jurnalteknik.unisla.ac.id/index.php/informatika **ISSN: 2503-071X**
JOUTICA Volume 6 No.2 2021 **E-ISSN: 2621-511X**

PROSES EKSTRAKSI DAN KLASIFIKASI CITRA EMOSI MENGGUNAKAN METODE PCA DAN CNN

Anggi Nur Fadzila[1], Danar Putra Pamungkas[2], Resty Wulanningrum[3]
[1,2,3)]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Nusantara PGRI Kediri
Jln. KH. Achmad Dahlan No.76 Mojoroto Kota Kediri
Email : fadzilaangginur@gmail.com[1)],danar@unpkediri.ac.id[2)],resty0601@gmail.com[3)]

ABSTRAKS
Manusia secara alami menggunakan ekspresi wajah untuk berkomunikasi dan menunjukan emosi mereka dalam berinteraksi sosial. Ekspresi wajah termasuk kedalam komunikasi non-verbal yang dapat menyampaikan keadaan emosi seseorang kepada orang yang telah mengamatinya. Penelitian ini menggunakan metode Principal Component Analysis (PCA) untuk proses ekstraksi ciri pada citra ekspresi dan metode Convolutional Neural Network (CNN) sebagai prosesi klasifikasi emosi, dengan menggunakan data Facial Expression Recognition-2013 (FER-2013) dilakukan proses training dan testing untuk menghasilkan nilai akurasi dan pengenalan emosi wajah. Hasil pengujian akhir mendapatkan nilai akurasi pada metode PCA sebesar 59,375% dan nilai akurasi pada pengujian metode CNN sebesar 59,386%.

Kata kunci : Convolutional Neural Nertwork, CNN, Principal Component Analysis, PCA

ABSTRACT
Humans naturally use facial expressions to communicate and express their emotions in social interactions. Facial expressions are included in non-verbal communication that can convey a person's emotional state to those who have observed it. This study uses the Principal Component Analysis (PCA) method for the feature extraction process in the expression image and the Convolutional Neural Network (CNN) method as an emotion classification process, using Facial Expression Recognition-2013 (FER-2013) data, training and testing processes are carried out to produce value accuracy and facial emotion recognition.The final test results get the accuracy value of the PCA method of 59.375% and the accuracy value of the CNN method of 59.386%.

Keywords: Convolutional Neural Nertwork, CNN, Principal Component Analysis, PCA

1. PENDAHULUAN

Pada umumnya manusia secara alami menggunakan ekspresi wajah untuk berkomunikasi dan menunjukkan emosi mereka dalam berinteraksi sosial. Kondisi emosional seseorang dapat dilihat dari perkataan, gerak tubuh, dan ekspresi wajah. Setiap manusia memiliki ekspresi wajah berbeda beda. Ekspresi wajah adalah dari pergerakan otot-otot pada sekitar wajah.

Ekspresi wajah termasuk kedalam komunikasi non-verbal yang dapat menyampaikan keadaan emosi seseorang. Pada era globalisasi seperti sekarang, deteksi pengenalan ekspresi wajah menjadi solusi untuk mempermudah dalam hal pengelompokkan emosi manusia. Seiring berkembangnya teknologi informasi, kita dapat mengelompokkan jenis emosi manusia dengan mesih, contohnya adalah *Machine Learning.* Pengelompokkan emosi wajah dapat dilakukan dengan berbagai metode, salah satunya menggunakan *deep learning*. *Deep Learning* adalah bagian dari *Machine Learning* yang dapat mempelajari metode komputasinya sendiri. Metode *deep learning* yang saat ini memiliki hasil memiliki hasil yang signifikan dalam sebuah pengenalan citra adalah *Convolutional Neural Network(CNN.*

Penggunaan Principal Component Analysis dan Euclidean Distance untuk Identifikasi Citra Tnada Tngan berhasil melakukan penerapan metode PCA dan memperoleh hasil pengujian menunjukkan tingkat akurasi terbaik pada nilai threshold sebesar 50 – 219 dengan nilai akurasi sebesar 95% [1]. Implementasi *deep learning convolutional neural network* berhasil melakukan klasifikasi citra emosi wajah dengan tingkat akurasi sebesar 65% [2]. Klasifikasi emosi berdasarkan ciri wajah menggunakan CNN dengan rata-rata akurasisebesar 80,75% [3]. Implementasi Metode PCA dan City Block Distance untuk Presensi Mahasiswa Berbasis Wajah telah berhasil melakukan pengujian dengan hasil akurasi rata-rata sebesar 69,86% [4]. Berdasarkan penelitian-penelitian sebelumnya metode *convolutional neural network(CNN)* memiliki tingkat akurasi yang tinggi, oleh karena itu penelitian ini menggunakan metode *principal component analysis (PCA)*untuk mengasilkan nilai ekstraksi ciri dan metode *convolutional neural network(CNN)* untuk proses klasifikasi. Program akan dibangun dengan bahasa pemrograman python. Pada penelitian ini data yang digunakan adalah data Facial Expression Recognition (FER-2013) dengan jumlah data training 28.709 dan data testing 3.589 dengan resolusi 48 x 48 pixel. Penelitian ini bertujuan untuk menerapkan metode *PCA* untuk perhitungan ekstraksi ciri, penerapan metode *CNN* untuk

proses klasifikasi, dan untuk mengetahui nilai akurasi atau tingkat keberhasilan serta pengenalan emosi wajah.

2. METODE PENELITIAN

2.1 Ekspresi

Ekspresi merupakan salah satu bentuk komunikasi non-verbal. Ekspresi dapat menyampaikan kondisi emosi seseorang kepada orang yang telah mengamatinya. Ekspresi wajah mengungkapkan pikiran yang terlintas pada diri seseorang. Pada ekspresi wajah mempunyai peran penting dalam interaksi sosial yang dapat memberikan efek sbesar 55% dari pesan yang akan disampaikan sementara bahasa dan suara masing-masing menyumbang 7% dan 38% [2].

2.2 *Principal Component Analysis (PCA)*

Principal Component Analysis (PCA) merupakan cara **mereduksi** suatu set variabel yang berdimensi tinggi menjadi lebih rendah namun masih mengandung sebagian besar informasi dari data awal. *PCA* adalah cara pengidentifikasian pola pada suatu data, dan selanjutnya data diekstraksi berdasarkan kesamaan dan perbedaannya. Semenjak sulinya menemukan pola pada data yang mempunyai dimensi besar, dimana gambaran grafiuk yang juga besar tidak mencukupi, *PCA* adalah metode yang tepat dalam menganalisadata itu [5]. Sesuai namanya, *PCA* membentuk variabel baru bernama PC. Variabil PC ini akan menggantikan variabel X pada analisis selanjutnya. Konsep *Principal Component Analysis (PCA)* adalah mengelompokkan variabel-variabel yang berkorelasi linear menjadi 1 komponen utama dengan persamaan sebagai berikut:

$$PC_1 = a_{11}X_1 + a_{12}X_2 + \cdots + a_{p1}X_p + \varepsilon_i$$

$$PC_2 = a_{12}X_1 + a_{22}X_2 + \cdots + a_{p1}X_p + \varepsilon_i$$

.

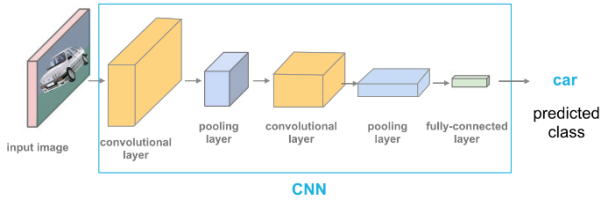
.

$$PC_k = a_{12}X_1 + a_{22}X_2 + \cdots + a_{p1}X_p + \varepsilon_i \quad \ldots(1)$$

Pada persamaan (1) akan mendapatkan nilai PC. Tujuan dilakukannya PCA adalah untuk mereduksi struktur variabel menjadi variabel baru dengan dimensi yang lebih kecil. Variabel baru tersebut mampu menerangkan sebagian besar total varian data dan saling bebas satu sama lain. Selanjutnya variabel baru ini dinamakan principal component (PC). Reduksi dimensi data pada PCA dengan cara mentransformasi variabel-variabel asli yang berkorelasi menjadi satu set variabel baru yang tidak berkorelasi, dengan tetap mempertahankan sebesar mungkin varian yang dapat dijelaskan. [6]

2.3 *Convolutional Neural Network (CNN)*

Convolutional Neural Network (CNN) adalah sebuah metode pengembangan dari *Multilayer Perceptron (MPL)* yang termasuk dalam neural networlk yang bertipe *feed forward* atau tidak berulang[7]. *CNN* didesain untuk mengolah data dua dimensi.*CNN* tidak berbeda jauh dengan *Neural Network* biasanya. *CNN* terdiri dari 3 layer diantaranya input layer, output layer dan hidden layer. Pada hidden layer terdapat banyak layer yang tersusun secara bertumpuk. Layer tersebut diantaranya *convolutional layer*, *pooling layer*, *normalization layer*, *relu layer*, *fully connected layer*, dan yang terakhir adalah *loss layer*.



Gambar 1. Arsitektur CNN

(https://medium.com/@gogoriay/convolutional-neural-network-cnn-1b3f69fcbfa)

Pada Gambar 1 Arsitektur CNN merupakan tahapan pembangunan metode CNN, penjelasan pada gambar 2.1 senagai berikut :

2.3.1 *Convolutioanal Layer*

Convolution Layer melakukan operasi konvolusi pada output dari layer sebelumnya. Layer tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada output fungsi lain 12 secara berulang [8].

2.3.2 *Pooling Layer*

Pooling Layer merupakan lapisan yang menggunakan fungsi dengan feature map sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. Lapisan pooling pada model CNN biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan pooling yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume output pada feature map sehingga jumlah parameter dan perhitungan di jaringan berkurang, serta untuk mengendalikan overfitting[9].

2.3.3 *Fully Connected Layer*

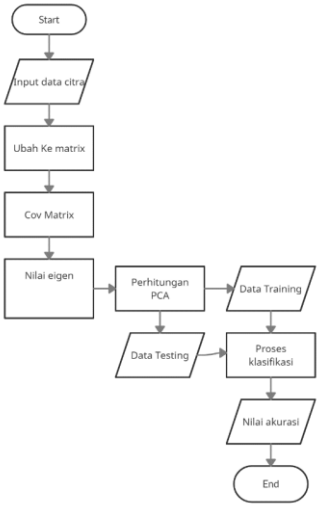
Hasil dari proses konvolusi menjadi input pada fully-connected layerFullyconnected layer adalah lapisan dimana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di fullyconnected layer.

3. PEMBAHASAN

3.1 Diagram Alir Aplikasi

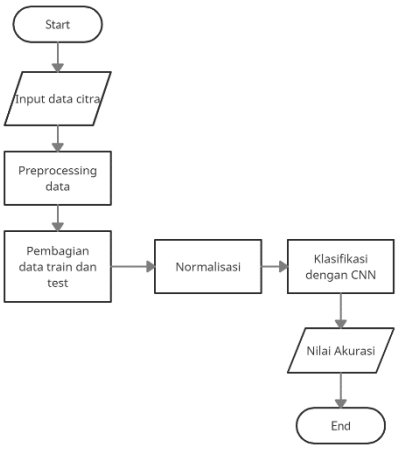
Tahapan untuk metode *principal component analysis* sebagai berikut :

ISSN: 2503-071X
E-ISSN: 2621-511X



Gambar 2. Diagram Alir PCA

Pada gambar 2 dapat dijelaskan sebagai berikut:Langkah pertama yang harus dilakukan adalah menginputkan data citra. Data citra yang dipakai pada penelitian ini menggunakan data FER(2013), dengan 2 emosi yaitu "disgust" dan "surpresed" masing masing berjumlah 100 data. Setelah menginputkan data, ubah data citra ke dalam bentuk matrix, lalu hitung covariance matrix, nilai eigen, eigen vector, eigen value. Setelah itu akan mendapat nilai PC. Lalu lakukan proses training dan testing dengan menggunakan metode yang telah tersedia yaitu KNN dan akan menghasilkan nilai akurasi. tahapan untuk metode CNN sebagai berikut :



Gambar 3. Diagram Alir CNN

Pada gambar 3 dapat dijelaskan sebagai berikut : Langkah pertama yang harus dilakukan adalah menginputkan data, pada penelitian ini menggunakan 5535 data yang terbagi pada 6 emosi yaitu "angry", "happy", "sad", "fear", "disgust", dan "surpresed". Setelah melakukan input data langkah selanjutnya adalah preprocessing data dengan cara menentukan batch size nya. Setelah melakukan preprocessing, langkah selanjutnya membagi data training dan validation, selanjutnya data dinormalisasi dan diklasifikasi dengan metode CNN untuk mendapatkan hasil akurasi atau nilai akurasi.

3.2 Pembahasan

Langkah 1. Pada perhitungan *PCA* dilakukan pengambilan data citra sejumlah 200 dataset FER(2013) dalam dua kelas. Pada penelitian ini hanya mengambil emosi disgust dan surpresed. Berikut adalah contoh dataset FER(2013).

Tabel 1. Dataset pada metode PCA

No	Data Citra	Emosi
1.		Disgust
2.		Disgust
3.		Surpresed
4.		Surpresed

Langkah 2. Setelah menyiapkan data, langkah selanjutnya adalah import library. Pada metode PCA, akan menggunakan library :

```
# membaca library yang dibutuhkan
import numpy as np # library untuk komputasi matriks
import cv2 # library untuk memproses gambar/video
import matplotlib.pyplot as plt # library untuk plot data ( visualisasi )
from sklearn.model_selection import StratifiedShuffleSplit
import os
```

Gambar 4. Library PCA

Seperti pada gambar 4 library PCA dapat dijelaskan :

1. Library numpy digunakan untuk menangani permasalahan angka-angka. Berfokus pada scientific computing, NumPy memiliki kemampuan dalam membentuk objek N-dimensional array.
2. Library Cv2 digunakan untuk memproses gambar/video.
3. Library matplotlib digunakan untuk menampilkan hasil analisis berupa grafik berwarna(visualisasi data)
4. Library Scikit-learn digunakan untuk Machine Learning open source untuk python yang juga bisa digunakan dalam Data Science

Langkah 3. Setelah mengimport library, kemudian mencari nilai rata-rata pada setiap kolom data yang telah diinputkan. Data citra yang digunakan dalam penelitian ini adalah 48 x 48 pixel. Nilai 2304 didapat dari perkalian 48 x 48.

```
def pca(input_ori, dimensi):
    #mencari avg setiap kolom
    avg = np.arange(2304) # array untuk menampun
    print("AVG :", avg)
    for x in range(0, len(avg)):
        avg[x] = 0 #menset semua nilai avg = 0
    # mencari mean untuk setiap kolom sebanyak 7
    print("AVG :", avg)
    for x in range(0, len(input_ori)):
        for y in range(0, len(input_ori[0])):
            avg[y] = avg[y]+input_ori[x][y]
    avg = avg/len(input_ori) #membagi hasil sigm
    print("AVG :", avg)
```

Gambar 5. Mencari rata-rata

ISSN: 2503-071X
E-ISSN: 2621-511X

```
AVG : [   0    1    2 ... 2301 2302 2303]
AVG : [0 0 0 ... 0 0 0]
AVG : [132.315 127.44  120.875 ... 125.855 124.33  126.58 ]
```

Gambar 6. Hasil rata-rata

Langkah 4. Melakukan perhitungan tranpose matriks

```
Xh = input_ori
Xh1 = input_ori
print("XH.shape :", Xh.shape)
print("XH1.shape :", Xh1.shape)
for x in range(0, len(input_ori)):
    for y in range(0, len(input_ori[0])):
        Xh[x][y] = Xh[x][y] - avg[y] # Xh[0][0] -
        # print("Xh[x][y] :", Xh[x][y])
print(Xh.shape)
```

Gambar 7. Tranpose Matriks

```
XH.shape : (200, 2304)
XH1.shape : (200, 2304)
Xh[x][y] : -110.315
Xh[x][y] : -114.44
Xh[x][y] : -106.875
Xh[x][y] : -99.87
Xh[x][y] : -101.785
Xh[x][y] : -56.095
Xh[x][y] : -26.8149999999999998
Xh[x][y] : -5.219999999999999
Xh[x][y] : 30.855000000000004
Xh[x][y] : 33.81
Xh[x][y] : 48.385000000000005
Xh[x][y] : 58.41
Xh[x][y] : 63.165000000000006
Xh[x][y] : 67.69
Xh[x][y] : 67.31
Xh[x][y] : 70.56
Xh[x][y] : 76.97
```

Gambar 8. Hasil Tranpose matriks

Langkah 5. Melakukan perhitungan matriks covariance.

```
covX = Xh.T.dot(Xh)
print("COV dari X :", covX)
```

Gambar 9. matriks Kovarian

```
COV dari X : [[1537143.155 1373302.28  1184139.875 ...  344407.135  414914.21
   476120.46 ]
 [1373302.28  1408145.28  1282472.    ...  315487.76   385515.96
   418275.96 ]
 [1184139.875 1282472.    1334061.875 ...  261107.375  319834.25
   355315.5  ]
 ...
 [ 344407.135  315487.76   261107.375 ... 1276576.795 1196210.57
  1092971.82 ]
 [ 414914.21   385515.96   319834.25  ... 1196210.57  1293106.22
  1258638.72 ]
 [ 476120.46   418275.96   355315.5   ... 1092971.82  1258638.72
  1411312.72 ]]
```

Gambar 10. Hasil matriks kovarian

Langkah 6. Melakukan perhitungan nilai eigen, eigen value dan eigen vector. Dengan kode program seperti berikut :

```
# eigen vector
eig = np.linalg.eig(covX)
print("Nilai Eigen :", eig)
eigVal = np.array(eig[0])
print("Nilai eigen value :", eigVal)
eigVec = np.array(eig[1]) # dapat eigenVector
print("Nilai Eigen Vector :", eigVec)
```

Gambar 11. Eigen, Eigen vec, Eigen Val

```
[-0.03081842+0.j,  0.02306467+0.j,  0.02126657+0.j, ...,
  0.00040227+0.j, -0.00019428+0.j,  0.00026816+0.j],
[-0.02994255+0.j,  0.02277164+0.j,  0.02258977+0.j, ...,
 -0.00039066+0.j, -0.00065492+0.j,  0.00050523+0.j],
...,
[-0.0280623 +0.j,  0.00344618+0.j, -0.03616237+0.j, ...,
 -0.00064923+0.j,  0.01979993+0.j, -0.01598867+0.j],
[-0.02902791+0.j,  0.00663781+0.j, -0.03286855+0.j, ...,
 -0.00839057+0.j, -0.03766694+0.j,  0.0175359 +0.j],
[-0.03096964+0.j,  0.00471598+0.j, -0.03064754+0.j, ...,
 -0.01060485+0.j,  0.01635514+0.j, -0.01158168+0.j]]))
```

Gambar 12. Hasil Perhitungan Eigen

```
Nilai eigen value : [ 5.29778787e+08+0.j  2.02771125e+08+0.j  1.48500080e+08+0.j ...
 -1.32916585e-10+0.j  8.02232337e-12+0.j  3.13550731e-10+0.j]
```

Gambar 13. Hasil nilai eigen val

```
Nilai Eigen Vector : [[-0.03192826+0.j  0.02272289+0.j  0.01895643+0.j ...  0.00140474+0.j
  -0.00247589+0.j  0.00342126+0.j]
 [-0.03081842+0.j  0.02306467+0.j  0.02126657+0.j ...  0.00040227+0.j
  -0.00019428+0.j  0.00026816+0.j]
 [-0.02994255+0.j  0.02277164+0.j  0.0225897 +0.j ... -0.00039066+0.j
  -0.00065492+0.j  0.00050523+0.j]
 ...
 [-0.0280623 +0.j  0.00344618+0.j -0.0361623 +0.j ... -0.00064923+0.j
   0.01979993+0.j -0.01598867+0.j]
 [-0.02902791+0.j  0.00663781+0.j -0.03286855+0.j ... -0.00839057+0.j
  -0.03766694+0.j  0.0175359 +0.j]
 [-0.03096964+0.j  0.00471598+0.j -0.03064754+0.j ... -0.01060485+0.j
   0.01635514+0.j -0.01158168+0.j]]
```

Gambar 14. Hasil nilai eigen vec

Langkah 7. Melakukan perhitungan untuk reduksi data. Dengan kode program :

```
Z = input_ori.dot(eigVec[:,0:dimensi])
# print("Data Z :", Z)
return Z # adalah data tereduksi menggunakan PCA
```

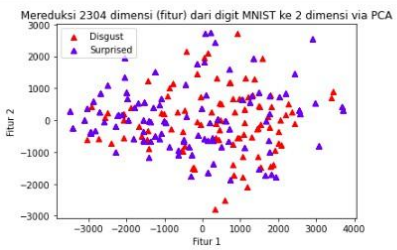
Gambar 15. Kode Reduksi data

```
Reduksi data : [[ -466.13632783  -798.77351757]
 [ 1326.57408597   799.69301474]
 [  491.67706831   501.98377128]
 [ -649.63747003   246.39343554]
 [ 1367.8864475   1294.82853132]
 [-1882.40158744   394.59398625]
 [ 1482.91291618   452.28584566]
 [ 1784.18735584   785.17625209]
 [ -468.06605672   308.71147685]
 [   38.39450833  1958.05383381]
 [ 1711.71601555   124.83702661]
 [ 1088.17610983  -447.07053317]
 [ -538.4644334   -945.42982955]
 [ 1903.93092151  -953.50382846]
 [-1678.02298832  -198.04579926]
 [  427.17035709  -602.32328596]
 [  336.84304391 -2788.56379807]
 [ 1194.51520831   734.40619712]
 [  367.3632251   1316.12924044]
```

Gambar 16. Hasil Reduksi Data

Langkah 8. Menampilkan plot dari kedua emosi.

```
plt.scatter(X[0:200,0], X[0:200,1], marker="^", c="red", label='Disgust')
plt.scatter(X[100:200,0], X[100:200,1], marker="^", c="blue", label='Surprised')
plt.title("Mereduksi 2304 dimensi (fitur) dari digit MNIST ke 2 dimensi via PCA")
plt.xlabel("Fitur 1")
plt.ylabel("Fitur 2")
plt.legend()
plt.show()
```



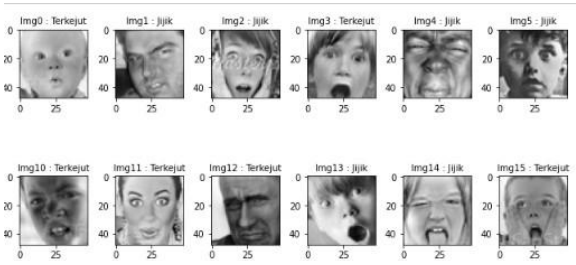
Gambar 17. Plot pada 2 emosi

Langkah 9. Klasifikasi dengan metode yang telah tersedia dalam library scikit-learn. Pada penelitian ini menggunakan klasifikasi SVM, berikut hasil akurasinya.

ISSN: 2503-071X
E-ISSN: 2621-511X

```
Akurasi SVM :  0.59375
```

Gambar 18. Nilai akurasi



Gambar 19. Prediksi PCA

Pada program CNN akan menggunakan library Tensorflow, keras dan beberapa data untuk mengilustrasikan penjelasan sederhana mengenai metode CNN.

Langkah 1 : Import library

```python
# import library
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
```

Gambar 20. library CNN

Seperti pada gambar .Import library dapat dijelaskan sebagai berikut :

1. Library matplotlib digunakan untuk menampilkan hasil analisis berupa grafik berwarna.
2. Library numpy digunakan untuk menangani permasalahan angka-angka. Berfokus pada scientific computing, NumPy memiliki kemampuan dalam membentuk objek N-dimensional array.
3. Library PIL digunakan untuk memanipulasi file gambar.
4. Library tensorflow digunakan untuk mengembangkan dan menerapkan Machine Learning dan algoritma lain yang memiliki banyak operasi matematika.

Langkah 2 : Menyiapkan dataset, pada metode *CNN* digunakan dataset FER(2013) sejumlah 5535 data citra terbagi dalam 6 kelas. Berikut adalah kode program untuk input data.

```python
import pathlib
data_dir = pathlib.Path('./ekspresi')

image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
5535
```

Gambar 21. Menyiapkan data

Langkah 3. Preprocessing data, buat parameter loader preprocessing, setelah itu buat pembagian antara data testing dan data traning dengan pembagian 80% training, 20% testing. Berikut adalah kode programnnya.

```python
#preprocessing data
batch_size = 32
img_height = 48
img_width = 48
```

Gambar 22. Preprocessing data

```python
#Training
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
data_dir,
validation_split = 0.2,
subset = "training",
seed = 123,
image_size = (img_height, img_width),
batch_size = batch_size)

#testing
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
data_dir,
validation_split = 0.2,
subset = "validation",
seed = 123,
image_size = (img_height, img_width),
batch_size = batch_size)
```

```
Found 5535 files belonging to 6 classes.
Using 4428 files for training.
Found 5535 files belonging to 6 classes.
Using 1107 files for validation.
```

Gambar 23. Pembagian train dan Test

Yang menjadi pembeda antara dua bagian training dan testing adalah variabel subset, ketika subset terisi training maka split yang digunakan adalah 1 – 0.2. namun jika subset yang digunakan validation maka akan menggunakan split 0.2.

```python
class_names = train_ds.class_names
print(class_names)
```

```
['Angry', 'Happy', 'Sad', 'Surpresed', 'disgust', 'fear']
```

Gambar 24. Nama Kelas Pada dataset

```python
#untuk mengetahui dataset sama dengan parameter atau tdk
for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
    print(labels_batch.shape)
    break
```

```
(32, 48, 48, 3)
(32,)
```

Gambar 25. Informasi pada dataset

Pada Gambar 25 menunjukkan bahwa informasi pada dataset sama dengan preprocessing data yang telah dibuat. Angka 3 pada keterangan diatas bernilai RGB dengan rentang nilai 0,255. Dimana nilai 0 untuk warna hitam dan 255 untuk warna putih. Sedangkan warna lain berada pada rentang nilai 0-255. Metode *CNN* harusnya memiliki nilai 0 dan 1. Jadi akan dilakukan normalisasi data dengan kode program.

Langkah 4. Normalisasi data

ISSN: 2503-071X
E-ISSN: 2621-511X

```
# Normalisasi data latih
AUTOTUNE = tf.data.experimental.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

normalization_layer = layers.experimental.preprocessing.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
```

Gambar 26. Normalisasi data

Langkah 5. Membangun model *CNN*

```
# Model CNN
num_classes = 6

model = Sequential([
    layers.experimental.preprocessing.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.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Gambar 27. Model CNN

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_1 (Rescaling)       (None, 48, 48, 3)         0
_________________________________________________________________
conv2d (Conv2D)              (None, 48, 48, 16)        448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 595,814
Trainable params: 595,814
Non-trainable params: 0
```

Gambar 28. Print Model CNN

Langkah 6. Menguji dataset dengan model yang telah dibuat.

```
#train dataset menggunakan model yang telah dibuat
epochs = 10
history = model.fit(
train_ds,
validation_data=val_ds,
epochs=epochs
)
```

Gambar 29. Pengujian data/klasifikasi

```
Epoch 1/10
139/139 [==============================] - 38s 223ms/step - loss: 1.6807 - accuracy: 0.2947 - val_loss: 1.5976 - val_accurac
y: 0.3695
Epoch 2/10
139/139 [==============================] - 11s 76ms/step - loss: 1.4789 - accuracy: 0.4151 - val_loss: 1.4772 - val_accurac
y: 0.4309
Epoch 3/10
139/139 [==============================] - 11s 78ms/step - loss: 1.3482 - accuracy: 0.4781 - val_loss: 1.4262 - val_accurac
y: 0.4345
Epoch 4/10
139/139 [==============================] - 11s 77ms/step - loss: 1.2281 - accuracy: 0.5287 - val_loss: 1.4148 - val_accurac
y: 0.4670
Epoch 5/10
139/139 [==============================] - 11s 81ms/step - loss: 1.1098 - accuracy: 0.5768 - val_loss: 1.4406 - val_accurac
y: 0.4661
Epoch 6/10
139/139 [==============================] - 11s 77ms/step - loss: 0.9963 - accuracy: 0.6292 - val_loss: 1.4265 - val_accurac
y: 0.4706
Epoch 7/10
139/139 [==============================] - 11s 76ms/step - loss: 0.8704 - accuracy: 0.6802 - val_loss: 1.4425 - val_accurac
y: 0.4878
Epoch 8/10
139/139 [==============================] - 11s 79ms/step - loss: 0.7556 - accuracy: 0.7337 - val_loss: 1.5441 - val_accurac
y: 0.4968
Epoch 9/10
139/139 [==============================] - 11s 78ms/step - loss: 0.6314 - accuracy: 0.7798 - val_loss: 1.6131 - val_accurac
y: 0.4724
Epoch 10/10
139/139 [==============================] - 10s 75ms/step - loss: 0.5191 - accuracy: 0.8223 - val_loss: 1.8178 - val_accurac
y: 0.4444
```

Gambar 30. Hasil Klasifikasi

Langkah 7. Prediksi dengan data baru

Gambar 31. Data uji baru

```
# coba data baru
test_path = pathlib.Path('./test_emosi/img_sd.jpg')

img = keras.preprocessing.image.load_img(
    test_path, target_size=(img_height, img_width)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions =  model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

Gambar 32. Kode Prediksi

```
This image most likely belongs to disgust with a 81.24 percent confidence.
```

Gambar 33. Hasil Prediksi

Langkah 8. Menampilkan visualisasi data accuracy dan loss.

Gambar 34. Visualisasi Data

3.3 Pengujian

Pada pengujiuan metode CNN pada saat menjalakan kode program prediksi, terdapat kesahalah pada gambar dan hasil, data yang digunakan untuk pengujian termasuk kedalam kelas “sad” tetapi hasil prediksi menunjukkan hasil “disgust” dengan presentase kemiripan 81,24% . Lalu peneliti mencoba mencoba mengatasi overfitting dengan menambahkan data augmentasi dan dropout, dengan kode program sebagai berikut.

Tabel 2. Perbandingan Reduksi data

	Uji 1 Emosi “Angry dan happy”	Uji 2 Emosi “Disguusted dan Surpresed”	Uji 3 Emosi “Happy, Angry dan Sad”
Data yang telah	[1941.17004669 -	[-466.13632783 -	[2.01550962e+03 -

ISSN: 2503-071X
E-ISSN: 2621-511X

direduksi	608.35670011] [152.30052138 1298.39632516] [-2127.96754938 -551.11520745] [1107.13207891 -160.25903563]	798.773517577] [1326.57408597 799.69301474] [491.677068361 501.983771278] [-649.637470003 246.393435554]	6.95591622e+02] [2.67187226e+02 1.19798863e+03] [-2.04512146e+03 -6.19234122e+02] [1.19271518e+03 -9.45307325e+01]

```
# Augmentasion DATA

data_augmentation = keras.Sequential(
[
    layers.experimental.preprocessing.RandomFlip("horizontal",
                                                 input_shape=(img_height,
                                                              img_width,
                                                              3)),
    layers.experimental.preprocessing.RandomRotation(0.1),
    layers.experimental.preprocessing.RandomZoom(0.1),
])
```

Gambar 35. Augmentasi data

```
# dropout

model = Sequential([
    data_augmentation,
    layers.experimental.preprocessing.Rescaling(1./255),
    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.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Gambar 36. DropOut

Setelah menambahkan Augmentation data dan Dropout, langkah berikutkan mengklasifikasi ulang data training.

```
Epoch 1/10
139/139 [==============================] - 15s 98ms/step - loss: 1.7483 - accuracy: 0.2319 - val_loss: 1.7016 - val_accurac
y: 0.2927
Epoch 2/10
139/139 [==============================] - 14s 98ms/step - loss: 1.6944 - accuracy: 0.2958 - val_loss: 1.6304 - val_accurac
y: 0.3487
Epoch 3/10
139/139 [==============================] - 14s 99ms/step - loss: 1.6458 - accuracy: 0.3245 - val_loss: 1.5951 - val_accurac
y: 0.3740
Epoch 4/10
139/139 [==============================] - 14s 100ms/step - loss: 1.6105 - accuracy: 0.3433 - val_loss: 1.5339 - val_accurac
y: 0.3902
Epoch 5/10
139/139 [==============================] - 14s 100ms/step - loss: 1.5849 - accuracy: 0.3616 - val_loss: 1.5774 - val_accurac
y: 0.3866
Epoch 6/10
139/139 [==============================] - 14s 100ms/step - loss: 1.5542 - accuracy: 0.3762 - val_loss: 1.5046 - val_accurac
y: 0.4002
Epoch 7/10
139/139 [==============================] - 14s 101ms/step - loss: 1.5307 - accuracy: 0.3853 - val_loss: 1.4931 - val_accurac
y: 0.4128
Epoch 8/10
139/139 [==============================] - 14s 102ms/step - loss: 1.4998 - accuracy: 0.4047 - val_loss: 1.4443 - val_accurac
y: 0.4381
Epoch 9/10
139/139 [==============================] - 14s 102ms/step - loss: 1.4680 - accuracy: 0.4160 - val_loss: 1.4161 - val_accurac
y: 0.4453
Epoch 10/10
139/139 [==============================] - 14s 102ms/step - loss: 1.4452 - accuracy: 0.4338 - val_loss: 1.4135 - val_accurac
y: 0.4544
```

Gambar 37. Hasil klasifikasi lagi

```
# coba data baru
test_path = pathlib.Path('./test_emosi/F03HA.jpg')

img = keras.preprocessing.image.load_img(
    test_path, target_size=(img_height, img_width)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions =  model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to Sad with a 79.79 percent confidence.
```

Gambar 38. Sebelum Dilakukan Augmentasi data

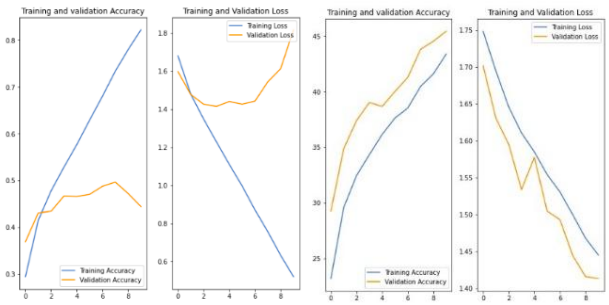
```
# coba data baru
test_path = pathlib.Path('./test_emosi/F03HA.jpg')

img = keras.preprocessing.image.load_img(
    test_path, target_size=(img_height, img_width)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)

predictions =  model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to Happy with a 55.26 percent confidence.
```

Gambar 39. Setelah Dilakukan Augmentasi data

Gambar 40. Gambar yang Diuji

Pada Gambar 38 adalah kode program prediksi sebelum dilakukannya proses Augmentation data dan dropout. Gambar uji termasuk dalam kelas “happy” sedangkan pada gambar 38 termasuk kedalam kelas “sad” dan setelah dilakukan augmentation data dan dropout dapat dilihat hasil pada gambar 39 hasil menunjukkan dia termasuk ke dalam kelas “happy” dengan presentase kemiripan sebesar 55,26%.



Gambar 41. Perbandingan Accuracy dan Loss

Pada Gambar 41 dapat dilihat perbandingan margin sebelah kiri saat sebelum adanya Data Augmentation dan Dropout dan margin sebelah kanan sesudah adanya Data Augmentation dan Dropout terlihat margin sudah tidak terlihat sangat besar pada validation accuracy, training accuracy, valodation loss dan training loss hal ini membuktikan bahwa model uji yang telah dibuat lebih baik dari sebelumnya.

4. KESIMPULAN

Pada penelitian ini berhasil menghitung nilai akurasi atau reduksi data dengan menggunakan metode *PCA* dan berhasil membuat model *CNN* yang baik sehingga tidak terjadi overfitting. Dengan hasil Nilai akurasi pada metode PCA sebesar 59,375% dan nilai akurasi pada metode CNN sebesar 59,386%.

Saran pada penelitian ini adalah agar menyempurnakan metode PCA dan CNN serta kedepannya diharap bisa digabung antara metode PCA dan CNN dengan catatan data training dan testing diolah pada metode PCA lalu diklasifikasi dengan metode CNN.

PUSTAKA

Wulanningrum Resty, Utami Ema "Penggunaan Principal Component Analysis dan Euclidean Distance untuk Identifikasi Citra Tanda Tangan", IPTEK-KOM, Vol. 16 No. 1, Juni 2014

Nugroho Adi Pulung,Fenriana Indah, Arijanto,M.Kom, Rudi "Implementasi Deep Learning Menggunakan Convolutional Neural Network(CNN) pada Ekspresi Manusia", April 2020. JURNAL ALGOR - VOL. 2 NO. 1

Yusuf Achmad, Wihandika Cahya Randy, Dewi Candra "Klasifikasi Emosi Berdasarkan Ciri Wajah Menggunakan Convolutional Neural Network, November 2019.

Hariri Rohman Fajar, Putra Pamungkas Danar, "Implementasi Metode PCA dan City Block Distance untuk Presensi Mahasiswa Berbasis Wajah", Seminar Nasional Teknologi Informasi, Komunikasi dan Aplikasinya Volume 04, Tahun 2017

Nasution Zulfahmi Muhammad, Nababan Addillah Adli, Syaliman Umam Khairul, Novelan Syahputra Muhammad, Jannah Miftahul "Penerapan Principal Component Analysis (PCA) Dalam Penentuan Faktor Dominan Yang Mempengaruhi Kanker Serviks (Studi kasus: Cervical Cancer dataset)" Jurnal Mantik Penusa Vol. 3, No. 1, Juni 2019

Johnson, W.A. & Wichern, D.W. 2007. Applied Multivariate Statistical Analysis. 6th Edition. Pearson Prentice Hall: New Jersey.

E. P Suartika I Wayan, Wijaya Yudhi Arya, Soelaiman Rully, "Klasifikasi Citra Menggunakan Convolutional Neural Network pada Caltech". Jurnal Teknik ITS Vol. 5, No. 1 Maret 2016

Alamsyah Derry, Pratama Dicky "Implementasi Convolutional Neural Networks (CNN) Untuk Klasifikasi Ekspresi Citra Wajah Pada FER-2013 Dataset" (Jurnal Teknologi Informasi) Vol.4, No.2, Desember 2020.

Shafira Tiara "Implementasi CNN untuk Klasifikasi Citra Tomat Menggunakan Keras". Maret 2018.