DETEKSI BENTUK WAJAH MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

Irfan Maulana, Nabila Khairunisa, Ratna Mufidah
Informatika, Universitas Singaperbangsa Karawang
Jl. HS. Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361
2010631170013@student.unsika.ac.id

ABSTRAK

Deteksi bentuk wajah menjadi elemen kritis dalam berbagai aplikasi pengenalan wajah, keamanan, dan interaksi manusia dengan sistem komputer. Deteksi tersebut dapat dilakukan dengan menggunakan ekstraksi fitur-fitur tertentu. Pada penelitian ini menggunakan metode arsitektur CNN yang telah terbukti dalam mengatasi kompleksitas visual untuk mengekstraksi fitur wajah secara hierarkis. Dataset pelatihan yang beragam digunakan untuk melatih model, meningkatkan kemampuannya dalam mengenali variasi bentuk wajah. Hasilnya menunjukkan bahwa CNN memberikan tingkat akurasi yang cukup tinggi dalam mendeteksi bentuk wajah bahkan dalam situasi yang challenging. Selain itu, kecepatan dan efisiensi model memungkinkan implementasi real-time dalam berbagai aplikasi, memperkuat potensinya dalam mendukung pengembangan teknologi pengenalan wajah yang lebih canggih dan andal. Penelitian ini memberikan kontribusi pada pemahaman mendalam tentang penerapan CNN untuk deteksi bentuk wajah serta memberikan dasar untuk peningkatan lebih lanjut dalam bidang pengenalan wajah dan aplikasi terkait. Dataset yang digunakan berjumlah 5000 dataset dengan resolusi 100 x 100 pixels. Berdasarkan hasil penelitian yang telah dilakukan dengan melakukan epoch sebanyak 40 kali *dan batch size* berukuran 16, didapatkan hasil akurasi *training* tertinggi sebesar 74%. Pada pengujian, model juga diimplementasikan ke dalam API menggunakan *framework* Flask.

Kata kunci : *Wajah, Deep Learning, Convolutional Neural Network*

1. PENDAHULUAN

Wajah merupakan salah satu bagian tubuh yang paling penting bagi manusia. Wajah menentukan identitas seseorang, memainkan peran penting dalam komunikasi dan interaksi sosial, dan menentukan seberapa menarik seseorang [1]. Pada era perkembangan teknologi informasi dan komunikasi saat ini, pengolahan citra dan pengenalan pola telah menjadi bagian integral dari berbagai aplikasi kehidupan sehari-hari. Salah satu aspek yang menarik perhatian adalah deteksi bentuk wajah menggunakan teknik *Convolutional Neural Network* (CNN) [2]. Deteksi bentuk wajah merupakan cabang penting dalam pengolahan citra yang memiliki implikasi luas dalam berbagai bidang, seperti keamanan, pengenalan wajah, pengembangan permainan, dan interaksi manusia-mesin [3].

CNN merupakan salah satu terobosan utama dalam bidang pembelajaran mendalam (*deep learning*) yang telah membuktikan kinerja luar biasa dalam berbagai tugas pengolahan citra dan salah satu kelebihannya yaitu mempunyai *feature learning* [4]. Teknik ini secara efektif dapat mengekstraksi fitur-fitur hierarkis dari citra, memungkinkan pengenalan pola yang lebih akurat dan kompleks. Keberhasilan CNN dalam tugas-tugas seperti pengenalan objek, segmentasi citra, dan klasifikasi telah mengilhami penelitian lebih lanjut dalam penerapan deteksi bentuk wajah [5]

Dalam konteks deteksi bentuk wajah, tantangan utama adalah merancang model yang dapat mengenali variasi yang luas dalam ekspresi wajah, rotasi kepala, pencahayaan yang berbeda, dan variasi lainnya [6]. Penelitian sebelumnya telah mengusulkan berbagai pendekatan untuk mengatasi tantangan ini, namun masih ada ruang untuk eksplorasi lebih lanjut dalam upaya meningkatkan akurasi dan keandalan deteksi bentuk wajah [5].

Oleh karena itu, penelitian ini bertujuan untuk melakukan deteksi bentuk wajah menggunakan CNN. Studi ini diharapkan dapat memberikan wawasan yang lebih mendalam tentang kelebihan dan keterbatasan metode CNN dalam menghadapi variasi yang kompleks dalam citra wajah. Dengan pemahaman yang lebih baik tentang kinerja relatif dari berbagai pendekatan, pengembang dan peneliti dapat membuat keputusan yang lebih terinformasi dalam memilih teknik yang sesuai dengan tujuan dan lingkup aplikasi mereka.

Secara garis besar, penelitian ini akan mengambil pendekatan eksperimental dengan menggunakan dataset wajah yang representatif dan mencakup matrik evaluasi yang beragam. Analisis mendalam akan dilakukan terhadap hasil eksperimen untuk menggambarkan kekuatan dan kelemahan masing-masing metode. Selain itu, studi ini juga diharapkan dapat menjadi landasan bagi pengembangan teknik deteksi bentuk wajah yang lebih unggul di masa depan [7].

Pada akhirnya, pemahaman yang lebih dalam tentang deteksi bentuk wajah menggunakan CNN dapat memberikan kontribusi positif bagi perkembangan sistem pengenalan wajah yang lebih canggih, aplikasi keamanan yang lebih andal, serta pengalaman interaksi manusia-mesin yang lebih mendalam. Dengan demikian, penelitian ini memiliki

relevansi yang signifikan dalam konteks perkembangan teknologi informasi dan komunikasi yang terus bergerak maju.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Pada penelitian sebelumnya yang dilakukan oleh [8], implementasi CNN dalam klasifikasi ekspresi wajah dapat dilakukan dengan mengekstraksi fitur-fitur tertentu. Penelitian bertujuan untuk mengetahui mengenai performa CNN dalam mengenali ekspresi wajah pada kondisi data yang kurang ideal. Pada penelitian ini, pada saat menggunakan Adamax optimizer menghasilkan performa paling baik yaitu dengan akurasi sebesar 66% dibanding optimizer Adam, N-Adam, dan SGD.

Adapun penelitian [9] mengenai implementasi *deep learning* menggunakan CNN untuk sistem pengenalan wajah. Dengnan menggunakan *Caffe Deep Learning* untuk mendeteksi wajah dapat menghasilkan akurasi sebesar 100% sesuai kriteria sebelumnya. Sementara metode *Convolutional Neural Network* untuk klasifikasi mencapai akurasi 98%, dengan *precision* 98.4%, *recall* 98%, dan akurasi total 99,84%.

2.2. *Deep Learning*

Deep Learning adalah salah satu cabang dari Machine Learning yang memanfaatkan jaringan neural yang memiliki banyak lapisan (*deep neural network*) untuk menangani permasalahan pemrosesan data yang kompleks. Konsep dasar yang digunakan *oleh Deep Learning* mirip dengan *Machine Learning* pada umumnya, di mana model dilatih menggunakan data untuk melakukan prediksi atau klasifikasi. Perbedaannya terletak pada keberadaan lapisan-lapisan kompleks dalam *neural network* [10]. Keunggulan utama *Deep Learning* terletak pada kemampuannya dalam menyelesaikan tugas yang kompleks dan fleksibilitas arsitekturnya untuk beradaptasi dengan mudah terhadap permasalahan baru. Meskipun demikian, terdapat beberapa kelemahan, seperti kebutuhan akan jumlah data yang besar, proses pelatihan yang memakan waktu, dan risiko *overfitting*.

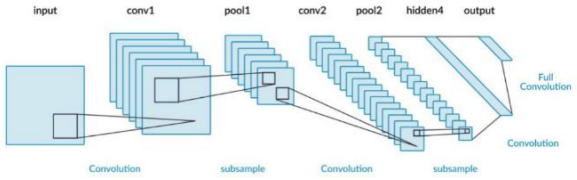
2.3. Citra Digital

Citra digital merujuk pada representasi gambar yang tersimpan dalam format digital, di mana setiap elemen gambar direpresentasikan oleh piksel [5]. Deteksi wajah dalam citra digital adalah suatu teknik atau proses yang bertujuan untuk mengidentifikasi dan menemukan lokasi wajah manusia dalam suatu gambar digital. Proses deteksi wajah pada citra digital melibatkan beberapa tahap atau langkah, dan salah satu pendekatan yang umum digunakan adalah menggunakan algoritma *computer vision* dan *machine learning*.

2.4. *Convolutional Neural Network* (CNN)

CNN adalah salah satu jaringan saraf tiruan *deep feedforward* yang luas diterapkan pada visi komputer. Metode ini juga dikenal sebagai ConvNet. Mirip dengan jaringan syaraf tiruan (JST), CNN memiliki arsitektur yang terdiri dari *node* atau *neuron* yang saling terhubung. *Neuron* yang saling terhubung satu sama lain dalam sebuah lapisan [11].

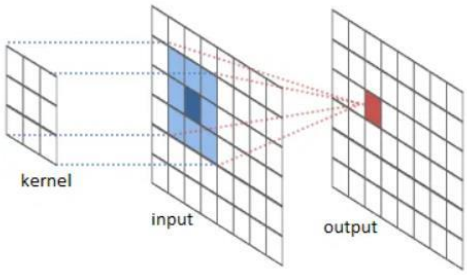
Penggunaan CNN secara luas dalam visi komputer visi komputer ditunjukkan oleh banyak arsitektur yang telah dikembangkan untuk sejumlah masalah. Beberapa arsitektur yang umum digunakan adalah AlexNet, VGG, Google Inception, dsb. Sementara itu arsitektur CNN itu sendiri terdiri dari Lapis tersembunyi (*Hidden Layer*) yang berisi *convolutional layers*, *pooling layers*, *normalization layers*, *ReLu layer*, *fully connected layers* dan *loss layer* [12]. Secara umum proses dari CNN dapat dilihat pada Gambar 1.



Gambar 1. Proses *convolutional neural network* (CNN)

2.4.1. *Convolutional Layer*

Lapisan yang pertama kali menerima input gambar langsung pada arsitektur. Operasi pada lapisan ini sama dengan operasi konvolusi, yaitu melakukan operasi *filter* kombinasi linier pada wilayah lokal. Filter ini merupakan representasi dari bidang reseptif dari *neuron-neuron* yang terhubung ke dalam lokal (konektivitas lokal) pada citra masukan. Bentuk dari direpresentasikan sebagai volume BxKxL atau lapisan Lapisan berukuran BxK dengan jumlah lapisan sebanyak L. Lapisan konvolusi memiliki hyperparameter dan parameter [2]. Contoh dari *convolutional layer* dapat dilihat pada Gambar 2.

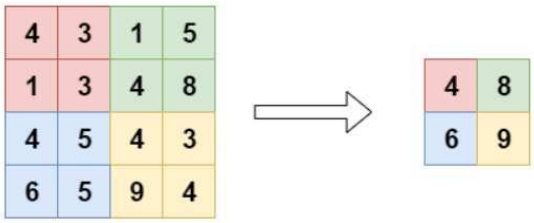


Gambar 2. *Convolutional layer*

2.4.2. *Pooling Layer*

Adalah bagian yang digunakan untuk mengurangi ukuran gambar (memeriksa ke bawah) yang berencana untuk bekerja dengan prosedur konvolusi pada lapisan berikut. Interaksi ini harus dimungkinkan dengan beberapa prosedur, termasuk

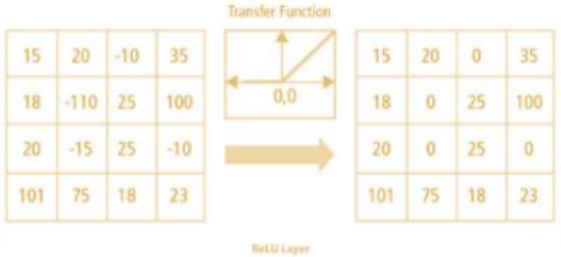
penggabungan maksimal (mengambil nilai terbesar dari sub-gambar), *normal pooling* (mengambil nilai normal dari sub-gambar), dan sebagainya. *pooling* (mengambil nilai normal dari sub gambar), dan seterusnya [13]. Sub gambar yang digunakan dalam kegiatan ini adalah 8x8 yang dapat dilihat pada contoh Gambar 3.



Gambar 3. *Pooling layer* (*max pooling*)

2.4.3. *ReLu Layer*

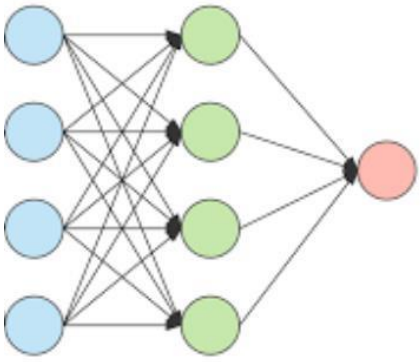
Lapisan *Redressed Straight Unit* (ReLu) adalah lapisan yang berencana untuk bekerja pada *non linieritas* dari kemampuan pilihan. Kemampuan inisiasi yang digunakan adalah f(x) = max(0, x). Tujuannya yaitu mengubah semua *input* negatif menjadi nol [1]. Untuk *ReLu layer* sendiri dapat dilihat pada Gambar 4.



Gambar 4. Relu layer

2.4.4. *Fully Connected Layer*

Pada lapisan ini semua neuron sepenuhnya terkait (*fully connected*) seperti pada *multi-facet perceptron* (MLP) dan *multi-facet perceptron* (MLP). Pada bagian ini terdapat muatan dan kecenderungan yang digunakan untuk mendemonstrasikan informasi yang telah diselesaikan oleh MLP [1]. Berikut *Fully connected layer* pada Gambar 5.



Gambar 5. *Fully connected layer*

2.4.5. *Loss Layer*

Lapisan ini adalah lapisan terakhir dalam CNN yang digunakan untuk memutuskan hukuman untuk hasil yang tidak sesuai dengan nama (target) dari siklus persiapan (penentuan bobot dan bias). Terdapat beberapa fungsi yang dapat digunakan dalam lapisan ini diantaranya *sigmoid cross-entropy loss, softmax loss, euclidean loss* dsb [12].

2.5. *TensorFlow*

TensorFlow dirancang khusus untuk keperluan pengembangan dan pelatihan model *machine learning* dan *deep learning*. *TensorFlow* memberikan dukungan yang luas untuk berbagai jenis tugas dalam domain *machine learning*, seperti klasifikasi, regresi, pengenalan pola, dan lainnya. Dalam konteks deteksi wajah *tensorFlow* menyediakan alat dan fungsi yang kuat untuk membangun, melatih, dan menerapkan model CNN untuk tugas deteksi wajah. Penggunaan *TensorFlow* dalam deteksi wajah dengan CNN memberikan fleksibilitas dan kekuatan dalam mengembangkan solusi deteksi wajah yang andal dan efisien. Dengan berbagai alat dan sumber daya yang disediakan oleh *TensorFlow* dapat berfokus pada implementasi dan evaluasi model deteksi wajah dengan lebih efisien [14].

2.6. *Framework* Flask

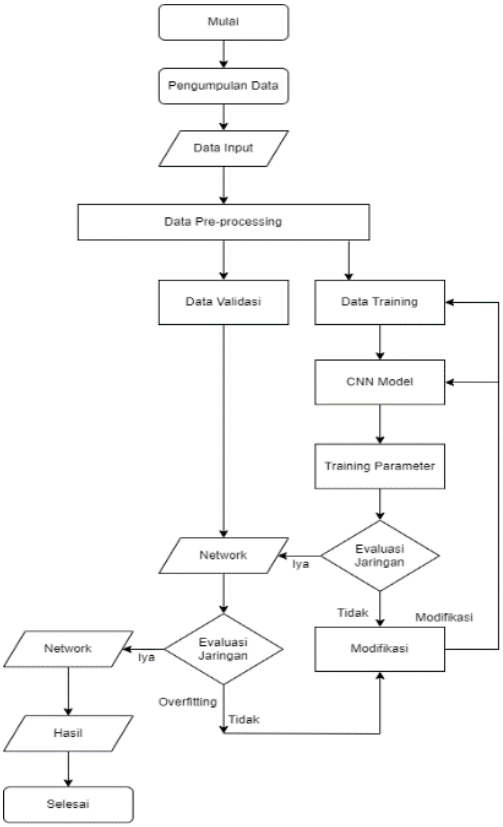
Flask adalah suatu kerangka kerja web yang dipakai dalam pemrograman Python. Flask juga menyajikan perpustakaan dan serangkaian kode program yang bisa dimanfaatkan untuk membuat sebuah situs tanpa perlu membuat semuanya dari awal. Kelasifikasinya sebagai *micro-framework* karena tidak mengharuskan penggunaan alat atau perpustakaan tertentu dan telah dilengkapi dengan basis data bawaan [15]. Pengembangan dengan Flask melibatkan Jinja Template Engine dan Werkzeug WSGI ToolKit. Flask dibagi menjadi dua kategori, yaitu *Static File* dan *Template File*. Pada proses pengembangan aplikasi menggunakan *framework* ini, penggunaan *virtual environment* diperlukan untuk menampung pustaka yang akan dipakai. Meskipun ringan karena memiliki inti yang sederhana, Flask menonjolkan sifat kesederhanaan dan fleksibilitas yang memungkinkannya dikembangkan dan disesuaikan dengan kebutuhan melalui penambahan ekstensi yang diperlukan [15].

3. METODE PENELITIAN

Sistem pengenalan wajah manusia dilakukan melalui beberapa tahapan sebelum mendapatkan keluaran yang diinginkan basis data. Sebuah input gambar wajah dimodifikasi menjadi agar memperjelas wajah dan membuat latar belakang gambarnya menjadi gelap, hasil dari modifikasi inilah yang kemudian diproses lebih lanjut untuk digunakan melatih basis data.

Secara umum, sistem ini terdiri dari beberapa tahap termasuk pengumpulan data yang berupa gambar wajah, pemrosesan gambar, ekstraksi fitur, klasifikasi dan evaluasi sistem. Berikut ini adalah

diagram alir dari desain sistem dapat dilihat pada Gambar 5.

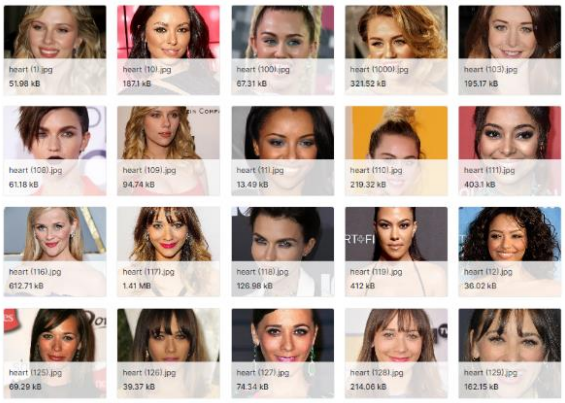


Gambar 5. Diagram alur sistem pengenalan wajah

Metodologi penelitian yang digunakan terdiri dari beberapa tahapan yaitu:

3.1. Pengumpulan Data

Dataset yang digunakan pada penelitian ini adalah Face Shape Dataset yang bersumber dari Kaggle yaitu salah satu dataset yang bertujuan untuk memberikan data beruba kumpulan gambar wajah dengan berbagai bentuk kategorinya seperti *Heart, Oblong, Oval, Square dan Circle*. Dataset ini dirilis pada tahun 2019 dan berisikan data gambar berjumlah 5000 data gambar. Pada masing-masing kategori berisi 1000 gambar dengan pembagian 800 gambar untuk *training* dan 200 gambar sisanya untuk *validation*. Gambar 6. menunjukkan mengenai pengumpulan data.



Gambar 6. Face Shape Dataset

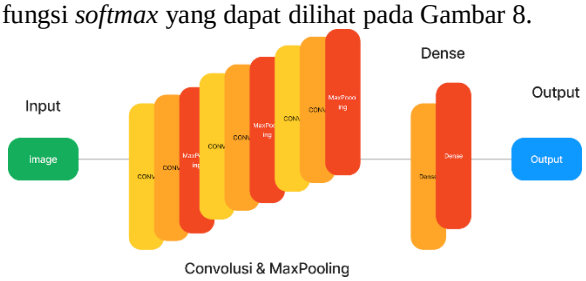
3.2. *Data Pre-processing*

Data pre-processing dalam penelitian ini antara lain melakukan cropping pada data gambar menjadi 100 x 100 pixel kemudian melakukan *Standardization* Image agar semua data mempunyai dimensi panjang dan lebar yang sama dan terakhir data yang sudah dikumpulkan dilakukan penyesuaian kecerahan untuk latar belakang dan memperjelas bagian bentuk wajah. Berikut hasil *pre-processing data* pada Gambar 7.

Gambar 7. Hasil dari Data Pre-processing

3.3. Perancangan Sistem

Merancang sistem yang dibuat dengan arsitektur CNN Arsitektur CNN yang terdiri dari 6 lapisan konvolusi dengan masing-masing fungsi aktivasi ReLu dan *max pooling* pada lapisan *pooling*. Sementara itu terdapat 3 lapisan dense dengan fungsi aktivasi ReLu dan *softmax*. Terdapat satu *output* dari model ini dimana kelas ditentukan berdasarkan model ini dimana kelas ditentukan berdasarkan berdasarkan hasil dari fungsi *softmax* yang dapat dilihat pada Gambar 8.



Gambar 8. Arsitektur CNN

3.4. Imlementasi Sistem

Implementasi sistem ini memanfaatkan optimizer jenis Adam untuk mengoptimalkan model *Convolutional Neural Network* (CNN). *Optimizer* Adam adalah salah satu algoritma optimasi yang umum digunakan dalam pelatihan model *machine learning*.

Selama proses pelatihan model CNN, data yang digunakan untuk melatih model dibagi menjadi dua set utama yaitu data *training* dan *validation*. Data *training* digunakan untuk melatih model, sementara data *validation* digunakan untuk mengukur performa model secara objektif dan menghindari *overfitting*. Pada setiap iterasi *training*, yang disebut sebagai *epoch*, model diperbarui menggunakan data *training*. Selanjutnya, performa model dievaluasi menggunakan data *validation* untuk memastikan bahwa model tidak hanya menghafal data pelatihan tetapi juga dapat menggeneralisasi dengan baik untuk data baru. Proses

pelatihan dimulai dengan 10 *epoch* dan berlanjut hingga 40 *epoch*. *Epoch* adalah satu kali iterasi melalui seluruh dataset pelatihan. Dengan memulai dari 10 *epoch*, model mulai belajar dari data pelatihan, dan melalui setiap *epoch* berikutnya, diharapkan model dapat meningkatkan kemampuannya untuk mengenali pola dan fitur dalam data.

Dengan pendekatan pelatihan yang berurutan dan penggunaan optimizer Adam, implementasi sistem ini bertujuan untuk menghasilkan model CNN yang optimal dan mampu menggeneralisasi dengan baik terhadap data yang belum pernah dilihat sebelumnya.

3.5. Evaluasi

Data yang digunakan untuk melatih model dibagi menjadi dua bagian utama yaitu training data dan validation data. Setiap model menjalani satu *epoch* selama proses pelatihan, *training data* dan *validation data* digunakan untuk mengukur dan memperbarui performa model. *Training data* yang berukuran 100 x 100 pixels dimasukkan ke dalam *input layer* dari model CNN. *Input layer* merupakan lapisan pertama dari model dan menerima citra berukuran 100 x 100 sebagai input. Proses ini merupakan tahap awal dari aliran data dalam model. Selanjutnya, dilakukan proses konvolusi pada data yang telah dimasukkan ke dalam input layer. Proses konvolusi ini melibatkan filter atau kernel yang bergerak melintasi citra input untuk mengekstrak fitur-fitur penting. Setelah proses konvolusi, hasilnya biasanya akan melalui lapisan aktivasi, seperti ReLU (*Rectified Linear Unit*), untuk memperkenalkan non-linearitas ke dalam model.

Dengan demikian, proses dimulai dengan memasukkan *training data* berukuran 100 x 100 pixels ke dalam input layer, diikuti oleh proses konvolusi untuk mengekstrak fitur-fitur penting dari data tersebut. Proses ini merupakan bagian awal dari pembelajaran model CNN selama satu epoch pada proses pelatihan.

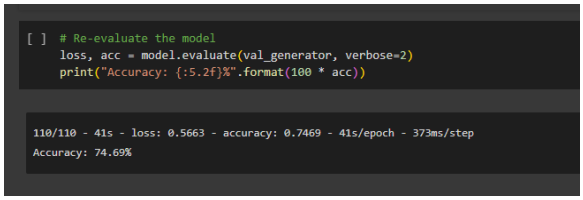
4. HASIL DAN PEMBAHASAN

4.1. Implementasi CNN

Implementasi *Convolutional Neural Network* (CNN) melibatkan tiga tahap utama diantaranya yaitu *training*, *validation*, dan *testing*. Pada tahap training, jaringan dilatih menggunakan data input untuk mempelajari pola dan fitur yang ada dalam dataset. Proses ini melibatkan penyesuaian parameter jaringan agar dapat membuat prediksi yang semakin akurat. Setelah pelatihan, jaringan diuji pada data validasi untuk mengevaluasi kinerjanya di luar data training. Evaluasi ini membantu mengidentifikasi apakah model tersebut telah memahami pola dengan baik atau justru terlalu disesuaikan dengan *data training* (*overfitting*). Jika hasil pada data validasi memuaskan, langkah terakhir adalah menguji jaringan pada data testing yang belum pernah dilihat sebelumnya. Performa pada tahap testing memberikan gambaran seberapa baik jaringan dapat menggeneralisasi dan klasifikasi data yang baru. Keseluruhan proses ini memastikan bahwa CNN tidak hanya mampu mempelajari *data training* tetapi juga dapat menghasilkan prediksi yang akurat pada data baru yang belum pernah dilihat sebelumnya.

4.2. *Data Training*

Dalam proses training CNN, digunakan sebanyak 80% dari keseluruhan data sebagai *data training*, yang setara dengan 4000 gambar dengan resolusi 100 x 100. Setiap kelas memiliki 800 sampel gambar untuk memberikan representasi yang seimbang dalam pelatihan model. Dengan proporsi ini, model akan mempelajari pola dan karakteristik dari dataset secara luas. Penggunaan parameter ini pada tahap pelatihan memastikan bahwa model dapat belajar dengan baik dari variasi yang mencakup setiap kelas, sehingga meningkatkan kemampuannya untuk mengklasifikasikan data yang baru dan belum pernah dilihat sebelumnya. Pada tahap training di dapatkan akurasi pelatihan tertinggi yaitu sebesar 74% dimana dapat dilihat pada gambar 9.



Gambar 9. Hasil Akurasi Proses Training

4.3. *Data Validation*

Proses validasi jaringan dilakukan dengan menggunakan set data berjumlah 1000, di mana setiap kelasnya terdiri dari 200 sampel gambar. Penting untuk dicatat bahwa data ini tidak digunakan dalam proses pelatihan jaringan sebelumnya, sehingga jaringan diuji pada informasi yang belum pernah dilihatnya sebelumnya. Tingkat akurasi yang cukup tinggi ini memberikan indikasi bahwa model dapat menggeneralisasi dengan baik ke data yang tidak terlibat dalam proses pelatihan, menunjukkan kemampuannya untuk mengenali pola dan fitur yang relevan.

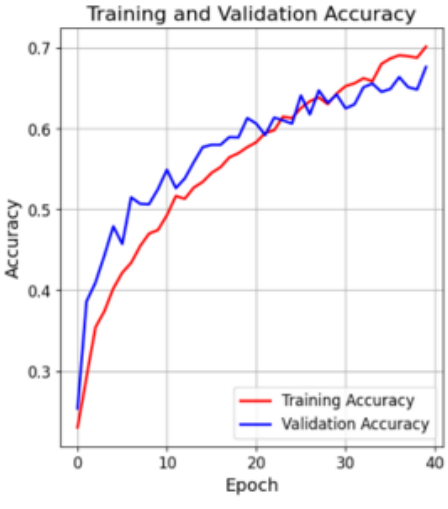
4.4. *Data Testing*

Interpreter memiliki kemampuan untuk memasukkan sampel data yang ingin diklasifikasikan oleh jaringan. Jaringan kemudian mengeluarkan label jenis tanaman berdasarkan karakteristik data yang dimasukkan. Hasil klasifikasi yang diberikan oleh jaringan dapat menjadi panduan bagi interpreter dalam mengidentifikasi jenis tanaman yang sulit dibedakan secara visual. Hasil *testing* menunjukkan kemampuan model untuk memberikan label klasifikasi dengan tingkat keakuratan yang baik pada data yang belum pernah dilihat sebelumnya.

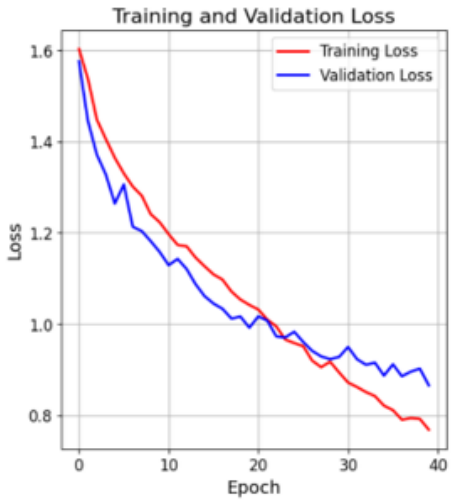
4.5. Pelatihan Model dan Akurasi

Hasil training pada penelitian ini menggunakan *learning rate* sebesar 0.0001 dengan *epoch* sebanyak 40 kali dan untuk *batch size* berukuran 16. Dari hasil

training yang dilakukan, tingkat pembelajaran pada model yang dibangun menghasilkan nilai akurasi tertinggi sebesar 74 %. Dibawah ini terdapat grafik nilai akurasi dan loss pada gambar 9.



(a)



(b)

Gambar 9. Grafik Accuracy dan Loss

Grafik akurasi pada Gambar 9 (a) mengalami fluktuasi selama proses pelatihan karena model sedang belajar fitur atau data yang ada dalam citra. Pada tahap awal pelatihan, akurasi rendah karena model belum menemukan pola yang tepat untuk memprediksi hasil, sehingga sering memberikan prediksi yang salah. Seiring waktu (*epoch*), model mulai menemukan pola yang lebih akurat, meningkatkan kemampuannya dalam memprediksi hasil. Namun, setelah mencapai puncak, akurasi mulai menurun karena model telah memahami pola yang ada dan tidak dapat menemukan pola yang lebih baik, menunjukkan stabilnya performa. Gambar 9 (b) menunjukkan bahwa awal pelatihan memiliki nilai *loss* yang tinggi, namun seiring waktu, nilai *loss* tersebut menurun. Untuk mendapatkan performa yang baik, nilai loss akurasi dan validasi yang mendekati nol diinginkan. Namun, nilai *loss* yang terlalu rendah dapat mengindikasikan *overfitting*, di mana model terlalu spesifik untuk data pelatihan dan kurang mampu menangani data baru dengan baik.

4.6. Pengujian

4.6.1. Imlementasi Model

Pada tahap pengujian, model yang telah melalui proses pelatihan akan diintegrasikan dan diimplementasikan ke dalam API menggunakan *framework* Flask. Setelah proses integrasi selesai, hasilnya akan berupa sebuah API yang memberikan respon dalam format JSON. Proses pengujian ini mencakup penerapan Flask untuk menyatukan model ke dalam lingkungan API. Sebagai hasilnya, pengguna dapat berinteraksi dengan model melalui API tersebut, yang kemudian memberikan respons dalam bentuk JSON.

4.6.2. Implementasi *Framework* Flask

a. Kode Project Flask

Pada project flask umumnya terdiri dari beberapa file dan direktori yaitu file utama (app.py atau sejenisnya), direktori templates, dan direktori static, yang dapat dilihat pada gambar 10 berikut:

```python
from flask import Flask, request
import tensorflow as tf
import base64
from keras.models import load_model
import numpy as np

def load_image_from_base64(base64_string, target_size=(100, 100)):
    img_bytes = base64.b64decode(base64_string)
    img = tf.io.decode_image(img_bytes, channels=3)
    img = tf.image.resize(img, target_size)
    img = img / 255.0
    img = tf.expand_dims(img, axis=0)
    return img

def load_model(model_path):
    model = tf.keras.models.load_model(model_path)
    return model

def predict_image(model, img):
    pred = model.predict(img)
    return pred[0] # Its an array within array, hence we need to extract it

def classify_face_shape(value):
    shapes = ['circle', 'heart', 'oblong', 'oval', 'square', 'triangle']
    probabilities = value.tolist()

    # Get the sorted probabilities array that would sort the probabilities in descending order
    sorted_probabilities_array = np.argsort(probabilities)[::-1]

    # Get the highest and second highest probabilities
    highest_probability = probabilities[sorted_probabilities_array[0]]

    # Get the corresponding shapes using the sorted probabilities array
    highest_shape = shapes[sorted_probabilities_array[0]]

    return {
        'shape': highest_shape,
        'probability': highest_probability,
    }


app = Flask(__name__)

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Blank</p>"

@app.route('/face_shape', methods=['POST'])
def process_json():
    content_type = request.headers.get('Content-Type')
    if (content_type == 'application/json'):
        param = request.json["image"]

        model = load_model("./model.h5")
        img = load_image_from_base64(param)
        pred = predict_image(model, img)
        result = classify_face_shape(pred)

        return result
    else:
        return 'Content-Type not supported!'
```

Gambar 10. Tampilan kode

b. Running Project Flask

Setelah membuat kode untuk project Flask, langkah selanjutnya adalah menjalankan aplikasi tersebut. Dalam banyak kasus, proyek Flask dapat dijalankan di localhost port 5000 dengan menggunakan perintah di terminal atau

command prompt yang dapat dilihat pada gambar 11.

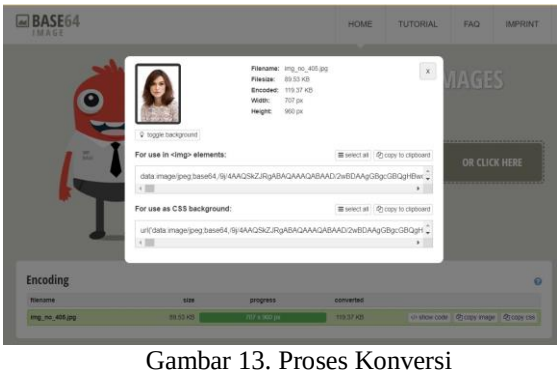
Gambar 11. Running Project

c. Pengujian Data

Pada pengujian data menggunakan foto yang memiliki bentuk wajah bulat atau *circle* seperti gambar 12.

Gambar 12. Data Uji

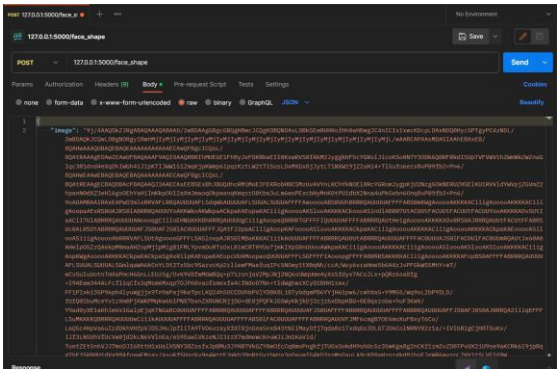
Sebelum melakukan pengujian, foto tersebut perlu di convert kedalam format base64. Dimana format base64 adalah representasi encoding teks biner menggunakan sebuah set karakter ASCII. Perlu menggunakan file tersebut dikarenakan pada saat pengujian kita mengirimkan file lewat API yang sudah dibuat dengan flask harus menggunakan format base64. Proses konversi dapat dilihat pada gambar 13.



Gambar 13. Proses Konversi

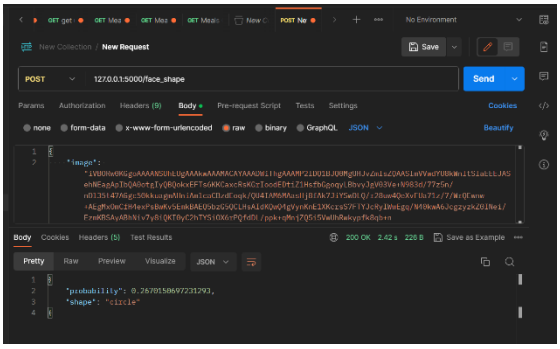
d. Hasil Pengujian

Untuk pengujian, setelah gambar diubah ke format Base64, kita menggunakan alat Postman untuk berinteraksi dengan API. Kurang lebih, struktur permintaan *(request*) akan seperti ini: terdapat permintaan dalam bentuk JSON dengan pasangan kunci 'image' dan nilai 'base64image'. Metodenya menggunakan POST dan diarahkan ke alamat localhost 127.0.0.1:5000/face_shape, yang dapat dilihat pada gambar 14.



Gambar 14. Proses Interaksi dengan API

Setelah mengirmkan permintaan yang sudah dijelaskan sebelumnya dan permintaan tersebut berhasil, makan respon dari API nya sendiri seperti gambar 15 berikut.



Gambar 15. Respon dari API

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan, deteksi bentuk wajah menggunakan CNN dengan Optimizer Adam telah berhasil diimplementasikan. Dataset yang digunakan berjumlah 5000 dataset dengan resolusi 100 x 100 pixels. Berdasarkan hasil penelitian yang telah dilakukan dengan melakukan epoch sebanyak 40 kali dan batch size berukuran 16, didapatkan hasil akurasi *training* tertinggi sebesar 74%. Metode ini menunjukkan kemampuan untuk mengenali dan mengklasifikasikan bentuk wajah secara efisien berdasarkan citra digital. Penerapan CNN pada deteksi bentuk wajah memungkinkan sistem untuk belajar fitur-fitur kompleks dari data citra, sehingga dapat membedakan antara berbagai bentuk wajah dengan tingkat akurasi yang tinggi. Dalam uji coba yang telah dilakukan, model CNN mampu mengatasi variasi dalam warna, tekstur, bentuk dan ukuran wajah yang memberikan hasil konsisten dan handal. Pada pengujian, model juga diimplementasikan ke dalam API menggunakan *framework* Flask. Dimana pada gambar yang akan diuji perlu dilakukan konversi kedalam format base64 dan menggunakan Postman sebagai alat untuk berinteraksi.

Terdapat beberapa saran setelah melakukan penelitian ini yaitu dapat meningkatkan keandalan model dengan perluasan dataset yang lebih luas, melibatkan berbagai kondisi dan variasi bentuk wajah.

Selain itu, disarankan untuk melakukan *fine-tuning* model dengan teknik *transfer learning*, mengoptimalkan parameter secara sistematis, dan mempertimbangkan integrasi dengan teknologi sensor lain, seperti sensor suara atau sensor kimia, guna meningkatkan ketepatan deteksi. Pada penelitian selanjutnya, model ini juga dapat diimplementasikan menjadi sebuah aplikasi agar dapat memudahkan pendeteksian bentuk wajah.

DAFTAR PUSTAKA

[1] "Klasifikasi Pengenalan Wajah Menggunakan Masker atau Tidak Dengan Mengimplementasikan Metode CNN (Convolutional Neural Network)."

[2] N. R. Fauziyya, "Metoda Convolutional Neural Network (CNN) untuk Pendeteksi Tangga pada Alat Pemandu Arah bagi Penyandang Tunanetra," *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 8, no. 2, pp. 145–153, Apr. 2021, doi: 10.34010/telekontran.v8i2.4709.

[3] Lia Farokhah, "Perbandingan Metode Deteksi Wajah Menggunakan OpenCV Haar Cascade, OpenCV Single Shot Multibox Detector (SSD) dan DLib CNN," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 3, pp. 609–614, Jun. 2021, doi: 10.29207/resti.v5i3.3125.

[4] M. Yusqi Alfan Thoriq, K. Eka Permana, I. Agustien Siradjuddin, T. Informatika, U. Trunojoyo Madura, and J. Raya Telang Kamal, "DETEKSI WAJAH MANUSIA BERBASIS ONE STAGE DETECTOR MENGGUNAKAN METODE YOU ONLY LOOK ONCE (YOLO)," 2023. [Online]. Available: https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

[5] D. Hardiyanto, D. Anggun Sartika, and I. Artikel, "Optimalisasi Metode Deteksi Wajah berbasis Pengolahan Citra untuk Aplikasi Identifikasi Wajah pada Presensi Digital," *Dyah Anggun Sartika / Setrum*, vol. 7, no. 1, pp. 107–116, 2018.

[6] R. R. Hajar *et al.*, "DETEKSI WAJAH BERBASIS FACIAL LANDMARK MENGGUNAKAN OPENCV DAN DLIB," *Jurnal Teknologi Informasi)*, vol. 5, no. 2, 2021.

[7] T. Cut Al-Saidina Zulkhaidi, E. Maria, P. Studi Teknologi Rekayasa Perangkat Lunak, and P. Pertanian Negeri Samarinda, "Pengenalan Pola Bentuk Wajah dengan OpenCV," *JURTI*, vol. 3, no. 2, 2019.

[8] D. Alamsyah and D. Pratama, "DATASET," *Jurnal Teknologi Informasi*, vol. 4, no. 2, 2020.

[9] N. Dewi and F. Ismawan, "IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CNN UNTUK SISTEM PENGENALAN WAJAH," *Faktor Exacta*, vol. 14, no. 1, p. 34, Mar. 2021, doi: 10.30998/faktorexacta.v14i1.8989.

[10] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning", doi: 10.1007/s12525-021-00475-2/Published.

[11] N. H. Harani, C. Prianto, and M. Hasanah, "Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode Convolutional Neural Network (CNN) Berbasis Python," 2019.

[12] I. Hartono, A. Noertjahyana, and L. W. Santoso, "Deteksi Masker Wajah dengan Metode Convolutional Neural Network."