Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.938047
Vol. 11, No. 3, Juni 2024, hlm. 519-528 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

MODEL *DEEP LEARNING* UNTUK KLASIFIKASI OBJEK PADA GAMBAR *FISHEYE*

Riza Ayu Putri[*1], Arief Suryadi Satyawan[2], Johanes Adi Prihantono[3], Rinda Safana Linggi[4], I Gusti Ayu Putri Surya Paramita[5], Ni Kadek Emy Iswarawati[6], Fabian Akbar[7], Prio Adjie Utomo[8],

[1,2,3,4,5,6,7,8]Universitas Nurtanio, Bandung
Email : [1]riza.te19@student.unnur.ac.id, [2]arief.suryadi@akane.waseda.jp, [3]prihantono.adi@gmail.com, [4]rinda.tp19@student.unnur.ac.id, [5]putrisuryaparamita@gmail.com, [6]emyiswarawatti@gmail.com, [7]fabianakbar.te19@student.unnur.ac.id, [8]prio.lp19@student.unnur.ac.id
*Penulis Korespondensi

(Naskah masuk: 14 November 2023, diterima untuk diterbitkan: 10 Juni 2024)

Abstrak

Pengenalan suatu objek secara otomatis adalah suatu pekerjaan yang sangat penting seperti halnya untuk mengidentifikasi sebuah objek tertentu. Jika hal ini dilakukan oleh manusia maka akan sulit untuk mendapatkan hasil yang baik dengan konsisten, oleh sebab itu digunakan komputer. Komputer dapat mengenali objek selayaknya kemampuan manusia dalam mengenali objek, dengan cara mengamati gambar yang diperoleh dari kamera, dan menerapkan metode pengenalan pada gambar tersebut. Pada penelitian ini metode pengenalan objek akan dikembangkan dengan menggunakan kamera *fisheye* yang memiliki luas tangkap empat kali kamera konvensional. Metode pengenalan objek yang digunakan yaitu *deep learning* dengan arsitektur CNN (*Convolution Neural Network*). CNN memiliki kemampuan untuk mengenali objek dalam gambar. Model CNN yang digunakan terdiri dari 1 *layer*, 2 *layer*, 3 *layer*, dan 7 *layer*. Sedangkan untuk melatih dan memvalidasi model tersebut digunakan 900 gambar dataset. Hasil pengujian pada penelitian ini menunjukan bahwa pada 7 *layer* CNN menghasilkan nilai *presisi*, *recall* dan akurasi tertinggi dengan komposisi nilai *presisi* 98,56%, *recall* 98,5% dan akurasi 98,59%. Nilai tersebut menunjukan bahwa hasil klasifikasi terhadap ketiga klasifikasi objek gambar manusia pada gambar *fisheye* dapat dilakukan dengan sangat baik.

Kata kunci: *Deep Learning, Convolution Neural Network, Layer, Fisheye*

DEEP LEARNING MODEL FOR OBJECT CLASSIFICATION IN FISHEYE IMAGES

Abstract

Object recognition is an important method of identifying object in an image. If this method is carried out by the humans, it is difficult to work continuously, therefore, it can be done by a computer. The computer can recognize an object on an image taken from a camera as if a human if it is trained whit a certain dataset of be object. In this research, a method a recognizing object is developed by using images captured from a fisheye camera. This camera provides better field of view four times then confessional camera. The object recognition method relies on convolutional neural network architecture that has capability to recognize object on an image. The CNN model is developed in four types of models. The first model consists of 1 layer, while the second model consist of 2 layers. The thread model consists of 3 layers, while the last consist of 7 layers. The number of datasets used for training and testing each model is 900 images. The experiment results showed that the four model which consist of 7 layers provide the best result. This is confirmed by the number of precisions, recall, and accuracy, which reach 98,56%, 98,5%, and 98,59% respectively. This result means that can classify three different objects of human on fisheye images well.

Keywords: *Deep Learning, Convolution Neural Network, Layer, Fisheye*

1. PENDAHULUAN

Pengenalan suatu objek secara otomatis adalah suatu pekerjaan yang sangat penting seperti halnya untuk mengidentifikasi sebuah objek tertentu. Jika hal ini dilakukan oleh manusia maka akan sulit untuk mendapatkan hasil yang baik dengan konsisten. Oleh sebab itu sistem pengenalan objek secara otomatis banyak dikembangkan orang dengan berbantukan kamera, komputer, dan algoritma identifikasi objek.

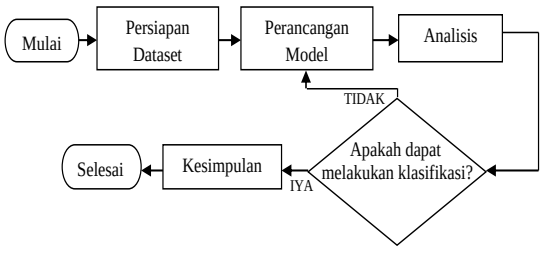
Adapun algoritma identifikasi objek yang berkembang adalah teknik klasifikasi objek berbasis *deep learning*. Pada umumnya algoritma tersebut bekerja menggunakan gambar dari kamera konvensional yang memiliki luas tangkap visual terbatas. Jika ingin mengamati dengan luas tangkap yang lebih besar maka dibutuhkan lebih dari satu buah kamera konvensional. Hal ini tidak efisien, karena pada dasarnya bisa digunakan kamera *fisheye* yang memiliki luas tangkap empat kali kamera konvensional. Untuk itu pada penelitian ini akan dibangun model algoritma klasifikasi objek berbasis *deep learning* menggunakan arsitektur CNN (*Convolution Neural Network*) untuk klasifikasi objek pada gambar *fisheye*.

Klasifikasi objek sangat berguna untuk bisa mengenali jenis objek tertentu dari sekian objek yang harus diklasifikasi. Komputer harus mampu mengenali objek pada gambar selayaknya kemampuan manusia dalam mengenali objek, seperti mengenali bunga, manusia, hewan, dan lain-lain. Bagi manusia, hal ini merupakan pekerjaan yang mudah. Namun berbeda dengan komputer. Komputer harus belajar dalam mengenali objek atau biasa disebut dengan *machine learning*. Penggunaan data dalam jumlah besar dibutuhkan agar komputer dapat mengenali pola pada suatu objek. Untuk bisa mengklasifikasi suatu objek dengan otomatis, salah satunya dengan menggunakan kamera. Kamera menghasilkan gambar-gambar objek yang nantinya akan digunakan komputer untuk mengenali pola pada suatu objek. Pada penelitian ini kamera yang digunakan adalah *fisheye camera* dengan luas tangkap empat kali kamera konvensional. Diharapkan klasifikasi objek dalam hal ini manusia pada gambar hasil kamera *fisheye* dapat diklasifikasikan dengan baik.

Sejumlah penelitian telah dilakukan oleh para peneliti dalam berbagai implementasi untuk klasifikasi gambar, diantaranya yaitu pada penelitian yang dikembangkan oleh (Peryanto, Yudhana & Umar, 2019) dalam penelitian ini, dilakukan penyesuaian jumlah *epoch* dan peningkatan ukuran data latih untuk meningkatkan akurasi dalam klasifikasi gambar mobil dan motor. Sedangkan penelitian yang dikembangkan oleh (Sutejo, 2022) menggunakan algoritma CNN yang diimplementasikan menggunakan *tensorflow* dengan bahasa pemrograman *Python* untuk mengklasifikasikan objek gambar wajah manusia. Kemudian penelitian yang dikembangkan oleh (Ocktavia Nurima Putri, 2020) dilakukan dengan menggunakan 100 *epoch*, skenario perbandingan data latih : validasi 80% : 20%, dan ukuran kernel yang digunakan adalah 3x3, untuk mengklasifikasikan gambar jamur berdasarkan genusnya dengan metode *deep learning*. Penelitian terkait lainnya yang dikembangkan oleh (Lee, 2021) menggunakan algoritma CNN untuk mengenali dan mengklasifikasikan gambar-gambar objek *moths* yang sangat kecil dan mendapatkan data gambar yang akurat. Selain itu, penelitian terkait lainnya yang dikembangkan oleh (Wang, Fan & Wang, 2021) Dalam penelitian ini, dilakukan perbandingan dan analisis antara *traditional machine learning* dan algoritma klasifikasi gambar dalam *deep learning*. Perbedaan utama dalam penelitian ini dibandingkan dengan penelitian serupa lainnya terletak pada sumber dataset yang dibuat secara independen menggunakan kamera *fisheye*, serta modifikasi yang dilakukan pada arsitektur CNN dengan penambahan *layer* pada tahap *feature extraction*.

2. METODE PENELITIAN

Metode penelitian yang digunakan pada penelitian ini meliputi beberapa tahapan, seperti pada Gambar 1. Pada diagram alir menunjukkan bahwa langkah awal dalam proses perancangan penelitian adalah mempersiapkan dataset yang diambil secara independen menggunakan kamera *fisheye*, yang mencakup tiga kategori gambar wajah manusia. Langkah berikutnya adalah merancang model klasifikasi objek sesuai dengan rencana yang telah disusun. Kemudian, dilanjutkan dengan tahap analisis hasil untuk menguji kesesuaian model yang telah dibuat sesuai dengan tujuan penelitian. Sebuah model dianggap sesuai jika dapat melakukan *training* pada dataset gambar *fisheye* dengan objek manusia, mampu melakukan klasifikasi pada data baru, dan mengkategorikan data baru ke dalam kelas yang sesuai. Jika model tidak memenuhi kriteria ini, maka proses akan kembali ke tahap perancangan model. Jika model sudah sesuai, maka penelitian akan melanjutkan ke tahap terakhir, yaitu pembuatan laporan penelitian. Dalam penelitian ini, akurasi ditentukan oleh berbagai skenario pelatihan yang telah dirancang untuk mencapai arsitektur model CNN yang optimal.

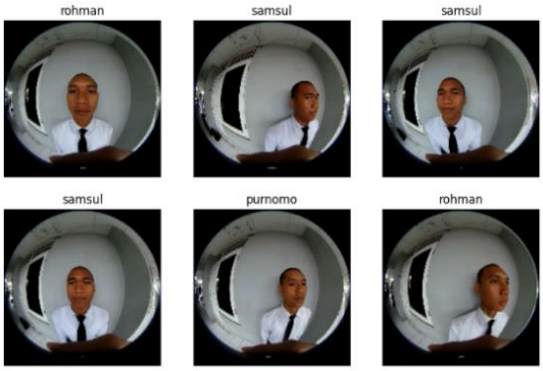


Gambar 1. Diagram Alir Penelitian

2.1 Persiapan Dataset

Proses pengambilan gambar dilakukan menggunakan kamera *fisheye* Ricoh Theta untuk mengambil objek berupa gambar manusia sebagai dataset dilakukan pada jarak 0,5 meter. Data gambar yang diambil di transfer ke komputer melalui koneksi USB. Hasil dari kamera *fisheye* berupa *video* dengan durasi per dataset kurang lebih satu menit. Kemudian dilakukan proses konversi menjadi jpg menggunakan matlab, dan dataset yang dihasilkan berjumlah 300 gambar per dataset. Sedangkan data gambar baru

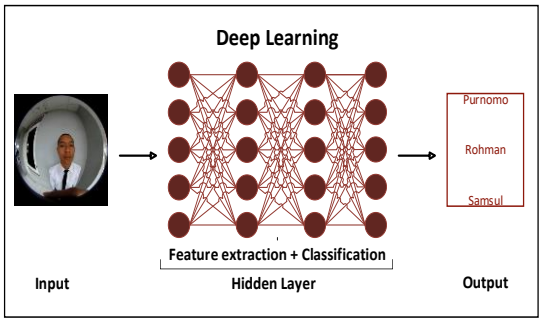
berjumlah 27 gambar. Selanjutnya memotong dan mumutar gambar yang telah di*convert* menggunakan matlab. Kemudian membuat folder klasifikasi dataset menjadi tiga pengelompokkan data pada kategori yang sesuai, yaitu Purnomo, Rohman, dan Samsul. Sampel dataset ditunjukkan pada Gambar 2.



Gambar 2. Dataset Gambar *Fisheye*

2.2 Perancangan Model

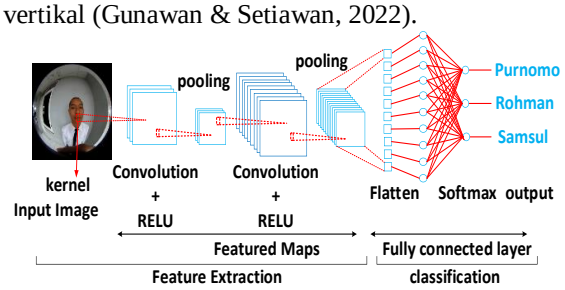
Penelitian ini menggunakan metode *deep learning* dengan arsitektur CNN untuk melakukan klasifikasi objek pada gambar *fisheye*. *Deep Learning* adalah sub-bidang dari *machine learning* berdasarkan jaringan saraf tiruan. Pada *deep learning*, komputer belajar mengklasifikasikan langsung dari gambar, teks atau suara. *Deep learning* merupakan teknik dalam jaringan saraf, menggunakan banyak lapisan untuk pembelajaran yang lebih kompleks. Lapisan dalam *deep learning* terdiri dari tiga bagian, yaitu *input layer*, *hidden layer* dan *output layer* (Tutut Furi Kusumaningrum, 2018) seperti pada gambar 3.



Gambar 3. *Deep Learning*

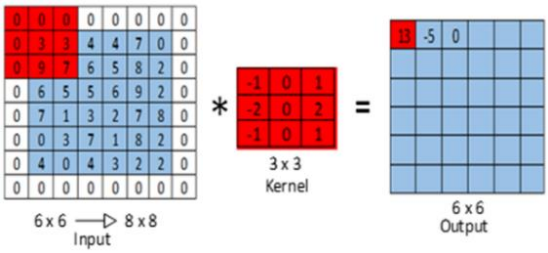
Salah satu arsitektur *deep learning* yang sedang berkembang saat ini adalah *convolutional neural network* (CNN). CNN memiliki kemampuan untuk mengenali objek dalam gambar. Arsitektur CNN terdiri dari *feature extraction* dan *classification layer*, seperti pada Gambar 4.

Pada tahapan *feature extraction*, terdiri dari *convolution layer*, *relu*, dan *pooling layer*. Lapisan pertama yang menerima input gambar adalah *convolutional layer*. Konvolusi *layer* digunakan untuk ekstraksi fitur. Hasil dari operasi konvolusi adalah *feature maps* yang digunakan untuk *input layer* berikutnya. *Feature maps* adalah hasil perkalian antara kernel dengan *array input* pada posisi yang berbeda, dengan mengskening secara horizontal dan vertikal (Gunawan & Setiawan, 2022).



Gambar 4. Arsitektur CNN

Convolution dijalankan pada data *input* menggunakan sebuah kernel, pada penelitian ini kernel yang digunakan berukuran 3x3. Operasi *convolution* dilakukan dengan menggeser kernel diatas input gambar. Pada Gambar 5 terdapat kernel (kotak berwarna merah) dan *image* yang akan dikonvolusi (kotak berwarna biru). Kernel akan bergerak dari pojok kiri atas sampai pojok kanan bawah, perkalian matriks dilakukan dan menjumlahkan hasilnya ke dalam peta fitur. Sehingga hasil konvolusi citra dapat dilihat pada citra sebelah kanan.



$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a,b)g(x-a,y-b)$$

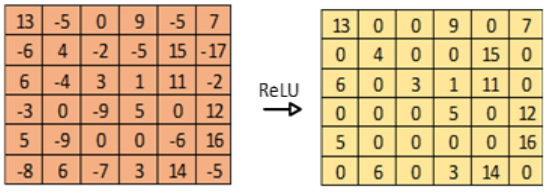
$$= (0x(-1)) + (0x0) + (0x1) + (0x(-2)) + (3x0) + (3x2) + 0x(-1) + (9x0) + (7x1)$$

$$= 0 + 0 + 0 + 0 + 0 + 6 + 0 + 0 + 7$$

$$= 13$$

Gambar 5. Proses *Convolution Layer*

Fungsi aktivasi adalah fungsi yang digunakan dalam jaringan saraf untuk mengaktifkan atau menonaktifkan *neuron* berdasarkan inputnya. Dalam kasus CNN, keluaran dari operasi konvolusi akan diteruskan melalui fungsi aktivasi.

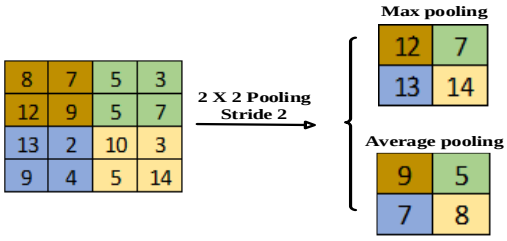


Gambar 6. *Activation Function* ReLU

Pada penelitian ini menggunakan Fungsi aktivasi ReLU (*Rectified Linear Unit*). Proses yang terjadi pada fungsi ini sangat sederhana. Fungsi ini akan mengeluarkan nilai berdasarkan nilai input, jika

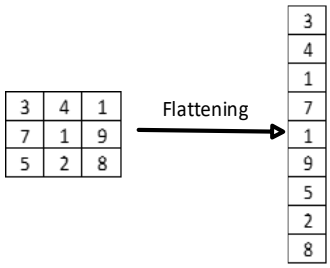
nilainya positif dan 0 jika nilai inputnya negative (Dzaky, 2021) seperti terlihat pada Gambar 6.

Pooling layer digunakan untuk memperkecil ukuran citra (*downsampling*), tujuannya untuk memudahkan operasi konvolusi pada *layer* berikutnya dan mempercepat komputasi. Proses ini dapat dilakukan dengan berbagai teknik, antara lain *max pooling* (mengambil nilai maksimal dari *sub-image*), *average pooling* (mengambil nilai rata-rata dari *sub-image*) (Alamsyah & Pratama, 2020) ditunjukkan pada Gambar 7. Pada penelitian ini menggunakan teknik *max pooling* 2x2.



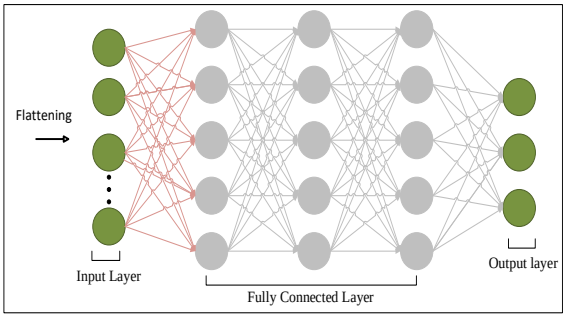
Gambar 7. *Pooling Layer*

Kemudian tahapan selanjutnya yaitu *classification layer* terdiri dari beberapa *layer* yang berisi *neuron-neuron* yang terhubung penuh dengan lapisan lainnya. *Layer* ini menerima *input* dari *output layer feature extrtaction*, kemudian di proses pada *flatten* dan *fully connected* untuk menghasilkan *output* berupa klasifikasi untuk masing-masing kelas (Ocktavia Nurima Putri, 2020). *Flatten* berfungsi untuk mengubah hasil dari *convolution layer* berupa *feature map* berbentuk multidimensi *array* menjadi matrik satu dimensi. *Flatten* tersebut selanjutnya dijadikan *input* untuk *fully connected layer* (Herdianto & Nasution, 2023) ditunjukkan pada Gambar 8.



Gambar 8. *Flatten*

Fully connected layer adalah lapisan dimana semua *neuron* yang dihasilkan dilapisan sebelumnya terhubung ke *neuron* dilapisan berikutnya. *Fully connected layer* bertujuan untuk memproses data sehingga dapat diklasifikasikan. *Output layer* pada *fully connected layer* berupa prediksi kelas pada gambar *input*. Lapisan ini menggunakan fungsi aktivasi *softmax* pada lapisan keluaran untuk klasifikasi. Fungsi aktivasi ini sering digunakan pada jaringan saraf dengan beberapa kelas *output* (*multiclass*) (Gunawan & Setiawan, 2022) dapat dilihat pada Gambar 9.



Gambar 9. *Fully Connected Layer*

2.3 EVALUASI MODEL

Evaluasi pada penelitian ini menggunakan *confusion matrix* dengan menghitung nilai akurasi, *presisi*, dan *recall* yang ditunjukkan pada persamaan 1-5.

$$Akurasi = \frac{TP}{Jumlah\ Data}\ X\ 100\% \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$All\ Precision = \frac{Precision\ P+R+S}{Jumlah\ Kelas} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$All\ Recall = \frac{Recall\ P+R+S}{Jumlah\ Kelas} \quad (5)$$

3. HASIL DAN PEMBAHASAN

Ada beberapa pengujian yang dilakukan untuk mendapatkan hasil terbaik dari model sistem klasifikasi objek pada gambar *fisheye* berbasis *Deep Learning* dengan arsitektur CNN. Dalam hal ini terdapat *training* dataset dan *testing* dataset, yang mana *training dataset* merupakan kumpulan data yang digunakan untuk melatih model. Sedangkan, *testing dataset* adalah kumpulan data yang digunakan untuk menguji model setelah proses latihan selesai.

3.1 Hasil Merubah *Batch Size* Dan Ukuran Gambar

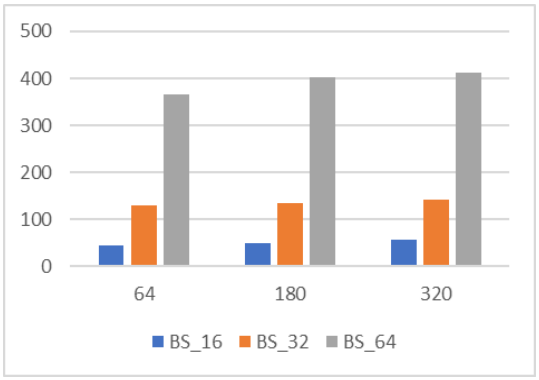
Berdasarkan parameter yang ditunjukkan pada Gambar 10. Kemudian dilakukan proses *training* dengan memberikan nilai *batch size* yang berbeda-beda, yaitu 16, 32, dan 64. Sedangkan untuk *height* dan *width* sebesar 64, 180, dan 320. Hasilnya dijelaskan pada Tabel 1 dan Gambar 11. Pada grafik dapat dilihat bahwa untuk setiap nilai *batch size* yang meningkat dari 16, 32, dan 64, menyebabkan kenaikan waktu proses *training*. Demikian pula jika ukuran gambar untuk proses *training* diubah dari 64, 180, dan 320, maka proses *training* juga meningkat meskipun nilai *batch size*nya tetap. Dengan demikian ukuran *batch size* dan ukuran gambar untuk proses akan menentukan lama waktu *training*.

```
num_classes = len(class_names)

model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Gambar 10. Parameter 3 *Layer* CNN

Tabel 1. Hasil Perubahan *Batch Size*, *Height* dan *Width*

Batch Size	Height & Width	Waktu 1 Epoch
16	64	45 ms
	180	49 ms
	320	57 ms
32	64	130 ms
	180	134 ms
	320	143 ms
64	64	367 ms
	180	402 ms
	320	411 ms



Gambar 11. Grafik Perubahan *Batch Size*, *Height*, dan *Width*

3.2 Hasil Reduksi *Overfitting*

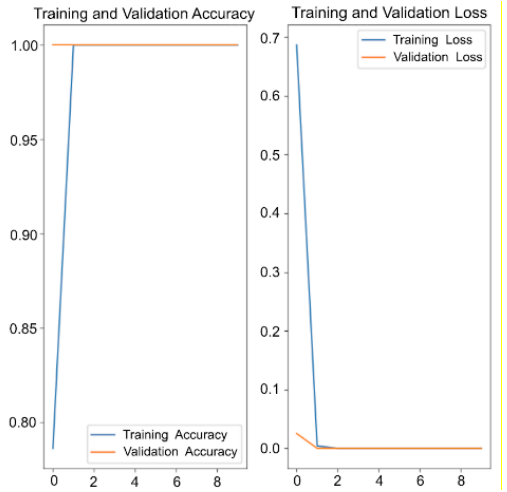
Berdasarkan parameter yang ditunjukkan pada gambar 12.a. model CNN tanpa reduksi *overfitting* dan gambar 12.b. model CNN dengan reduksi *overfitting*. Kemudian dilakukan proses *training* dengan memberikan nilai *batch size* sebesar 32, *height* 180, dan *width* 180, serta *epoch* yang digunakan pada proses ini adalah 10 dan 15. Hasilnya dapat dilihat pada Gambar 13.a. dan 13.b. Ketika menggunakan gambar dengan kamera *fisheye* pada *epoch* 10 didapatkan hasil *validasi accuracy* 1.0000, tetapi ketika dilanjutkan proses reduksi *overfitting* dengan augmentasi data dan menambah *layer dropout* serta menggunakan *epoch* 15 hasil *validasi accuracy* sebesar 0.4167. Hal ini berarti bahwa untuk jenis gambar *fisheye* proses augmentasi data dan penambahan *layer dropout* tidak diperlukan lagi.

```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)

 conv2d (Conv2D)             (None, 180, 180, 16)

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)
 2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 64)

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)
 2D)

 flatten (Flatten)           (None, 30976)

 dense (Dense)               (None, 128)

 dense_1 (Dense)             (None, 3)

=================================================================
Total params: 3,989,027
Trainable params: 3,989,027
Non-trainable params: 0
_________________________________________________________________
```

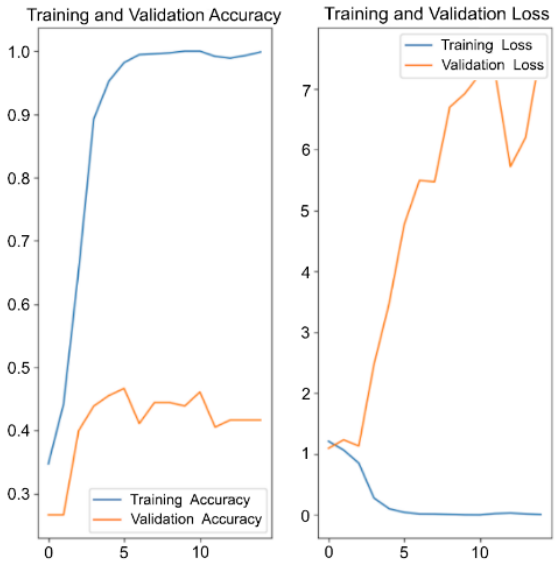
Gambar 12.a. Hasil Model CNN Tanpa Reduksi *Overfitting*

```
model.summary()
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3
 rescaling_2 (Rescaling)     (None, 180, 180, 3
 conv2d_3 (Conv2D)           (None, 180, 180, 1
 max_pooling2d_3 (MaxPooling  (None, 90, 90, 16
 2D)
 conv2d_4 (Conv2D)           (None, 90, 90, 32)
 max_pooling2d_4 (MaxPooling  (None, 45, 45, 32
 2D)
 conv2d_5 (Conv2D)           (None, 45, 45, 64)
 max_pooling2d_5 (MaxPooling  (None, 22, 22, 64
 2D)
 dropout (Dropout)           (None, 22, 22, 64)
 flatten_1 (Flatten)         (None, 30976)
 dense_2 (Dense)             (None, 128)
 dense_3 (Dense)             (None, 3)
=================================================================
Total params: 3,989,027
Trainable params: 3,989,027
Non-trainable params: 0
_________________________________________________________________
```

Gambar 12.b. Hasil Model CNN Dengan Reduksi *Overfitting*



Gambar 13.a. Hasil *Training* 10 *Epoch* tanpa Reduksi *Overfitting*



Gambar 13.b. Hasil *Training* 15 *Epoch* Reduksi *Overfitting*

3.3 Hasil Penambahan *Layer* Pada Arsitektur CNN

Pada pengujian ini membangun konfigurasi *layer* CNN dengan penambahan *layer* dalam arsitektur CNN yang dilakukan pada *feature extraction*. Konfigurasi *layer* CNN dengan kedalaman 1 *layer*, 2 *layer*, 3 *layer*, dan 7 *layer* dapat dilihat pada Gambar 14-17. Kemudian berdasarkan parameter tersebut dilakukan proses *training* dengan memberikan nilai *batch size* sebesar 32, *height* 180, dan *width* 180, serta *epoch* yang digunakan pada proses ini adalah 10. Hasilnya dapat dilihat pada Gambar 18-21. Ketika menggunakan gambar dengan kamera *fisheye*, serta konfigurasi CNN dengan kedalaman 1 *layer*, 2 *layer*, 3 *layer*, dan 7 *layer* dan tidak digunakan proses reduksi *overfitting* dengan augmentasi data dan menambah *layer dropout*, didapatkan hasil *validasi accuracy* pada masing-masing penambahan *layer* sebesar 1.0000.

```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 flatten (Flatten)           (None, 129600)            0

 dense (Dense)               (None, 32)                4147232

 dense_1 (Dense)             (None, 3)                 99

=================================================================
Total params: 4,147,779
Trainable params: 4,147,779
Non-trainable params: 0
_________________________________________________________________
```

Gambar 14. Hasil Model CNN 1 *Layer*

```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 flatten (Flatten)           (None, 64800)             0

 dense (Dense)               (None, 64)                4147264

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 4,152,547
Trainable params: 4,152,547
Non-trainable params: 0
_________________________________________________________________
```

Gambar 15. Hasil Model CNN 2 *Layer*

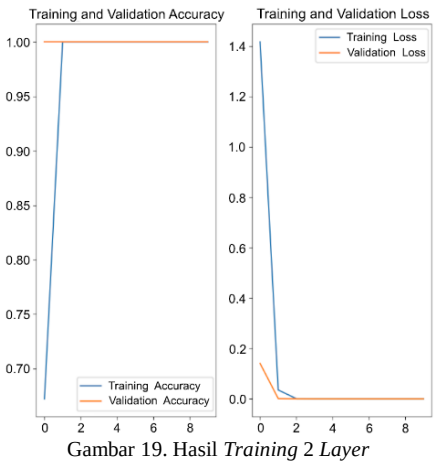
```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 flatten (Flatten)           (None, 30976)             0

 dense (Dense)               (None, 128)               3965056

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 3,989,027
Trainable params: 3,989,027
Non-trainable params: 0
_________________________________________________________________
```

Gambar 16. Hasil Model CNN 3 *Layer*

```
model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d (Conv2D)             (None, 180, 180, 16)      448

 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 22, 22, 128)       73856

 max_pooling2d_3 (MaxPooling  (None, 11, 11, 128)      0
 2D)

 conv2d_4 (Conv2D)           (None, 11, 11, 256)       295168

 max_pooling2d_4 (MaxPooling  (None, 5, 5, 256)        0
 2D)

 conv2d_5 (Conv2D)           (None, 5, 5, 512)         1180160

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 conv2d_6 (Conv2D)           (None, 2, 2, 1024)        4719616

 max_pooling2d_6 (MaxPooling  (None, 1, 1, 1024)       0
 2D)

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 2048)              2099200

 dense_1 (Dense)             (None, 3)                 6147

=================================================================
Total params: 8,397,731
Trainable params: 8,397,731
Non-trainable params: 0
```

Gambar 17. Hasil Model CNN 7 *Layer*

Gambar 18. Hasil *Training* 1 *Layer*

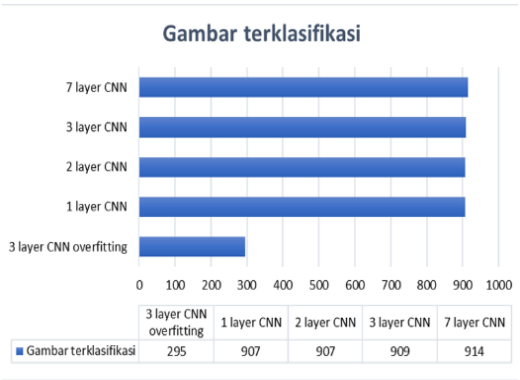


Gambar 19. Hasil *Training* 2 *Layer*

Gambar 20. Hasil *Training* 3 *Layer*

Gambar 21. Hasil *Training* 7 *Layer*

Kemudian hasil pengujian klasifikasi objek gambar *fisheye* dapat dilihat pada Gambar 22. Hasilnya pada 3 *layer* CNN dengan reduksi *overfitting* gambar yang terklasifikasi hanya sejumlah 295 gambar, sedangkan pada 1 *layer*, 2 *layer*, 3 *layer*, dan 7 *layer* CNN tanpa reduksi *overfitting* mendapatkan hasil yang baik dalam mengklasifikasikan gambar, dengan jumlah gambar yang terklasifikasi berjumlah 907 gambar pada 1 *layer* CNN, 907 gambar pada 2 *layer* CNN, 909 gambar pada 3 *layer* CNN, dan 914 gambar pada 7 *layer* CNN. Sehingga dapat dikatakan bahwa penambahan *layer* pada model arsitektur CNN mendapatkan hasil yang baik untuk klasifikasi objek pada gambar *fisheye* tanpa reduksi *overfitting*.



Gambar 22. Grafik Hasil Pengujian Gambar *Fisheye*

3.3.5 Analisis Hasil

Hal ini dimaksudkan untuk mengetahui kinerja sistem klasifikasi objek saat diuji coba dengan menggunakan konsep *confusion matrix*. Tabel 2-6 memperlihatkan *confusion matrix* pada konfigurasi 3 *layer* CNN dengan reduksi *overfitting*, konfigurasi kedalaman *layer* CNN yang berbeda yaitu1 *layer*, 2 *layer*, 3 *layer* dan 7 *layer* CNN.

Tabel 2. *Confusion Matrix* 3 *Layer* CNN Dengan Reduksi *Overfitting*

		Prediksi			
	Kelas	Purnomo	Rohman	Samsul	Total
Aktual	Purnomo	200	100	0	300
	Rohman	205	95	0	300
	Samsul	162	138	0	300
	Total	567	333	0	900

Tabel 3. *Confusion Matrix* 1 *Layer* CNN

		Prediksi			
	Kelas	Purnomo	Rohman	Samsul	Total
Aktual	Purnomo	306	1	2	309
	Rohman	9	300	0	309
	Samsul	7	1	301	309
	Total	322	302	303	927

Tabel 4. *Confusion Matrix* 2 *Layer* CNN

		Prediksi			
	Kelas	Purnomo	Rohman	Samsul	Total
Aktual	Purnomo	306	2	1	309
	Rohman	9	300	0	309
	Samsul	6	2	301	309
	Total	321	304	302	927

Tabel 5. *Confusion Matrix* 3 *Layer* CNN

		Prediksi			
	Kelas	Purnomo	Rohman	Samsul	Total
Aktual	Purnomo	308	0	1	309
	Rohman	9	300	0	309
	Samsul	7	1	301	309
	Total	324	301	302	927

Tabel 6. *Confusion Matrix* 7 *Layer* CNN

		Prediksi			
	Kelas	Purnomo	Rohman	Samsul	Total
Aktual	Purnomo	308	1	0	309
	Rohman	4	305	0	309
	Samsul	4	4	301	309
	Total	316	310	301	927

Tabel 7. Hasil Perhitungan *Presisi*, *Recall* dan Akurasi

Model	*Presisi*	*Recall*	Akurasi
3 *layer* CNN *overfitting*	21,23%	32,73%	32,77%
1 *layer* CNN	97,8%	97,8%	97,84%
2 *layer* CNN	97,8%	97,8%	97,84%
3 *layer* CNN	98%	98%	98,05%
7 *layer* CNN	98,56%	98,5%	98,59%

Kemudian pada Tabel 7 ditunjukkan hasil dari perhitungan *presisi*, *recall*, dan akurasi. Hasilnya pada 3 *layer* CNN dengan reduksi *overfitting* didapatkan hasil *presisi* 21,23%, *recall* 32,73%, akurasi 32,77%, pada 1 *layer* CNN didapatkan hasil *presisi* 97,8%, *recall* 97,8%, akurasi 97,84%, serta pada 2 *layer* CNN didapatkan hasil *presisi* 97,8%, *recall* 97,8%, akurasi 97,84%, kemudian pada 3 *layer* CNN didapatkan hasil *presisi* 98%, *recall* 98%, akurasi 98,05%, dan pada 7 *layer* CNN didapatkan hasil *presisi* 98,56%, *recall* 98,5%, akurasi 98,59%. Hasil pada 7 *layer* CNN tanpa reduksi *overfitting* mendapatkan hasil terbaik untuk klasifikasi objek, nilai tersebut menunjukkan bahwa hasil terhadap ketiga klasifikasi objek gambar manusia termasuk kategori baik dalam mengklasifikasikan gambar menggunakan kamera *fisheye.*

4. KESIMPULAN DAN SARAN

Pada penelitian ini telah dilakukan klasifikasi objek berbasis *deep learning* untuk klasifikasi objek pada gambar *fisheye*. Adapun hasilnya dapat disimpulkan sebagai berikut:

1. Klasifikasi *object* berhasil dilakukan pada *object* manusia menggunakan kamera *fisheye* dengan arsitektur CNN. *Layer* yang digunakan terdiri dari 1 *layer*, 2 *layer*, 3 *layer*, dan 7 *layer*, dengan *convolution layer, maxpooling* dan fungsi aktivasi yang digunakan yaitu RELU, serta menggunakan *fully connected layer* untuk output klasifikasi.
2. Kinerja model *deep learning* untuk klasifikasi objek menggunakan kamera *fisheye* berhasil dilakukan, mendapatkan hasil bahwa ukuran *batch size* dan ukuran gambar untuk proses akan menentukan lama waktu *training.* Kemudian hasil pengujian menunjukkan bahwa pada 7 *layer* CNN tanpa reduksi *overfitting* mendapatkan hasil terbaik untuk klasifikasi objek dengan hasil *presisi* 98,56%, *recall* 98,5%, dan akurasi 98,59%. Sementara pada 3 *layer* CNN dengan reduksi *overfitting* menunjukkan hasil yang kurang baik dalam mengklasifikasi objek dengan hasil *presisi* 21,23%, *recall* 32,73%, dan akurasi 32,77%. Dengan demikian ketika menggunakan jenis gambar *fisheye*, proses reduksi *overfitting* tidak diperlukan lagi, karena berpengaruh pada hasil pengujian model klasifikasi.

Pada penelitian ini memiliki saran, adapun saran untuk mengembangkan penelitian ini, yaitu:

1. Sistem klasifikasi *object* dapat dikembangkan lagi dengan menggunakan lebih banyak dataset dan pengambilan dataset dapat dilakukan didalam dan diluar ruangan agar dapat mengklasifikasi *object* lebih banyak.
2. Sistem klasifikasi objek dapat dikembangkan lagi dengan mengasumsikan lebih dari satu objek, serta dapat memperbesar ukuran gambar dan jumlah layer pada CNN, supaya mendapatkan hasil klasifikasi yang lebih baik dan menambah tingkat akurasi.

DAFTAR PUSTAKA

ALAMSYAH, D. AND PRATAMA, D. 2020 'Implementasi *Convolutional Neural Networks* (CNN) Untuk Klasifikasi Ekspresi Citra Wajah Pada Fer-2013 dataset', Jurnal Teknologi

Informasi, 4(2), pp. 350–355. doi:10.36294/jurti.v4i2.1714.

DZAKY, A.T.R. 2021 'Deteksi Penyakit Tanaman Cabai Menggunakan Metode *Convolutional Neural Network'*, *e-Proceeding of Engineering*, 8(2), pp. 3039–3055.

GUNAWAN, D. and SeETIAWAN, H. 2022 '*Convolutional Neural Network* Dalam Citra Medis', KONSTELASI: Konvergensi Teknologi dan Sistem Informasi, 2(2). doi:10.24002/konstelasi.v2i2.5367.

HERDIANTO, H. and NASUTION, D. 2023 'Implementasi Metode CNN Untuk Klasifikasi objek', METHOMIKA Jurnal Manajemen Informatika dan Komputerisasi Akuntansi, 7(1), pp. 54–60. doi:10.46880/jmika.vol7no1.pp54-60.

LEE, S.-H. 2022 '*A study on classification and detection of small moths using CNN model*', Computers, Materials & Continua, 71(1), pp. 1987–1998. doi:10.32604/cmc.2022.022554..

PUTRI, O.N. 2020 Implementasi Metode CNN dalam Klasifikasi Gambar Jamur pada Analisis *Image Processing* (Studi Kasus: Gambar Jamur dengan *Genus Agaricus* dan *Amanita*). thesis.

SUTEJO, M. F., SATYAWAN, A. S. and SISWANTI, S. D. 2022 "*Face Classification of in 360 Degree Images (Fish Eye) Using Tensorflow*", *Prosiding Seminar Nasional Sains Teknologi dan Inovasi Indonesia (SENASTINDO)*, 4, pp. 365-375. doi: 10.54706/senastindo.v4.2022.213.

KUSUMANINGRUM, T.F. 2018 Implementasi *Convolution Neural Network* (CNN) Untuk Klasifikasi Jamur Konsumsi Di Indonesia Menggunakan Keras (Studi Kasus : Jamur Kuping, Jamur Merang dan Jamur Tiram). thesis.

WANG, PIN, FAN, E. and WANG, PENG 2021 '*Comparative analysis of image classification algorithms based on traditional machine learning and Deep Learning', Pattern Recognition Letters*, 141, pp. 61–67. doi:10.1016/j.patrec.2020.07.042.

Halaman ini sengaja dikosongkan.