“LAPORAN *FINAL PROJECT*”

Pengenalan Emosi Pada Ekspresi Wajah Manusia Menggunakan Metode *Convolutional Neutral Network* (CNN)

Dosen pengampu : Dr. Basuki Rahmat, S.Si., M.T.

Disusun Oleh :

1. Aura Choirun Nisa 21081010173
2. Najwa Laila Anggraini 21081010191

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UPN “VETERAN” JAWA TIMUR

2023

1. PENDAHULUAN

Ekspresi wajah manusia adalah salah satu bentuk komunikasi non-verbal yang kaya dengan informasi emosional. Memahami dan mengenali emosi yang terkandung dalam ekspresi wajah sangat penting dalam berbagai aplikasi seperti pengenalan emosi otomatis, deteksi kebohongan, pengenalan emosi pada robot, dan interaksi manusia-komputer. Oleh karena itu, pengenalan emosi pada ekspresi wajah manusia telah menjadi topik penelitian yang menarik dan penting dalam bidang kecerdasan buatan dan pengolahan citra.

Seiring dengan kemajuan teknologi, metode *Convolutional Neural Network* (CNN) telah menunjukkan keunggulannya dalam pengenalan pola dan pengolahan citra. CNN adalah jenis arsitektur jaringan saraf yang terinspirasi oleh struktur visual korteks manusia, dengan kemampuan untuk mempelajari fitur-fitur yang relevan secara otomatis melalui lapisan konvolusi dan *pooling*.

Pada penelitian sebelumnya, metode CNN telah terbukti berhasil dalam pengenalan emosi pada ekspresi wajah manusia. Dengan menggunakan dataset berisi wajah-wajah manusia yang dilengkapi dengan label emosi yang sesuai, CNN dapat mempelajari pola-pola penting dalam wajah yang berkorelasi dengan ekspresi emosi tertentu.

Namun, masih ada beberapa tantangan yang perlu diatasi dalam pengenalan emosi pada ekspresi wajah manusia. Misalnya, variasi dalam ekspresi wajah, kondisi pencahayaan yang berbeda, perbedaan ras atau budaya, dan perbedaan individu dapat mempengaruhi akurasi pengenalan emosi. Oleh karena itu, penelitian lebih lanjut diperlukan untuk meningkatkan kinerja dan ketahanan sistem pengenalan emosi menggunakan metode CNN.

Dalam konteks ini, tujuan *project* ini adalah untuk menyelidiki dan mengembangkan metode pengenalan emosi pada ekspresi wajah manusia menggunakan metode *Convolutional Neural Network* (CNN). *Project* ini bertujuan untuk menganalisis efektivitas metode CNN dalam mengenali emosi pada ekspresi wajah manusia, serta mengatasi tantangan yang terkait dengan variasi dan keragaman wajah manusia. Hasil penelitian diharapkan dapat memberikan kontribusi dalam pengembangan aplikasi pengenalan emosi yang lebih akurat dan dapat diandalkan.

2. ISI

Dataset : https://www.kaggle.com/deadskull7/fer2013

Source Code :

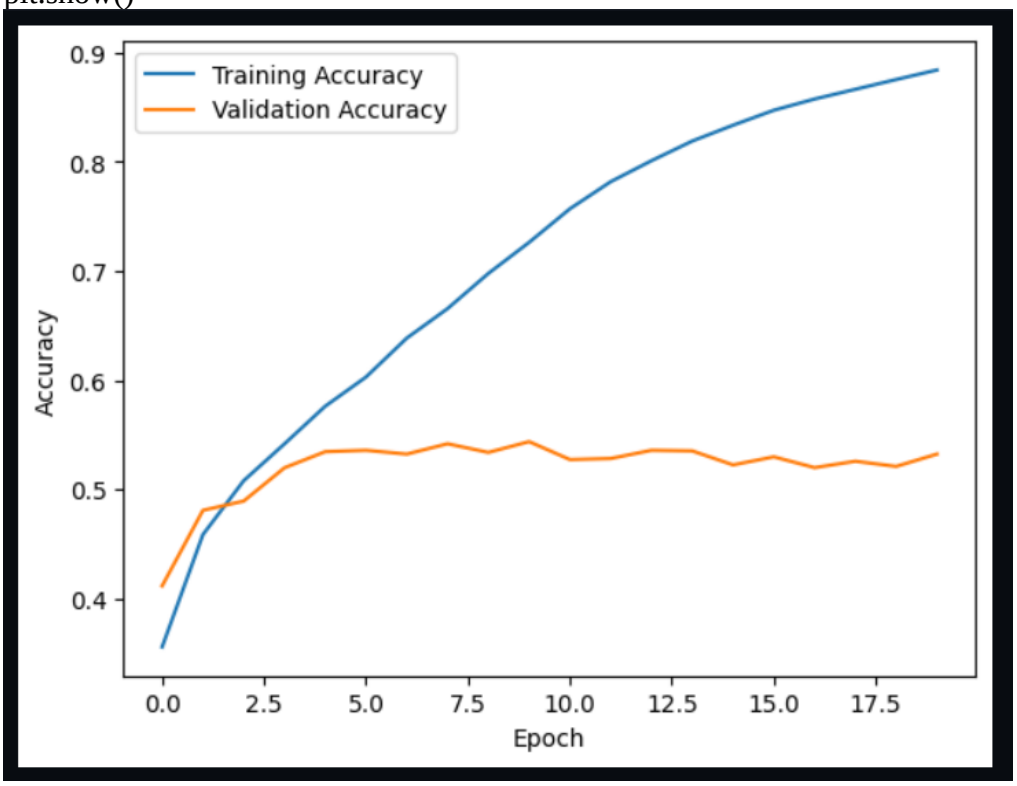
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
import cv2
import matplotlib.pyplot as plt

# input dataset
df = pd.read_csv('fer2013.csv')

# Preprocess the dataset
pixels = df['pixels'].tolist()
emotions = pd.get_dummies(df['emotion']).values
faces = []
for pixel_sequence in pixels:
    face = [int(pixel) for pixel in pixel_sequence.split(' ')]
    face = np.asarray(face).reshape(48, 48)
    faces.append(face.astype('float32'))
faces = np.asarray(faces)
faces = np.expand_dims(faces, -1)
faces_train, faces_test, emotions_train, emotions_test = train_test_split(faces, emotions,
test_size=0.2, random_state=42)

#model CNN
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(7, activation='softmax'))
model.add(layers.Flatten())
model.add(layers.Dense(200, activation='relu'))
model.add(layers.Dense(5))

# menampilkan arsitektur lengkap model CNN
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 46, 46, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 23, 23, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 128)               262272

 dense_3 (Dense)             (None, 7)                 903

...
Total params: 358,452
Trainable params: 358,452
Non-trainable params: 0
_________________________________________________________________
```

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

history = model.fit(faces_train, np.argmax(emotions_train, axis=1), epochs=20,
            validation_data=(faces_test, np.argmax(emotions_test, axis=1)))

test_loss, test_acc = model.evaluate(faces_test, np.argmax(emotions_test, axis=1),
verbose=2)
print('Test accuracy:', test_acc)
```

```
Epoch 1/20
898/898 [==============================] - 34s 37ms/step - loss: 1.7922 - accuracy: 0.3555 - val_loss: 1.5180 - val_accuracy: 0.4114
Epoch 2/20
898/898 [==============================] - 33s 36ms/step - loss: 1.4099 - accuracy: 0.4584 - val_loss: 1.3715 - val_accuracy: 0.4806
Epoch 3/20
898/898 [==============================] - 34s 37ms/step - loss: 1.2899 - accuracy: 0.5078 - val_loss: 1.3252 - val_accuracy: 0.4891
Epoch 4/20
898/898 [==============================] - 35s 39ms/step - loss: 1.2095 - accuracy: 0.5416 - val_loss: 1.2898 - val_accuracy: 0.5196
Epoch 5/20
898/898 [==============================] - 36s 40ms/step - loss: 1.1352 - accuracy: 0.5761 - val_loss: 1.2512 - val_accuracy: 0.5344
Epoch 6/20
898/898 [==============================] - 37s 41ms/step - loss: 1.0647 - accuracy: 0.6028 - val_loss: 1.2537 - val_accuracy: 0.5357
Epoch 7/20
898/898 [==============================] - 38s 42ms/step - loss: 0.9764 - accuracy: 0.6385 - val_loss: 1.3090 - val_accuracy: 0.5322
Epoch 8/20
898/898 [==============================] - 40s 44ms/step - loss: 0.8956 - accuracy: 0.6655 - val_loss: 1.3128 - val_accuracy: 0.5417
Epoch 9/20
898/898 [==============================] - 43s 47ms/step - loss: 0.8103 - accuracy: 0.6978 - val_loss: 1.4427 - val_accuracy: 0.5337
Epoch 10/20
898/898 [==============================] - 46s 52ms/step - loss: 0.7385 - accuracy: 0.7263 - val_loss: 1.5107 - val_accuracy: 0.5436
Epoch 11/20
898/898 [==============================] - 47s 52ms/step - loss: 0.6607 - accuracy: 0.7570 - val_loss: 1.6309 - val_accuracy: 0.5272
Epoch 12/20
898/898 [==============================] - 47s 52ms/step - loss: 0.5887 - accuracy: 0.7818 - val_loss: 1.7914 - val_accuracy: 0.5281
Epoch 13/20
...
Epoch 20/20
898/898 [==============================] - 48s 53ms/step - loss: 0.3266 - accuracy: 0.8842 - val_loss: 2.5823 - val_accuracy: 0.5322
225/225 - 3s - loss: 2.5823 - accuracy: 0.5322 - 3s/epoch - 14ms/step
Test accuracy: 0.5321816802024841
```

#accurary diagram
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()



#menampilkan hasil data latih
class_names = ['angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']

plt.figure(figsize=(10, 10))

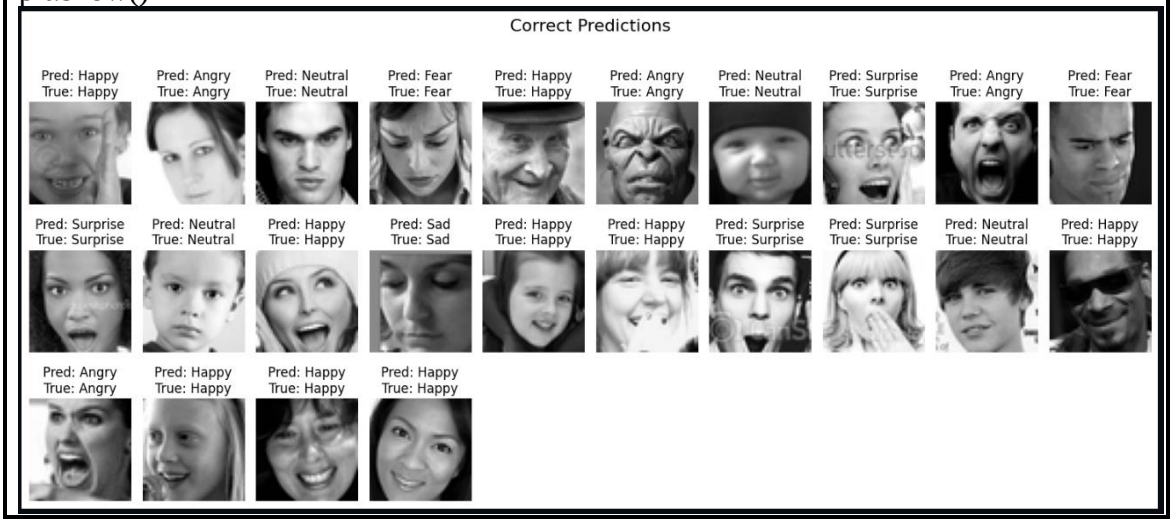
```
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_train[i], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_train[i])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_train[i + 4], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_train[i + 4])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_train[i + 8], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_train[i + 8])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_train[i + 12], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_train[i + 12])])
plt.show()
```

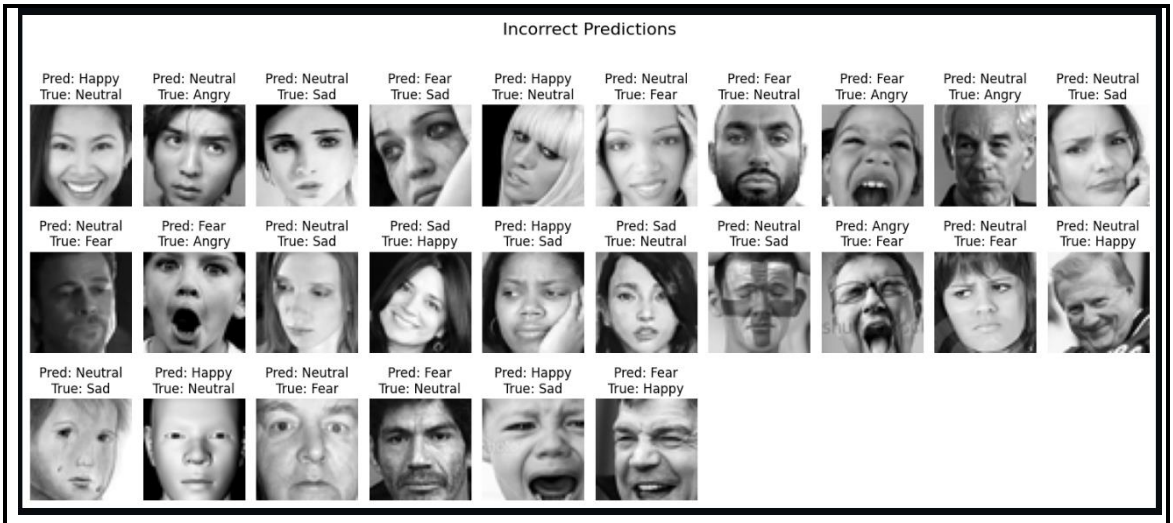

```
#menampilkan hasil data uji
class_names = ['angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_test[i], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_test[i])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
```

```
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_test[i + 4], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_test[i + 4])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_test[i + 8], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_test[i + 8])])
plt.show()

plt.figure(figsize=(10, 10))
for i in range(4):
    plt.subplot(4, 4, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(faces_test[i + 12], cmap='gray')
    plt.xlabel(class_names[np.argmax(emotions_test[i + 12])])
plt.show()
```


```
# Predict emotions for 50 images from the test dataset
n_images = 50
test_images = faces_test[:n_images]
test_labels = emotions_test[:n_images]

predictions = model.predict(test_images)
predicted_emotions = np.argmax(predictions, axis=1)
true_emotions = np.argmax(test_labels, axis=1)

# Define emotion labels
class_names = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

# Group predictions into correct and incorrect
correct_predictions = []
incorrect_predictions = []
for i in range(n_images):
```

```
    if predicted_emotions[i] == true_emotions[i]:
        correct_predictions.append(i)
    else:
        incorrect_predictions.append(i)

# Display the images with correct predictions
plt.figure(figsize=(15, 10))
for i, idx in enumerate(correct_predictions):
    plt.subplot(5, 10, i+1)
    plt.imshow(test_images[idx].reshape(48, 48), cmap='gray')
    plt.axis('off')
    plt.title("Pred: {}\nTrue: {}".format(class_names[predicted_emotions[idx]],
class_names[true_emotions[idx]]))
plt.tight_layout()
plt.suptitle('Correct Predictions', fontsize=16, y=1.05)
plt.show()

# Display the images with incorrect predictions
plt.figure(figsize=(15, 10))
for i, idx in enumerate(incorrect_predictions):
    plt.subplot(5, 10, i+1)
    plt.imshow(test_images[idx].reshape(48, 48), cmap='gray')
    plt.axis('off')
    plt.title("Pred: {}\nTrue: {}".format(class_names[predicted_emotions[idx]],
class_names[true_emotions[idx]]))
plt.tight_layout()
plt.suptitle('Incorrect Predictions', fontsize=16, y=1.05)
plt.show()
```



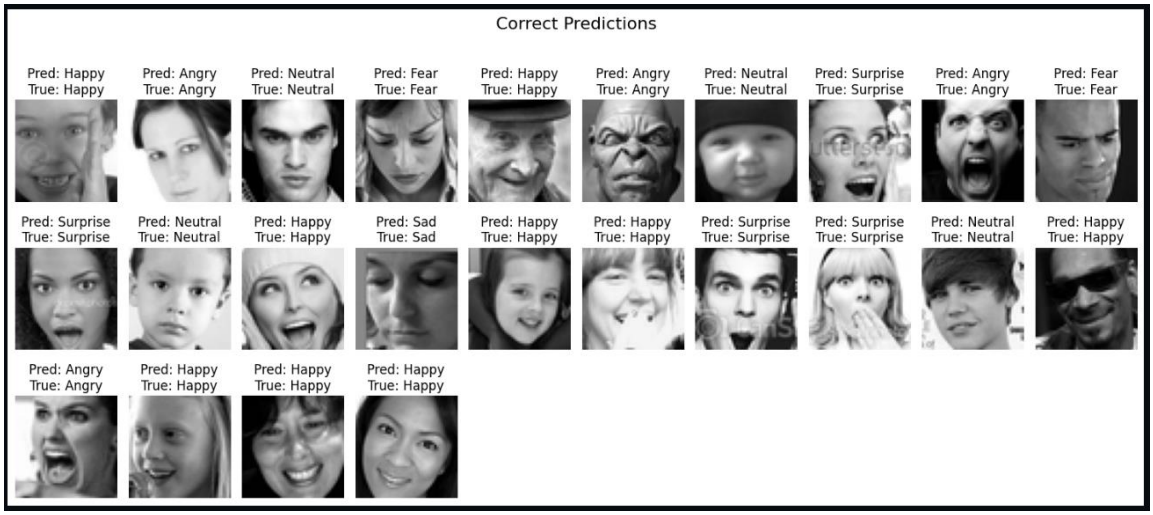



```
# Predict emotions for the entire test dataset
predictions = model.predict(faces_test)
predicted_emotions = np.argmax(predictions, axis=1)
true_emotions = np.argmax(emotions_test, axis=1)

# Calculate accuracy
accuracy = np.mean(predicted_emotions == true_emotions)
accuracy_percentage = accuracy * 100

print("Accuracy: {:.2f}%".format(accuracy_percentage))
```

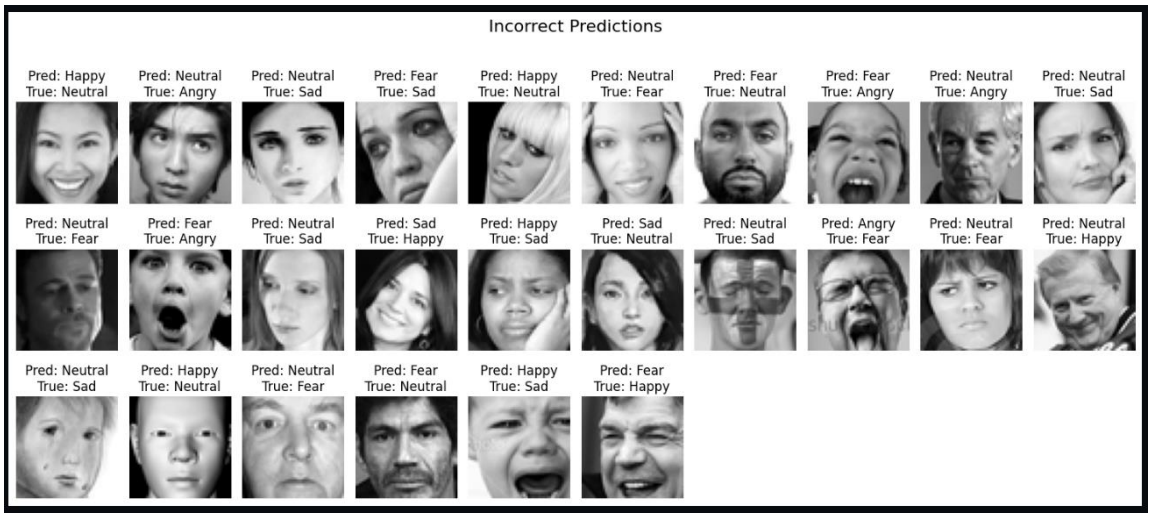
```
225/225 [==============================] - 2s 10ms/step
Accuracy: 53.22%
```

3. HASIL DAN PEMBAHASAN

Correct Predictions



Incorrect Predictions



Akurasi Presentase

```
225/225 [==============================] - 2s 10ms/step
Accuracy: 53.22%
```

Setelah dihitung menggunakan rumus, akurasi presentasi menunjukkan angka **53.22%.** ada beberapa faktor yang menyebabkan kecilnya akurasi tersebut, antara lain:

1. Keterbatasan dataset, hal ini dikarenakan pengenalan emosi sangat tergantung pada kualitas dan keberagaman dataset yang digunakan untuk melatih CNN;
2. Kondisi pencahayaan dan kualitas gambar juga dapat mempengaruhi akurasi pengenalan emosi, karena dapat mempengaruhi kejelasan gambar yang diambil;
3. Ekstraksi fitur yang tepat dari wajah manusia merupakan langkah penting dalam pengenalan emosi. Jika pengolahan fitur pada CNN tidak memadai, hal ini pasti akan mempengaruhi akurasi pengenalan emosi;
4. *Overfitting* dan *underfitting*, model CNN yang kompleks dapat mengalami *overfitting* jika terlalu diperkuat dengan data pelatihan. Hal ini dapat menyebabkan penurunan akurasi.

Peningkatan akurasi pengenalan emosi pada ekspresi wajah manusia menggunakan metode CNN dapat dilakukan dengan memperbaiki kualitas dataset pelatihan serta menambah kuantitasnya, memaksimalkan pengolahan fitur, serta melakukan tuning parameter model CNN.

DAFTAR PUSTAKA

Deadskull7. (2013). FER2013 [Dataset]. Kaggle. Diakses dari:

https://www.kaggle.com/deadskull7/fer2013