1R	91.11	94.8	91.11	92.92
2L	95	90	95	92.43
2R	93.89	93.37	93.89	93.63
3L	90.56	94.77	90.56	92.61
3R	93.89	95.48	93.89	94.68
4L	98.89	97.8	98.89	98.34
4R	98.89	100	98.89	99.44
5L	100	100	100	100
5R	99.44	99.44	99.44	99.44
Overall	**96.34**	**96.37**	**96.34**	**96.34**
Total Waktu *Training*			**323.733 detik**	

J. Arsitektur *LeNet* menggunakan *Max Pooling* dengan *Optimizer* SGD

Hasil yang diperoleh pada skenario kesepuluh dapat dlihat pada Tabel 17.

Tabel 17. Hasil Pengujian Skenario ke-10

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	82.22	90.24	82.22	86.05
0R	93.89	88.02	93.89	90.86
1L	83.33	83.33	83.33	83.33
1R	81.67	82.58	81.67	82.12
2L	72.78	74.43	72.78	73.6
2R	79.44	72.96	79.44	76.06
3L	76.67	78.86	76.67	77.75
3R	84.44	82.61	84.44	83.52
4L	93.89	82.86	93.89	93.37
4R	86.67	87.64	86.67	87.15
5L	97.22	96.69	97.22	96.95
5R	95	98.28	95	96.61
Overall	**85.6**	**85.71**	**85.6**	**85.61**
Total Waktu *Training*			**188.43 detik**	

K. Arsitektur *LeNet* menggunakan *Max Pooling* dengan *Optimizer* RMSprop

Hasil yang diperoleh pada skenario kesebelas dapat dlihat pada Tabel 18.

Tabel 18. Hasil Pengujian Skenario ke-11

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	98.89	93.34	98.89	98.61
0R	97.78	98.88	97.78	98.32
1L	96.67	92.06	96.67	94.31
1R	92.78	95.98	92.78	94.35
2L	96.11	96.11	96.11	96.11
2R	97.78	93.12	97.78	95.39
3L	95	95	95	95
3R	92.22	96.51	92.22	94.32
4L	98.33	98.33	98.33	98.33
4R	98.89	99.89	98.89	98.89
5L	99.44	100	99.44	99.72
5R	98.89	100	98.89	99.44
Overall	**96.9**	**96.94**	**96.9**	**96.9**
Total Waktu *Training*			**183.9 detik**	

L. Arsitektur *LeNet* menggunakan *Max Pooling* dengan *Optimizer* Adam

Hasil yang diperoleh pada skenario keduabelas dapat dlihat pada Tabel 19.

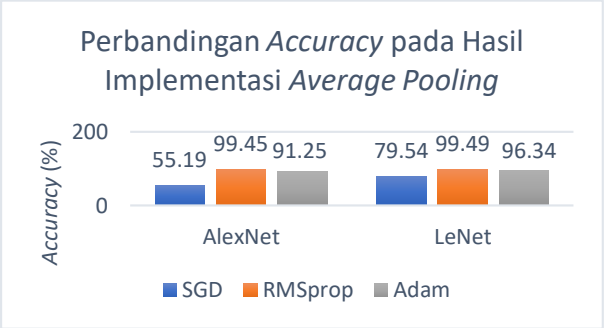
Tabel 19. Hasil Pengujian Skenario ke-12

Kelas	***Confusion Matrix* (%)**			
	Accuracy	*Precision*	*Recall*	*F1-Score*
0L	96.67	96.67	96.67	96.67
0R	97.22	96.15	97.22	96.69
1L	86.67	88.14	86.67	87.39
1R	90.56	87.88	90.56	84.06
2L	87.78	80.61	87.78	84.04
2R	85.56	81.48	85.56	83.47
3L	81.67	84	81.67	82.82
3R	78.89	82.56	78.89	80.68
4L	94.44	85.51	94.44	94.97
4R	96.11	83.51	96.11	94.79
5L	97.78	97.74	97.78	97.51
5R	98.33	98.33	98.33	98.33
Overall	**90.14**	**90.17**	**90.14**	**90.12**
Total Waktu *Training*			**226.69 detik**	

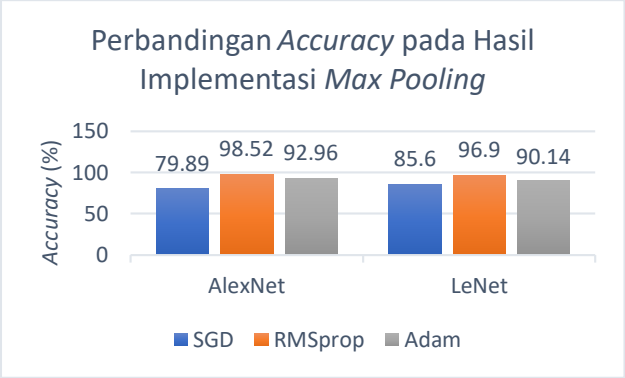
3.6. Analisis Hasil Pengujian

Analisis yang akan dilakukan adalah dengan melakukan analisis perbandingan *accuracy* pada jenis *pooling layer* yang digunakan kemudian melakukan analisis perbandingan 12 skenario pengujian pada penelitian ini.

Gambar 6 menunjukkan hasil perbandingan akurasi dengan penggunaan *Average Pooling* masing-masing mendapatkan tingkat akurasi sebesar 99.45% (*AlexNet-RMSprop*) serta 99.49% (*LeNet-RMSprop*).



Gambar 6. Perbandingan *Accuracy* pada Hasil Implementasi *Average Pooling*



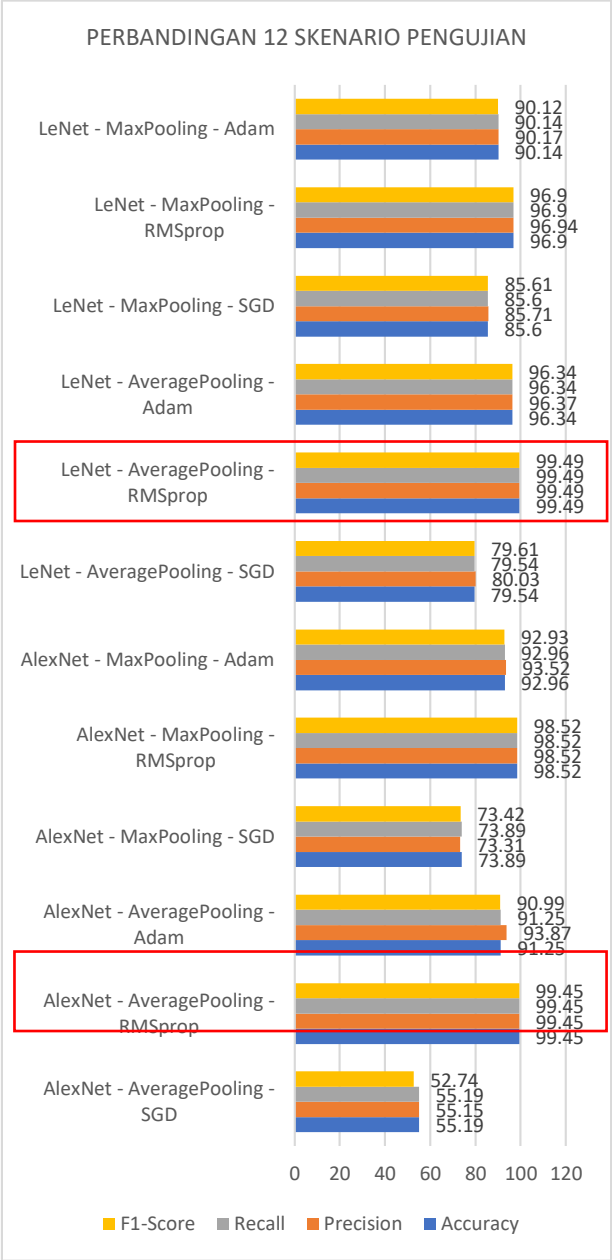
Gambar 7. Perbandingan *Accuracy* pada Hasil Implementasi *Max Pooling*

Gambar 7 menunjukkan Hasil perbandingan akurasi dengan penggunaan *Max Pooling* masing-masing mendapatkan tingkat akurasi sebesar 98.52% (*AlexNet-RMSprop*) kemudian pada arsitektur *LeNet* menghasilkan tingkat akurasi sebesar 96.9% (*LeNet-RMSprop*).

Kemudian perbandingan terhadap hasil skenario pengujian yang telah dilakukan secara keseluruhan yang dapat dilihat pada Gambar 8.

Hasil perbandingan klasifikasi pengujian terdapat 2 skenario pengujian terbaik dengan nilai keseluruhan *accuracy, recall, precision,* dan *f1-score* pada skenario *AlexNet-AveragePooling-RMSprop* sebesar 99,45 % dan *LeNet-AveragePooling-RMSprop* sebesar 99,49%.

Jika dilihat dari jumlah parameter yang di-*train AlexNet* memiliki 98124 parameter sedangkan *LeNet* memiliki 62716 parameter. Namun hasil yang didapat dengan menggunakan *LeNet* memiliki selisih yang kecil yaitu 0,04%. Jika berdasarkan jumlah parameter terlihat bahwa *LeNet* dengan parameter yang lebih sedikit dibandingkan *AlexNet* memberikan hasil yang lebih baik walaupun selisih hasil yang kecil. Jenis *Pooling* yang digunakan juga mempengaruhi hasil. Baik menggunakan *AlexNet* maupun *LeNet, Average Pooling* memberikan hasil paling baik dibandingkan dengan *Max Pooling*. Sama seperti *pooling, optimizer* baik pada *AlexNet* maupun *LeNet, optimizer* terbaik yaitu menggunakan *RMSProp*.



Gambar 8. Perbandingan 12 Skenario Pengujian

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dapat ditarik beberapa kesimpulan. Penggunaan metode *thresholding* dapat membantu untuk mendeteksi objek gestur angka pada tangan dengan mendapatkan bentuk dari objek dan menghilangkan *background* objek. Penggunaan *pooling layer* dengan *average pooling* dan *optimizer* RMSprop pada arsitektur *AlexNet* maupun *LeNet* mendapatkan akurasi dan *f1-score* yang tertinggi dalam melakukan klasifikasi gestur angka pada tangan dengan nilai akurasi dan *f1-score* sebesar 99.45% (AlexNet) dan 99.49% (LeNet). Untuk penelitian lebih lanjut pengenalan gestur angka bisa dengan menggunakan arsitektur yang lain. Selai itu dapat menggunakan *pooling* dan *optimizer* yang berbeda.

DAFTAR PUSTAKA

[1] H. Purnama, *Seni Bicara Bahasa Tubuh*. Yogyakarta: Mantra Books, 2014.

[2] A. Sunyoto and A. Harjoko, "Review Teknik, Teknologi, Metodologi dan Implementasi Pengenalan Gestur Tangan Berbasis Visi," *Semin. Nas. Apl. Teknol. Inf.*, p. H-7, 2014.

[3] J.-J. Park and C.-K. Kwon, "Korean Finger Number Gesture Recognition Based on CNN Using Surface Electromyography Signals," *J. Electr. Eng. Technol.*, vol. 16, no. 1, pp. 591–598, Jan. 2021, doi: 10.1007/s42835-020-00587-3.

[4] M. Bagus, S. Bakti, and Y. M. Pranoto, "Pengenalan Angka Sistem Isyarat Bahasa Indonesia Dengan Menggunakan Metode Convolutional Neural Network," *Semin. Nas. Inov. Teknol.*, pp. 11–16, 2019.

[5] M. E. Al Rivan, H. Irsyad, K. Kevin, and A. T. Narta, "Pengenalan Alfabet American Sign Language Menggunakan K-Nearest Neighbors Dengan Ekstraksi Fitur Histogram Of Oriented Gradients," *J. Tek. Inform. dan Sist. Inf.*, vol. 5, no. 3, Jan. 2020, doi: 10.28932/jutisi.v5i3.1936.

[6] W. Kurniawan, "Program Aplikasi Komputer Pengenalan Angka Dengan Pose Jari Tangan Sebagai Media Pembelajaran Interaktif Anak Usia Dini," *J. Sains dan Mat.*, pp. 68–73, 2012.

[7] H. Dafitri, M. S. Asih, and R. I. Astuti, "Media Interaktif Pengenalan Angka Dengan Jari Tangan Menggunakan Metode PCA (Principal Component Analysis)," *Query J. Sist. Inf.*, vol. 03, no. 02, pp. 57–65, 2019.

[8] R. M. Awangga, R. Andarsyah, and E. C. Putro, *Tutorial Object Detection People With Faster region-Based Convolutional Neural Network(Faster R-CNN)*. Jakarta: Kreatif, 2020.

[9] R. Rouhi, M. Jafari, S. Kasaei, and P. Keshavarzian, "Benign and malignant breast tumors classification based on region growing and CNN segmentation," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 990–1002, 2015, doi: 10.1016/j.eswa.2014.09.020.

[10] R. Rokhana, J. Priambodo, T. Karlita, I. M. G. Sunarya, E. M. Yuniarno, I. K. E. Purnama, M. H. Purnomo., "Convolutional Neural Network untuk Pendeteksian Patah Tulang Femur pada Citra Ultrasonik B–Mode," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 8, no. 1, p. 59, 2019, doi: 10.22146/jnteti.v8i1.491.

[11] P. Koryakin, "Fingers," 2019. https://www.kaggle.com/koryakinp/fingers (accessed Jun. 13, 2021).

[12] M. E. Al Rivan and A. G. Riyadi, "Perbandingan Arsitektur LeNet dan AlexNet Pada Metode Convolutional Neural Network Untuk Pengenalan American Sign Language," *J. Komput. Terap.*, vol. 7, no. 1, pp. 53–61, 2021, doi: https://doi.org/10.35143/jkt.v7i1.4489.

[13] I. W. S. E. Putra, A. Y. Wijaya, and R. Soelaiman, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, 2016, doi: 10.12962/j23373539.v5i1.15696.

[14] O. N. Putri, "Implementasi metode cnn dalam klasifikasi gambar jamur pada analisis image processing," Tugas Akhir. Universitas Islam Indonesia, 2020.

[15] W. Setiawan, "Perbandingan Arsitektur Convolutional Neural Network Untuk Klasifikasi Fundus," *J. Simantec*, vol. 7, no. 2, pp. 48–53, 2020, doi: 10.21107/simantec.v7i2.6551.

[16] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," *JUSTINDO (Jurnal Sist. dan Teknol. Inf. Indones.*, vol. 3, no. 2, pp. 49–56, 2018.