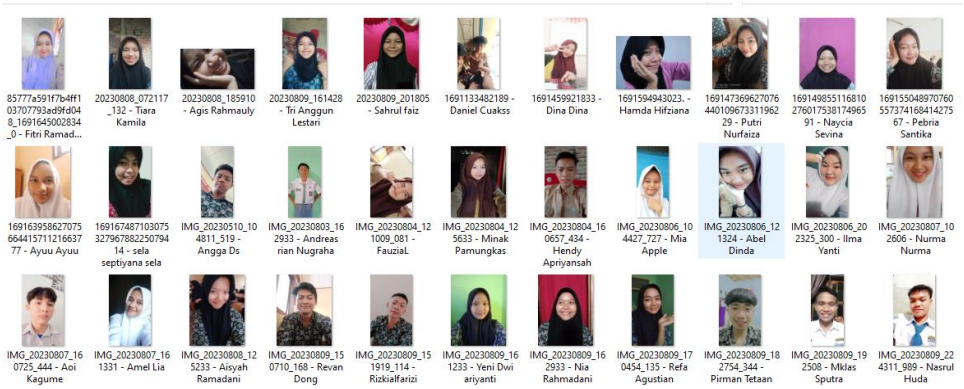
BAB IV
HASIL DAN PEMBAHASAN

4.1. *Preprocessing* Data

Setelah dilakukan perancangan maka dilakukan implementasi dari tiap Langkah-langkah tersebut yang dimulai dari *data collecting*, *data cleaning*, *data labelling* dan *data augmentation*, membagi data ke dalam data uji dan data latih.

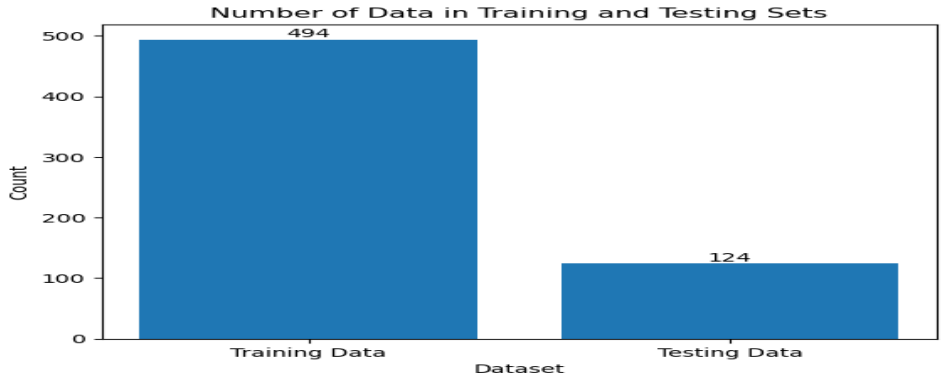
4.1.1. *Data Collecting*

Pengumpulan data dilakukan dengan mengambil citra gambar siswa dari SMAN 1 PENENGAHAN, Lampung Selatan. Dataset ini telah dianotasi dengan label yang mencakup informasi tentang ekspresi wajah, seperti marah, muak, takut, senang, netral, sedih, dan kaget. Jumlah total gambar ekspresi wajah yang berhasil dikumpulkan sebanyak 618 gambar, dengan masing-masing gambar memiliki ukuran piksel yang berbeda-beda. Seluruh gambar menampilkan variasi warna yang dapat dilihat pada contoh gambar 4.1. Berikut ini merupakan hasil dari proses pengumpulan data gambar ekspresi wajah tersebut.



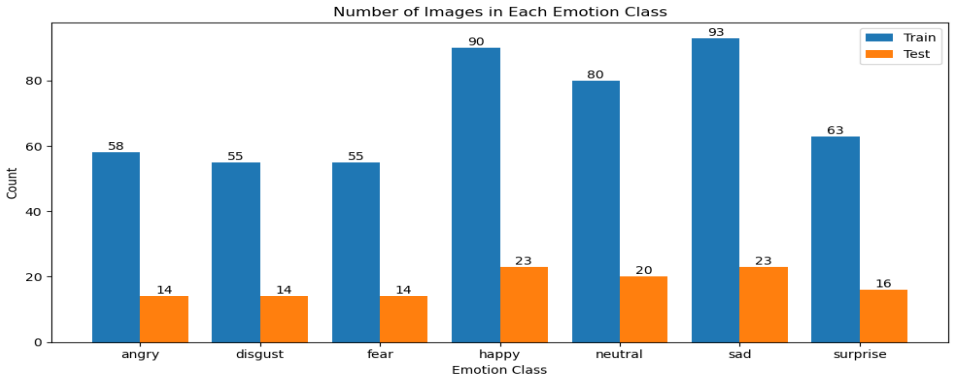
Gambar 4. 1 Hasil Pengumpulan Data ekspresi wajah

Dari hasil pengumpulan gambar data ekspresi wajah tersebut diperoleh hasil untuk data *training* dan data *testing* sebagai berikut :



Gambar 4. 2 Pembagian Data Training dan Testing

Berdasarkan gambar diatas dengan jumlah data sebanyak 618 gambar dengan gambar yang digunakan untuk data *training* sebanyak 494 gambar dan data testing sebanyak 124 gambar dimana dari jumlah gambar tersebut dibagi lagi menjadi tujuh bagian atau tujuh kelas ekspresi wajah sebagai berikut



Gambar 4. 3 Tampilan Splitting Data Berdasarkan Kelas

Berdasarkan gambar 4.3 di atas, dapat dijelaskan bahwa dataset untuk ekspresi wajah "marah/angry" terdiri dari total 60 gambar, dengan 46 gambar digunakan untuk pelatihan (train set) dan 14 gambar untuk pengujian (test set). Kelas "muak/disgust" memiliki total 58 gambar, di mana 44 gambar digunakan untuk pelatihan dan 14 gambar untuk pengujian. Kelas "takut/fear" memiliki total 58 gambar, dengan 44 gambar dalam set pelatihan dan 14 gambar dalam set pengujian. Kelas "senang/happy" memiliki total 95 gambar, terdiri dari 72 gambar dalam set pelatihan dan 23

gambar dalam set pengujian. Kelas "biasa/netral" mencakup total 84 gambar, dengan 64 gambar dalam set pelatihan dan 20 gambar dalam set pengujian. Kelas "sedih/sad" memiliki total 97 gambar, dengan 74 gambar dalam set pelatihan dan 23 gambar dalam set pengujian. Terakhir, kelas "kaget/surprise" memiliki total 66 gambar, dengan 50 gambar dalam set pelatihan dan 16 gambar dalam set pengujian.

4.1.2. *Data Cleaning*

Pembersihan data dilakukan agar didapatkan citra dengan kualitas baik, dan menyeragamkan ukuran dari citra. Berikut adalah kode program yang digunakan untuk melakukan pembersihan data.

```
# Fungsi untuk preprocessing gambar
def preprocess_images(folder_path, target_size=(48, 48)):
    class_folders = os.listdir(folder_path)

    for class_folder in class_folders:
        class_path = os.path.join(folder_path, class_folder)
        if os.path.isdir(class_path):
            images = os.listdir(class_path)

            for image_name in images:
                image_path = os.path.join(class_path, image_name)
                img = Image.open(image_path)

                # Resize gambar
                img = img.resize(target_size, Image.ANTIALIAS)

                # Konversi ke grayscale
                img = img.convert("L")

                # Simpan gambar yang telah diproses
                new_image_path = os.path.join(class_path, f"preprocessed_{image_name}")
                img.save(new_image_path)

                # Hapus gambar asli yang belum diproses
                os.remove(image_path)

# Preprocessing folder train
preprocess_images(train_folder)

# Preprocessing folder test
preprocess_images(test_folder)
```

Gambar 4. 4 Proses Pembersihan Data

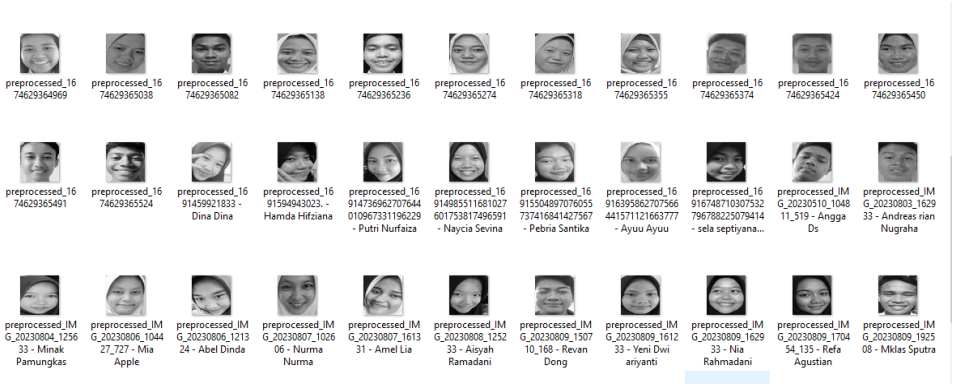
Pertama, kode mengimpor modul-modul yang diperlukan dan mendefinisikan fungsi preprocess_images untuk melakukan pra-pemrosesan pada gambar-gambar dalam folder yang diberikan.

Kemudian, dalam loop pertama, kode iterasi melalui setiap folder kelas dalam direktori yang ditentukan. Di dalam loop kelas folder, kode iterasi melalui gambar-gambar dalam setiap kelas.

Pada setiap iterasi gambar, gambar dibuka dan diubah ukurannya menjadi ukuran target yang diinginkan menggunakan metode .resize(). Selanjutnya, gambar dikonversi ke skala abu-abu menggunakan .convert("L") untuk menghapus komponen warna. Gambar yang telah diproses disimpan

kembali ke direktori dengan nama baru yang dimulai dengan "preprocessed_" menggunakan .save(). Gambar asli yang belum diproses dihapus dari direktori menggunakan os.remove().

Terakhir, kode menjalankan pra-pemrosesan pada folder train dan test secara terpisah dengan memanggil fungsi preprocess_images() untuk masing-masing folder. Sehingga hasil dari data *cleaning* yang telah dilakukan dapat dilihat pada gambar berikut:



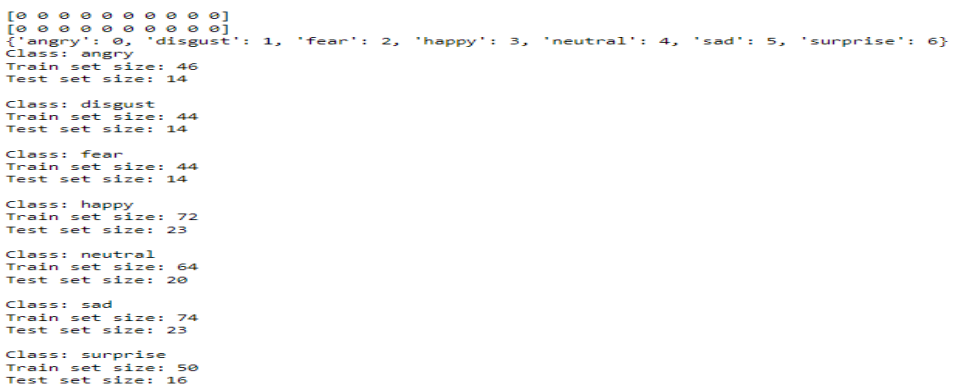
Gambar 4. 5 Proses Menyeragamkan Ukuran Gambar

Dalam proses penyeragaman gambar tersebut dilakukan pada semua gambar dataset yang dimiliki dimulai dari folder data train dan folder data test secara merata dari setiap kelas yang ada sehingga hasil akhir dari data cleaning ini adalah teknik pembersihan data yang dilakukan berhasil mengubah ukuran piksel semua gambar dalam dataset menjadi seragam, yaitu 48x48 piksel. Hal ini memiliki manfaat penting dalam pelatihan model, karena model mesin belajar cenderung lebih baik dalam memahami data yang memiliki ukuran seragam. Dengan ukuran piksel yang konsisten, dapat menghindari masalah ketidakcocokan ukuran saat proses pelatihan dan pengujian model dan konversi semua gambar ke skala abu-abu menghilangkan komponen warna dan mengubahnya menjadi gambar dengan skala keabuan. Ini berguna karena model pembelajaran mesin sering lebih baik dalam mengenali pola dan fitur dalam gambar monokromatis seperti gambar abu-abu. Selain itu, perubahan ini juga membantu

mengurangi kompleksitas data, sehingga dapat mempercepat proses pelatihan model dan mengurangi konsumsi sumber daya.

4.1.3. *Data Labeling*

Pelabelan data digunakan untuk memberikan label pada setiap kelas ekspresi wajah (marah, muak, takut, senang, netral, sedih dan kaget), dalam proses melakukan *labeling* data dikarenakan data yang digunakan merupakan data yang bersifat *open-source* maka, dataset gambar yang digunakan telah diberikan label oleh pemilik dataset tersebut sehingga peneliti tidak perlu melakukan pelabelan data kembali, oleh karena itu prose pelabelan data ini dilakukan dengan mengganti atau mengkonversi label kelas yang sebelumnya berupa teks atau tipe data string menjadi label berupa angka atau menjadi label kelas dengan tipe data array (angry': 0, 'disgust': 1, 'fear': 2, 'happy': 3, 'neutral': 4, 'sad': 5, 'surprise': 6). Hasil dari data labeling ini dapat dilihat pada 4.6 gambar berikut:



Gambar 4. 6 Proses Pelabelan Data

Dapat dilihat pada gambar diatas bahwa terdapat tujuh label kelas yang telah diubah menjadi label dalam bentuk array dengan masing-masing kelas memiliki jumlah data untuk setiap set yang dapat dilihat pada tabel berikut

Tabel 4. 1 Hasil Pelabelan Data

Class	Lables	Train set size	Test set size
Angry	**0**	**46**	**14**
Disgust	**1**	**44**	**14**
Fear	**2**	**44**	**14**
Happy	**3**	**72**	**23**
Neutral	**4**	**64**	**20**
Sad	**5**	**74**	**23**
Surprise	**6**	**50**	**16**

Dari tabel diatas dapat dijelaskan bahwa sebelum dilakukan labeling data diketahui kelas masih dalam bentuk string yaitu 'angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral' yang kemudian setelah dilakukan labeling data maka maka label kelas menjadi bentuk array yaitu '0', '1', '2', '3', '4', '5', '6' yang kemudian diterapkan untuk setiap gambar yang ada dalam dataset.

4.1.4. *Data Augmentation*

Data *augmentation* dilakukan bertujuan untuk mengurangi *overfitting* pada *dataset* yang dimiliki. Selain itu fungsi *augmentasi* gambar ini juga dapat meningkatkan jumlah data pada gambar. Namun, data tersebut hanya bisa digunakan pada model pelatihan. Berikut adalah kode yang digunakan untuk melakukan *augmentasi* gambar.

```
# Preprocess the image data using ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_folder,
    target_size=(48, 48),
    color_mode='grayscale',
    batch_size=64,
    class_mode='categorical',
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    test_folder,
    target_size=(48, 48),
    color_mode='grayscale',
    batch_size=64,
    class_mode='categorical',
    shuffle=False
)
```

Gambar 4. 7 Proses Augmentasi

Pada program di atas memiliki fungsi *ImageDataGenerator* berfungsi untuk melakukan *augmentasi* gambar yang kemudian akan disimpan pada variabel

datagen berikut adalah parameter yang akan digunakan untuk membuat *augmentasi* gambar :

- Objek *ImageDataGenerator* yang akan digunakan untuk pra-pemrosesan data gambar, yaitu *train_datagen* dan *test_datagen.*
- Objek train_datagen akan digunakan untuk melatih model, sedangkan test_datagen akan digunakan untuk menguji model.
- Pada objek *ImageDataGenerator,* peneliti menggunakan argumen rescale=1./255. Argumen ini akan mengubah rentang nilai piksel gambar menjadi 0-1, dengan tujuan untuk normalisasi data. Dalam hal ini, setiap nilai piksel gambar akan dibagi oleh 255.
- Selanjutnya, membuat generator untuk data latih menggunakan train_datagen.flow_from_directory(). Generator ini akan mengambil data gambar dari direktori train_folder yang sudah dijelaskan sebelumnya dalam script yang tidak tercantum di atas.
- Di dalam train_datagen.flow_from_directory(), kita memberikan beberapa argumen sebagai berikut:
    - → train_folder: Direktori tempat data latih gambar berada.
    - → target_size=(48, 48): Ukuran yang diharapkan untuk gambar setelah diresize.
    - → color_mode='grayscale': Mode warna gambar yang diharapkan. Dalam hal ini, gambar akan diubah menjadi skala abu-abu.
    - → batch_size=64: Jumlah sampel gambar yang akan digunakan pada setiap iterasi dalam proses pelatihan.
    - → class_mode='categorical': Mode kelas yang digunakan. Dalam hal ini, peneliti menggunakan kelas-kelas kategori.
    - → shuffle=True: Mengacak urutan sampel gambar setiap kali epoch baru dimulai.
- Fungsi train_datagen.flow_from_directory() akan menghasilkan sebuah generator yang akan menghasilkan batch-batch gambar dan labelnya pada setiap iterasi selama proses pelatihan.

- Selanjutnya, membuat generator untuk data uji menggunakan test_datagen.flow_from_directory(). Generator ini akan mengambil data gambar dari direktori test_folder yang juga sudah dijelaskan sebelumnya.
- Argumen dalam test_datagen.flow_from_directory() mirip dengan argumen dalam train_datagen.flow_from_directory(). Namun, terdapat perbedaan pada argumen shuffle yang diatur menjadi False. Hal ini dilakukan agar urutan sampel gambar pada data uji tetap sesuai dengan urutan aslinya, sehingga dapat digunakan untuk evaluasi model dengan benar.
- Fungsi test_datagen.flow_from_directory() akan menghasilkan sebuah generator yang akan menghasilkan batch-batch gambar dan labelnya pada setiap iterasi selama proses pengujian model.

4.2. Pembuatan Model CNN

Pada pemodelan arsitektur CNN peneliti menggunakan Convolutional Layer (Conv2D), MaxPooling Layer (MaxPooling2D), Convolutional Layer (Conv2D), Flatten Layer, Dense Layer (Fully Connected), Dropout Layer dan Dense Layer (Output Layer). Pemodelan CNN menggunakan *tensorflow* dapat di lihat di bawah ini

```
# Create the CNN model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Gambar 4. 8 Proses Pembuatan Model

Dapat dilihat bahwa dalam model CNN yang diberikan, terdapat beberapa layer konvolusi dengan ukuran filter yang sama (3x3) yang dapat dijelaskan sebagai berikut:

- Convolutional Layer (Conv2D) adalah layer pertama dalam model. Pada penelitian ini, 32 filter dengan ukuran (3, 3) diterapkan pada gambar input dengan fungsi aktivasi ReLU (Rectified Linear Activation). Ini bertujuan untuk mengekstrak fitur-fitur penting dari gambar, seperti tepi, sudut, atau tekstur.
- MaxPooling Layer (MaxPooling2D) setelah setiap layer Convolutional, model memiliki layer MaxPooling. Layer ini berfungsi untuk mereduksi dimensi spasial gambar dengan mengambil nilai maksimum di dalam jendela (misalnya, 2x2) dari gambar. Ini membantu mengurangi jumlah parameter yang perlu diproses oleh model, serta membantu mengidentifikasi fitur-fitur yang paling signifikan.
- Convolutional Layer (Conv2D) model memiliki dua layer konvolusi tambahan dengan 64 dan 128 filter, masing-masing juga menggunakan fungsi aktivasi ReLU. Ini bertujuan untuk melakukan lebih banyak ekstraksi fitur berdasarkan hasil dari langkah sebelumnya, semakin kompleks dan abstrak.
- Flatten Layer setelah ekstraksi fitur melalui layer konvolusi, gambar di-flatten menjadi vektor 1D. Ini mempersiapkan data untuk masukan ke lapisan Dense (fully connected) selanjutnya.
- Dense Layer (Fully Connected) merupakan layer yang memiliki 512 neuron dan menggunakan fungsi aktivasi ReLU. Ini adalah bagian dari jaringan saraf yang sepenuhnya terhubung, yang menggabungkan fitur-fitur yang dihasilkan oleh layer konvolusi menjadi representasi yang lebih abstrak.
- Dropout Layer adalah layer Dropout yang membantu mencegah overfitting. Dropout menghilangkan acak sebagian dari unit (neuron)

dalam layer sebelumnya selama pelatihan. Ini membantu model menjadi lebih umum dan lebih robust.

- Dense Layer (Output Layer) layer ini memiliki neuron sejumlah 'num_classes', yang dalam penelitian ini mewakili kelas-kelas ekspresi wajah. Pada laye ini menggunakan fungsi aktivasi softmax, yang menghasilkan probabilitas untuk setiap kelas, memberikan hasil yang mengindikasikan kemungkinan kelas ekspresi yang paling sesuai dengan gambar input.

Berdasarkan penjelasan diatas maka, hasil dari pembuatan model CNN dapat dilihat seperti pada gambar 4.9 di bawah ini :

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

 max_pooling2d_3 (MaxPoolin  (None, 23, 23, 32)        0
 g2D)

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

 max_pooling2d_4 (MaxPoolin  (None, 10, 10, 64)        0
 g2D)

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

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 512)               1049088

 dropout_1 (Dropout)         (None, 512)               0

 dense_3 (Dense)             (None, 7)                 3591

=================================================================
Total params: 1145351 (4.37 MB)
Trainable params: 1145351 (4.37 MB)
Non-trainable params: 0 (0.00 Byte)
```

Gambar 4. 9 Hasil Pembuatan Model

Dari hasil pembuatan model CNN di atas memiliki beberapa bagian yaitu

- Layer = deskripsi setiap layer yang sudah di rancang
- Output_shape = mendeskripsikan ukuran lebar, tinggi dari gambar dan juga jumlah dari filter.
- Untuk mencari *output shape* atau *output size* pada layer konvolusi dapat menggunakan persamaan 2.1.

  Contoh untuk mencari *output shape* pada layer konvolusi pertama yang memiliki *input* sebagai berikut

  - $n = (48)$

- f = (3)
- $p = 0\ (tidak\ ada\ padding)$
- s = 1

sehingga dapat dilakukan dengan cara seperti di bawah ini

$$\frac{48 + 2 * 0 - 3}{1} + 1 = 46$$

Jadi *output_shape* yang dihasilkan adalah $(none, 46, 46, 32)$

- Sedangkan untuk mencari *output shape* dari layer *MaxPooling* pertama dapat menggunakan persamaan 2.3.
  contoh untuk mendapatkan *output shape* dari layer *max pooling* yang memiliki *input*

- n = (46)
- p = (0)
- s = 2

sehingga dapat dilakukan dengan cara seperti di bawah ini

$$\frac{46 - 0}{2} + 1 = 23$$

Jadi *output_shape* yang dihasilkan adalah $(none, 23, 23, 32)$

4.3. Proses Pelatihan Model

Sebelum melakukan pelatihan peneliti mengcompile model yang bertujuan untuk mengkonfigurasi model sebelum dilakukan proses pelatihan. Pengaturan ini dilakukan agar model yang dihasilkan memiliki performa yang baik. Dapat dilihat pada *source code* berikut ini.

```
# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

Gambar 4. 10 Proses Pelatihan Model dengan optimizer Adam

Pada *source code* di atas, model di-compile menggunakan optimizer Adam, loss function 'categorical_crossentropy', dan metrics 'accuracy'. dimana fungsinya adalah sebagai berikut :

- Optimizer = Optimizer Adam digunakan untuk mengoptimalkan model selama proses pelatihan dengan memperbarui bobot (weights) berdasarkan gradien loss function.
- Loss = Loss function (fungsi kerugian) digunakan untuk mengukur seberapa baik model memprediksi target yang benar. Pemilihan loss function pada tugas klasifikasi multikelas yang umum digunakan adalah, 'categorical_crossentropy'
- Metrik = Metrik evaluasi digunakan untuk mengukur kinerja model selama pelatihan dan evaluasi dimana dalam penelitian ini peneliti menggunakan Metrics 'accuracy' untuk mengukur akurasi model selama pelatihan.

Setelah model selesai di compile maka selanjutnya adalah melakukan proses latih berikut adalah *source code* untuk melatih model CNN.

```
# Train the model
history = model.fit(train_generator, epochs=20, validation_data=test_generator)
```

Gambar 4. 11 Proses Pelatihan Model dengan Metode Fit()

Pada *source code* di atas dilakukan proses pelatihan model menggunakan metode fit() pada objek model. Proses pelatihan dilakukan dengan menggunakan data pelatihan (train_generator) dan dilakukan dalam beberapa epoch (dalam contoh ini, 20 epoch).

Selama pelatihan, model akan diperbarui secara iteratif dengan memperhitungkan loss function dan menggunakan optimizer yang telah ditentukan sebelumnya. Setiap epoch, model akan melalui semua batch data pelatihan dan mengoptimalkan bobot (weights) berdasarkan gradien loss function.

Selain itu, untuk memonitor kinerja model selama pelatihan, dilakukan validasi menggunakan data validasi (test_generator). Hal ini memungkinkan untuk memantau kinerja model pada data yang tidak digunakan selama pelatihan.

Setelah proses pelatihan selesai, hasil pelatihan akan disimpan dalam objek history, yang berisi riwayat loss dan metrik evaluasi pada setiap epoch. Riwayat ini dapat digunakan untuk menganalisis kinerja model selama pelatihan dan melakukan visualisasi grafik loss dan akurasi.

Hasil yang diperoleh dari proses *training* model CNN yang telah dibuat dapat dilihat pada gambar di bawah ini:

```
_________________________________________________________________
Epoch 1/20
8/8 [==============================] - 5s 478ms/step - loss: 1.9455 - accuracy: 0.1606 - val_loss: 1.9245 - val_accuracy: 0.185
5
Epoch 2/20
8/8 [==============================] - 3s 354ms/step - loss: 1.9404 - accuracy: 0.1707 - val_loss: 1.9218 - val_accuracy: 0.161
3
Epoch 3/20
8/8 [==============================] - 3s 362ms/step - loss: 1.9333 - accuracy: 0.1870 - val_loss: 1.9319 - val_accuracy: 0.185
5
Epoch 4/20
8/8 [==============================] - 3s 327ms/step - loss: 1.9319 - accuracy: 0.1809 - val_loss: 1.9231 - val_accuracy: 0.185
5
Epoch 5/20
8/8 [==============================] - 3s 337ms/step - loss: 1.9242 - accuracy: 0.1890 - val_loss: 1.9187 - val_accuracy: 0.185
5
Epoch 6/20
8/8 [==============================] - 3s 331ms/step - loss: 1.9275 - accuracy: 0.2053 - val_loss: 1.9193 - val_accuracy: 0.185
5
Epoch 7/20
8/8 [==============================] - 3s 346ms/step - loss: 1.9266 - accuracy: 0.1707 - val_loss: 1.9200 - val_accuracy: 0.185
5
Epoch 8/20
8/8 [==============================] - 3s 336ms/step - loss: 1.9165 - accuracy: 0.1870 - val_loss: 1.9163 - val_accuracy: 0.209
7
Epoch 9/20
8/8 [==============================] - 3s 353ms/step - loss: 1.9110 - accuracy: 0.2093 - val_loss: 1.9097 - val_accuracy: 0.217
7
Epoch 10/20
8/8 [==============================] - 3s 349ms/step - loss: 1.8911 - accuracy: 0.2175 - val_loss: 1.8897 - val_accuracy: 0.201
6
Epoch 11/20
8/8 [==============================] - 3s 332ms/step - loss: 1.8482 - accuracy: 0.2439 - val_loss: 1.8628 - val_accuracy: 0.290
3
Epoch 12/20
8/8 [==============================] - 3s 333ms/step - loss: 1.8126 - accuracy: 0.2805 - val_loss: 1.8348 - val_accuracy: 0.250
0
Epoch 13/20
8/8 [==============================] - 3s 330ms/step - loss: 1.7730 - accuracy: 0.3008 - val_loss: 1.7999 - val_accuracy: 0.282
3
Epoch 14/20
8/8 [==============================] - 3s 333ms/step - loss: 1.7260 - accuracy: 0.2846 - val_loss: 1.7835 - val_accuracy: 0.298
4
Epoch 15/20
8/8 [==============================] - 3s 335ms/step - loss: 1.6579 - accuracy: 0.3801 - val_loss: 1.7530 - val_accuracy: 0.282
3
Epoch 16/20
8/8 [==============================] - 3s 332ms/step - loss: 1.6231 - accuracy: 0.3699 - val_loss: 1.7573 - val_accuracy: 0.314
5
Epoch 17/20
8/8 [==============================] - 3s 341ms/step - loss: 1.5735 - accuracy: 0.4167 - val_loss: 1.7407 - val_accuracy: 0.282
3
Epoch 18/20
8/8 [==============================] - 3s 380ms/step - loss: 1.5364 - accuracy: 0.4248 - val_loss: 1.7070 - val_accuracy: 0.314
5
Epoch 19/20
8/8 [==============================] - 5s 618ms/step - loss: 1.4387 - accuracy: 0.4797 - val_loss: 1.6754 - val_accuracy: 0.330
6
Epoch 20/20
8/8 [==============================] - 3s 383ms/step - loss: 1.4021 - accuracy: 0.4593 - val_loss: 1.6790 - val_accuracy: 0.330
```

Gambar 4. 12 Proses Pelatihan Setiap Epoch

Hasil pelatihan model pada gambar diatas dapat dilihat dari output yang diberikan pada setiap epoch:

- Epoch 1

  Pada awal pelatihan, model mulai dengan bobot acak. Selama epoch pertama, model menjalani proses pelatihan dengan menghitung fungsi loss berdasarkan perbandingan prediksi model dengan nilai sebenarnya pada data pelatihan. Dalam kasus ini, fungsi loss

mencapai 1.9455, menunjukkan bahwa model belum mampu mengenali pola dengan baik. Akurasi yang rendah, yaitu 0.1606, mengindikasikan bahwa model masih belum efektif dalam mengklasifikasikan ekspresi wajah pada dataset pelatihan. Hasil validasi juga menggambarkan performa serupa, dengan validasi loss sebesar 1.9245 dan validasi akurasi sebesar 0.1855.

- Epoch 2

  Pada epoch kedua, model dilatih kembali dengan bobot yang telah diperbarui dari epoch sebelumnya. Meskipun terdapat sedikit peningkatan dalam akurasi menjadi 0.1707, hasil yang diperoleh masih tetap rendah. Fungsi loss yang mendekati 1.9404 menunjukkan bahwa model masih kesulitan dalam memahami pola pada data pelatihan. Performa pada validasi juga belum menunjukkan perbaikan signifikan, dengan validasi loss sebesar 1.9218 dan validasi akurasi sebesar 0.1613.

- Epoch 3

  Pada epoch ketiga, terjadi peningkatan dalam akurasi menjadi 0.1870, menunjukkan bahwa model secara perlahan mulai menangkap pola pada data pelatihan. Meskipun demikian, hasil validasi tetap belum menunjukkan kemajuan yang signifikan, dengan validasi loss sebesar 1.9319 dan validasi akurasi sebesar 0.1855.

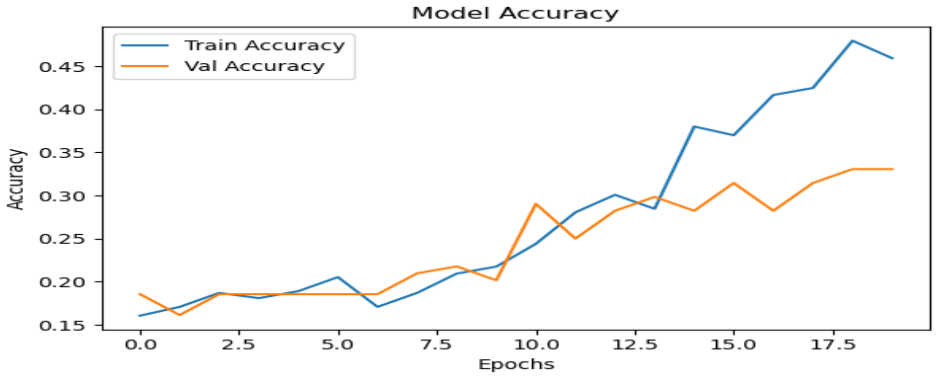
- Epoch 4-20

  Proses pelatihan berlanjut pada epoch selanjutnya. Model terus diperbarui dengan mempertimbangkan hasil pelatihan sebelumnya. Meskipun terdapat variasi dalam performa pada data pelatihan, hasil validasi masih cenderung fluktuatif. Terlihat bahwa akurasi pada data pelatihan naik menjadi sekitar 0.4593 pada epoch ke-20, namun, hasil validasi masih belum stabil dengan validasi akurasi sebesar 0.3306.

Proses pelatihan model Convolutional Neural Network (CNN) dengan arsitektur yang disediakan berlangsung selama 20 epoch. Pada awalnya, model mengalami kinerja rendah dengan akurasi yang masih terbatas, terutama pada dataset validasi. Selama iterasi berikutnya, terlihat sedikit peningkatan dalam akurasi pada dataset pelatihan, tetapi hasil validasi masih fluktuatif. Meskipun terdapat upaya dalam penyesuaian bobot model, performa masih belum stabil dan belum mencapai tingkat yang memadai pada tugas klasifikasi ekspresi wajah ini. Diperlukan langkah-langkah tambahan, seperti penyesuaian arsitektur, hyperparameter, atau pendekatan pelatihan, untuk mengoptimalkan model dan mencapai hasil yang lebih baik dalam mengenali ekspresi wajah.

4.4. Hasil Akurasi dan *Loss*

Berikut adalah hasil akurasi dan *loss*. Hasil akurasi dan *loss* digunakan sebagai perbandingan apakah proses pelatihan sudah berjalan dengan baik atau tidak.



Gambar 4. 13 Hasil akurasi Training dan Validasi

Dari hasil pelatihan yang sudah dilakukan dapat dilihat pada gambar 4.13. bahwa hasil akurasi dari proses *training* dimana grafik menunjukan tren perubahan akurasi pelatihan dan validasi selama 20 epoch. Pada awal pelatihan, terdapat fluktuasi pada akurasi pelatihan dan validasi, menunjukkan adaptasi awal model terhadap data. Selanjutnya, terlihat

adanya peningkatan yang lebih stabil pada akurasi pelatihan seiring berjalannya epoch. Namun, akurasi validasi cenderung memiliki variasi dan tidak selalu mengikuti tren peningkatan yang sama dengan akurasi pelatihan. Meskipun akurasi pelatihan terus meningkat, akurasi validasi tidak selalu mengikuti tren tersebut dan mungkin mengalami fluktuasi atau bahkan stagnasi pada beberapa epoch.

Pada tabel 4.2. dapat dilihat perbandingan akurasi antara data uji dan data validasi

Tabel 4. 2 Hasil Akurasi Training dan Validasi

Epoch	***Training ACC***	**Validation ACC**
1	0.1606	0.1855
2	0.1707	0.1613
3	0.1870	0.1855
4	0.1809	0.1855
5	0.1890	0.1855
6	0.2053	0.1855
7	0.1707	0.1855
8	0.1870	0.2097
9	0.2093	0.2177
10	0.2175	0.2016
11	0.2439	0.2903
12	0.2805	0.2500
13	0.3008	0.2823
14	0.2846	0.2984
15	0.3801	0.2823
16	0.3699	0.3145
17	0.4167	0.2823
18	0.4248	0.3145
19	0.4797	0.3306

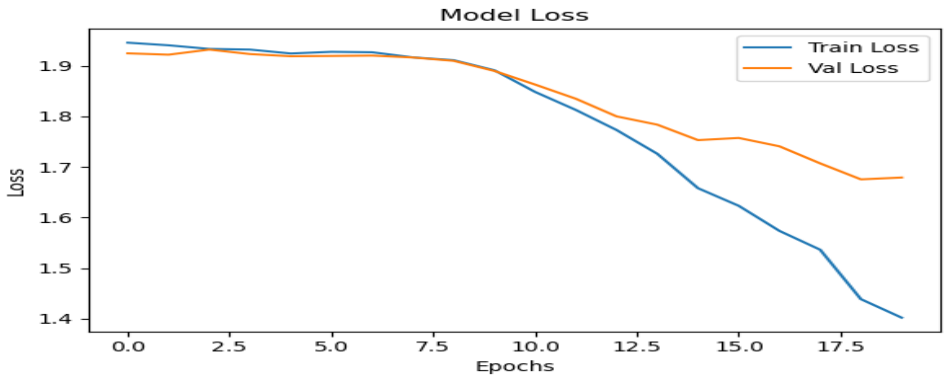
Epoch	***Training ACC***	**Validation ACC**
20	0.4593	0.3306

Tabel 4.2 menggambarkan hasil evaluasi akurasi pada tahap pelatihan dan validasi selama 20 epoch dalam konteks analisis klasifikasi ekspresi wajah. Selama lima epoch pertama, terdapat pola naik-turun yang signifikan dalam akurasi pelatihan, mengindikasikan adaptasi awal model terhadap data pelatihan. Namun, hasil validasi pada periode tersebut belum menunjukkan tren konsisten dan berfluktuasi di sekitar titik awal.

Pada epoch ke-6 hingga ke-15, terlihat tren perbaikan yang lebih konsisten pada akurasi pelatihan, mencerminkan kemampuan model untuk mengekstrak fitur-fitur relevan dari data pelatihan. Meskipun akurasi pelatihan cenderung meningkat, hasil validasi tetap bervariasi dan belum menunjukkan kemajuan yang sejajar. Namun, pada akhir periode ini, terdapat peningkatan signifikan dalam akurasi validasi pada beberapa epoch, menunjukkan adanya perkembangan dalam generalisasi model.

Pada tahap akhir pelatihan (epoch ke-16 hingga ke-20), akurasi pelatihan terus meningkat, mencapai nilai maksimum sekitar 0.4797. Namun, nilai akurasi validasi cenderung stagnan di sekitar 0.3306, mengindikasikan bahwa model mungkin mengalami overfitting, di mana performa pada data pelatihan lebih baik daripada pada data yang belum pernah dilihat sebelumnya.

Secara keseluruhan, hasil dari Tabel 1 menunjukkan bahwa meskipun ada peningkatan dalam akurasi pelatihan sepanjang iterasi, performa validasi masih belum stabil. Setelah melihat grafik akurasi dari model berdasarkan performa model pada tahap training maka, selanjutnya adalah menganalisa performa atau kinerja model berdasarkan hasil loss yang dapat dilihat pada gambar 4.14 berikut:



Gambar 4. 14 Hasil Loss Training dan Validasi

Dari hasil pelatihan yang sudah di dilakukan dapat dilihat pada gambar 4.17. bahwa hasil *loss* dari proses *training* di mana grafik loss menunjukkan tren perubahan nilai kehilangan (loss) pelatihan dan validasi selama 20 epoch. Pada awal pelatihan, terdapat fluktuasi pada nilai loss pelatihan dan validasi, yang mencerminkan proses adaptasi awal model terhadap data. Selanjutnya, seiring berjalannya epoch, terlihat bahwa nilai loss pelatihan cenderung menurun secara stabil. Ini mengindikasikan bahwa model semakin efektif dalam mengurangi kesalahan prediksi pada data pelatihan. Namun, seperti yang terlihat pada akurasi, dimana nilai loss validasi tidak selalu mengikuti tren yang sama dengan loss pelatihan.

Pada tabel 4.3 di bawah ini adalah hasil *loss* antara data uji dan data latih:

Tabel 4. 3 Hasil Loss Training dan Validasi

Epoch	***Training Loss***	**Validation Loss**
1	1.9455	1.9245
2	1.9404	1.9218
3	1.9333	1.9319
4	1.9319	1.9231
5	1.9242	1.9187
6	1.9275	1.9193
7	1.9266	1.9200
8	1.9165	1.9163

Epoch	*Training Loss*	Validation Loss
9	1.9110	1.9097
10	1.8911	1.8897
11	1.8482	1.8628
12	1.8126	1.8348
13	1.7730	1.7999
14	1.7260	1.7835
15	1.6579	1.7530
16	1.6231	1.7573
17	1.5735	1.7407
18	1.5364	1.7070
19	1.4387	1.6754
20	1.4021	1.6790

Dari tabel 4.3 dapat Selama lima epoch awal, terlihat fluktuasi yang signifikan pada nilai loss pelatihan dan loss validasi, mencerminkan fase adaptasi awal model terhadap dataset pelatihan. Namun, seiring berlanjutnya epoch, terlihat tren menurun secara stabil dalam nilai loss pelatihan, mencirikan kemampuan model dalam mengurangi error prediksi pada dataset pelatihan.

Nilai loss validasi juga menunjukkan tren penurunan, namun tidak selalu sejajar dengan loss pelatihan. Hal ini dapat mengisyaratkan kompleksitas pengenalan pola yang mungkin lebih sulit pada dataset validasi, serta potensi masalah generalisasi model. Adanya variasi atau stagnasi dalam loss validasi mungkin mengindikasikan kemungkinan overfitting, di mana model terlalu memfokuskan pada data pelatihan dan tidak dapat memadukan dengan baik pada data baru.

Secara keseluruhan, analisis dari Tabel 4.3 menggambarkan perubahan loss pelatihan dan loss validasi selama proses pelatihan model. Fluktuasi awal

mengindikasikan penyesuaian model awal, sementara penurunan stabil dalam loss pelatihan mencerminkan kemajuan dalam estimasi kesalahan prediksi. Variasi dalam loss validasi mengacu pada kompleksitas tugas dan potensi masalah generalisasi model, yang perlu dipertimbangkan dalam upaya pengembangan model yang lebih baik dan kemampuan prediksi yang lebih akurat. Analisis ini memberikan panduan penting dalam memahami dinamika pelatihan model dan memperbaiki kualitas prediksi dalam konteks klasifikasi ekspresi wajah.

Dari Tabel 4.2 dan Tabel 4.3, terlihat bahwa model mengalami peningkatan dalam akurasi pelatihan seiring berjalannya epoch, dan nilai loss pelatihan secara konsisten menurun. Namun, hasil akurasi validasi dan loss validasi tidak selalu mengikuti tren yang sama, menunjukkan tantangan dalam menggeneralisasi model pada data yang belum pernah dilihat sebelumnya. Variasi ini dapat mengindikasikan potensi overfitting atau kesulitan dalam penyesuaian pada data validasi. Sehingga diperoleh analisa, meskipun terdapat kemajuan dalam akurasi pelatihan dan pengurangan loss pelatihan, penilaian terhadap performa model dalam hal generalisasi dan mencegah overfitting perlu menjadi perhatian dalam pengembangan lebih lanjut.

4.5. Pengujian Data Testing

Setelah dilakukan proses *training* dan dihasilkan sebuah model maka model tersebut akan dilakukan *pengujian*. Pengujian dilakukan untuk melihat apakah model tersebut dapat memprediksi data dengan benar. hasil dari pengujian yang sudah dilakukan dapat dilihat pada gambar 4.18.

Gambar 4. 15 Hasil Pengujian Menggunakan Data Testing

Dari hasil pengujian model menggunakan 30 data uji yang menghasilkan akurasi yang berbeda. Untuk melihat lebih jelas bisa dilihat pada tabel 4.4.

Tabel 4. 4 Hasil Pengujian Model CNN

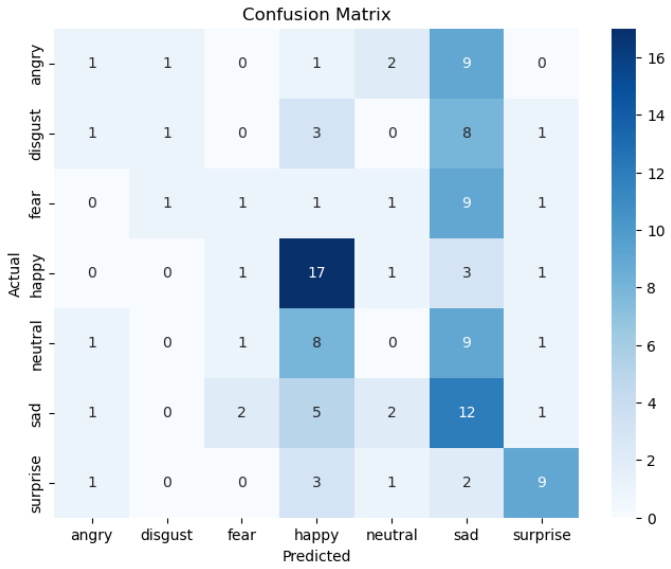
No	ACC	Prediction	Actual
1	43.79%	Sad	Disgust
2	29.47%	Surprise	Disgust
3	50.40%	Neutral	Disgust
4	50.30%	Sad	Disgust
5	22.08%	Disgust	Disgust
6	69.26%	Neutral	Surprise
7	39.76%	Sad	Surprise
8	63.81%	Angry	Surprise
9	39.08%	Angry	Surprise
10	60.03%	Happy	Surprise
11	29.09%	Sad	Fear

No	ACC	Prediction	Actual
12	33.35%	Surprise	Fear
13	48.19%	Sad	Fear
14	35.87%	Sad	Fear
15	79.23%	Sad	Fear
16	24.33%	Fear	Neutral
17	62.34%	Neutral	Neutral
18	27.09%	Sad	Neutral
19	22.62%	Neutral	Neutral
20	27.70%	Angry	Neutral
21	35.06%	Sad	Happy
22	69.78%	Neutral	Happy
23	25.59%	Sad	Happy
24	42.65%	Sad	Happy
25	54.53%	Neutral	Happy
26	31.41%	Sad	Sad
27	68.15%	Sad	Sad
28	57.66%	Disgust	Sad
29	40.78%	Angry	Sad
30	37.43%	Sad	Sad

Dari tabel di atas dapat dilihat bahwa baris yang berwarna kuning melambangkan hasil akurasi yang salah sedangkan tulisan yang berwarna putih menandakan hasil akurasi yang benar

4.6. *Confusion Matrix*

Berikut adalah hasil dari evaluasi *matrix* menggunakan *confusion matrix*.



Gambar 4. 16 Hasil confusion matrix

Pada gambar di atas peneliti mendapatkan informasi yang lebih detail terkait dengan performa model berdasarkan hasil training dan testing untuk setiap kelas ekspresi sebagai berikut:

1. Kelas "Angry" (Marah):

   True Positive (TP): Terdapat 1 kasus yang berhasil diprediksi dengan benar sebagai "Angry".

   False Negative (FN): Terdapat 1 kasus yang seharusnya adalah "Angry", namun diprediksi sebagai kelas lain.

   True Negative (TN): Tidak ada kasus yang seharusnya bukan "Angry" dan diprediksi dengan benar sebagai kelas lain.

False Positive (FP): Terdapat total 1+1+2+9=13 kasus yang salah diprediksi sebagai "Angry".

2. Kelas "Disgust" (Jijik):

   TP: Terdapat 1 kasus yang berhasil diprediksi dengan benar sebagai "Disgust".

   FN: Terdapat 3 kasus yang seharusnya adalah "Disgust", namun diprediksi sebagai kelas lain.

   TN: Tidak ada kasus yang seharusnya bukan "Disgust" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 1+1+8+1=11 kasus yang salah diprediksi sebagai "Disgust".

3. Kelas "Fear" (Takut):

   TP: Terdapat 1 kasus yang berhasil diprediksi dengan benar sebagai "Fear".

   FN: Terdapat 1 kasus yang seharusnya adalah "Fear", namun diprediksi sebagai kelas lain.

   TN: Tidak ada kasus yang seharusnya bukan "Fear" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 1+1+1+9=12 kasus yang salah diprediksi sebagai "Fear".

4. Kelas "Happy" (Senang):

   TP: Terdapat 17 kasus yang berhasil diprediksi dengan benar sebagai "Happy".

   FN: Tidak ada kasus yang seharusnya adalah "Happy" namun salah diprediksi sebagai kelas lain.

   TN: Terdapat 1+1+3+1=6 kasus yang seharusnya bukan "Happy" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 1+2+5+12+1=21 kasus yang salah diprediksi sebagai "Happy".

5. Kelas "Sad" (Sedih):

   TP: Terdapat 2 kasus yang berhasil diprediksi dengan benar sebagai "Sad".

   FN: Terdapat 5 kasus yang seharusnya adalah "Sad", namun diprediksi sebagai kelas lain.

   TN: Terdapat 1+2+12+1=16 kasus yang seharusnya bukan "Sad" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 2+3=5 kasus yang salah diprediksi sebagai "Sad".

6. Kelas "Surprise" (Kaget):

   TP: Terdapat 9 kasus yang berhasil diprediksi dengan benar sebagai "Surprise".

   FN: Terdapat 3 kasus yang seharusnya adalah "Surprise", namun diprediksi sebagai kelas lain.

   TN: Terdapat 1+3+1+2=7 kasus yang seharusnya bukan "Surprise" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 8+1+9=18 kasus yang salah diprediksi sebagai "Surprise".

7. Kelas "Neutral":

   TP: Terdapat 1 kasus yang berhasil diprediksi dengan benar sebagai "Neutral".

   FN: Terdapat 8 kasus yang seharusnya adalah "Neutral", namun diprediksi sebagai kelas lain.

   TN: Terdapat 1+1+1+9=12 kasus yang seharusnya bukan "Neutral" dan diprediksi dengan benar sebagai kelas lain.

   FP: Terdapat total 9+1=10 kasus yang salah diprediksi sebagai "Neutral".

Dari hasil *confusion matrix* di atas dapat diperoleh classification report yang berisi *precission*, *recall*, *f1 score* dan *support* yang dapat di lihat pada tabel di bawah ini.

Tabel 4. 5 Perbandingan Score Klasifikasi (Classification Report)

Classification	***Precision***	***Recall***	***F1-Score***	***Support***
Angry	0.2	0.07	0.11	14
Disgust	0.33	0.07	0.12	14
Fear	0.2	0.07	0.11	14
Happy	0.45	0.74	0.56	23
Neutral	0.29	0.09	0.13	23
Sad	0.04	0.12	0.06	16
Surprise	0.07	0.05	0.06	20

Tabel diatas menunjukan performa dari setiap kelas yang diukur berdasarkan nilai precision, recall dan f1-score serta distribusi dari setiap kelas sehingga didapatkan analisa sebagai berikut:

1. Kelas "Angry" (Marah):

   Kelas "Angry" memiliki precision yang rendah sebesar 0.20, menandakan bahwa model memiliki tingkat ketepatan yang rendah dalam mengidentifikasi ekspresi "Angry." Hal ini mengindikasikan bahwa banyak prediksi yang diberi label "Angry" ternyata salah dan sebenarnya termasuk dalam kelas lain. Recall juga rendah dengan nilai 0.07, menunjukkan bahwa model gagal mengidentifikasi sebagian besar ekspresi "Angry" yang sebenarnya ada dalam data. F1-score sebesar 0.11 mencerminkan keseimbangan antara precision dan recall, dan nilai ini menunjukkan bahwa model masih memiliki kinerja yang rendah dalam mengklasifikasikan ekspresi "Angry." Dengan jumlah support sebesar 14, dapat disimpulkan bahwa kelas "Angry" memiliki jumlah data yang terbatas, yang mungkin menjadi salah satu faktor penurunan performa model.

2. Kelas "Disgust" (Jijik):

   Kelas "Disgust" memiliki precision yang lebih tinggi dibandingkan dengan "Angry," namun tetap rendah sebesar 0.33. Ini menunjukkan bahwa model memiliki tingkat ketepatan yang sedikit lebih baik dalam mengklasifikasikan ekspresi "Disgust," meskipun masih ada sejumlah prediksi yang salah. Recall sebesar 0.07 mengindikasikan bahwa model gagal mengenali sebagian besar ekspresi "Disgust" yang seharusnya termasuk dalam data. F1-score yang rendah sebesar 0.12 mencerminkan kesulitan dalam mencapai keseimbangan antara precision dan recall untuk kelas ini. Seperti kelas "Angry," jumlah support yang rendah (14) juga dapat mempengaruhi performa model dalam mengklasifikasikan kelas "Disgust."

3. Kelas "Fear" (Takut):

   Kelas "Fear" memiliki karakteristik serupa dengan kelas "Angry" dan "Disgust." Precision, recall, dan F1-score semuanya rendah dengan nilai masing-masing 0.20, 0.07, dan 0.11. Hal ini menunjukkan bahwa model memiliki tantangan dalam mengklasifikasikan ekspresi "Fear" dengan tepat. Jumlah support yang rendah (14) juga menjadi faktor yang mempengaruhi performa model terhadap kelas "Fear."

4. Kelas "Happy" (Senang):

   Kelas "Happy" memiliki precision yang lebih baik dibandingkan dengan kelas-kelas sebelumnya, sebesar 0.45. Ini mengindikasikan bahwa model memiliki tingkat ketepatan yang lebih baik dalam mengidentifikasi ekspresi "Happy." Recall yang lebih tinggi, yaitu 0.74, menunjukkan bahwa model berhasil mengenali sebagian besar ekspresi "Happy" yang ada dalam data. F1-score sebesar 0.56 mencerminkan kinerja yang lebih baik dalam mencapai keseimbangan antara precision dan recall. Jumlah support sebesar 23 juga lebih besar dibandingkan dengan kelas sebelumnya, yang dapat membantu meningkatkan performa model.

5. Kelas "Sad" (Sedih):

   Kelas "Sad" memiliki precision sebesar 0.29, menunjukkan bahwa model memiliki tingkat ketepatan yang cukup rendah dalam mengklasifikasikan

ekspresi "Sad." Recall yang rendah, yaitu 0.09, mengindikasikan bahwa model kesulitan dalam mengenali sebagian besar ekspresi "Sad" yang seharusnya ada dalam data. F1-score sebesar 0.13 mencerminkan tantangan dalam mencapai keseimbangan antara precision dan recall untuk kelas ini. Jumlah support sebesar 23 juga tergolong cukup rendah, yang mungkin berkontribusi pada performa rendah model.

6. Kelas "Surprise" (Kaget):

   Kelas "Surprise" memiliki precision yang sangat rendah, sebesar 0.04. Hal ini menunjukkan bahwa model memiliki tingkat ketepatan yang sangat rendah dalam mengklasifikasikan ekspresi "Surprise." Recall yang sedikit lebih tinggi, yaitu 0.12, menunjukkan bahwa model berhasil mengenali sebagian dari ekspresi "Surprise," tetapi masih banyak yang terlewatkan. F1-score yang sangat rendah sebesar 0.06 mencerminkan kinerja yang rendah dalam mencapai keseimbangan antara precision dan recall. Jumlah support sebesar 16 juga terbatas, yang mungkin mempengaruhi performa model terhadap kelas "Surprise."

7. Kelas "Neutral":

   Kelas "Neutral" memiliki precision yang rendah sebesar 0.07, menunjukkan bahwa model memiliki tingkat ketepatan yang rendah dalam mengklasifikasikan ekspresi "Neutral." Recall yang juga rendah, yaitu 0.05, mengindikasikan bahwa model gagal mengenali sebagian besar ekspresi "Neutral" yang seharusnya ada dalam data. F1-score sebesar 0.06 mencerminkan tantangan dalam mencapai keseimbangan antara precision dan recall untuk kelas ini. Jumlah support sebesar 20 juga terbatas, yang dapat mempengaruhi performa model terhadap kelas "Neutral."

Secara keseluruhan, Classification Report menunjukkan bahwa model memiliki kinerja yang rendah dalam mengklasifikasikan hampir semua kelas ekspresi. Terdapat tantangan dalam mencapai keseimbangan antara precision dan recall, serta dalam mengatasi jumlah data yang terbatas dalam beberapa kelas.

4.7. Hasil Implementasi

Setelah pengembangan model analisa ekspresi wajah, maka tahap selanjutnya adalah melakukan uji coba pada model yang telah dikembangkan. Pengujian ini akan dilakukan dengan 30 data uji. Hasil dari pengujian tersebut akan dicatat jumlah prediksi data yang benar dan jumlah prediksi data yang salah, data tersebut akan digunakan untuk mengetahui nilai akurasi model ketika dilakukannya uji coba dengan menggunakan 30 data. Hasil dari uji coba yang dilakukan dapat dilihat pada tabel 4.6.

Tabel 4. 6 Hasil Pengujian Akurasi Model CNN

Benar	6
Salah	24
Total data	30

Sehingga untuk menghitung akurasi pada data pengujian dapat menggunakan persamaan 2.5. Berikut adalah proses perhitungan untuk mencari nilai akurasi.

$$accuracy = \frac{6}{6 + 24} = 0.20$$

Dari hasil perhitungan di atas dapat dilihat bahwa akurasi dari prediksi ekpresi wajah menggunakan model yang sudah di implementasikan ke dalam model analisis ekspresi wajah sebesar 20%. Jika dilihat dari besaran akurasi model tersebut dapat dikategorikan sangat membutuhkan perbaikian untuk meningkatkan hasil prediksi atau performa dari model yang telah dibuat.

4.8. Kelemahan Sistem

Berdasarkan analisa yang dilakukan dalam penelitian ini maka kelemahan sistem yang telah dikembangkan dapat dijelasakan sebagai berikut:

1. Jumlah Data Training yang Tidak Seimbang:

   Jumlah data training untuk setiap kelas emosi tidak seimbang. Beberapa kelas memiliki jumlah data training yang jauh lebih banyak daripada yang lain. Ketidakseimbangan ini dapat menyebabkan model cenderung lebih baik dalam mengenali kelas-kelas yang memiliki lebih banyak data training dan lebih buruk dalam mengenali kelas-kelas dengan data training yang sedikit.
2. Jumlah Data Testing yang Terbatas:

   Jumlah data testing yang terbatas untuk setiap kelas juga dapat mempengaruhi kualitas evaluasi model. Data testing yang terbatas dapat menyebabkan hasil evaluasi menjadi kurang dapat diandalkan.
3. Model Overfitting:

   Dalam beberapa epoch awal, model mungkin mengalami overfitting pada data training. Ini ditunjukkan oleh perbedaan antara akurasi pada data training dan data validasi yang semakin membesar seiring berjalannya waktu.
4. Performa yang Rendah:

   Akurasi model pada data testing cukup rendah (sekitar 33%). Ini menunjukkan bahwa model mungkin belum sepenuhnya mampu memahami dan memprediksi emosi dari gambar wajah siswa.
5. Kemampuan Model yang Terbatas:

   Arsitektur model yang Anda gunakan mungkin kurang kompleks untuk tugas yang kompleks seperti pengenalan emosi. Model hanya memiliki beberapa lapisan konvolusi dan lapisan terhubung, yang mungkin tidak cukup kuat untuk mengekstraksi fitur-fitur penting dari data gambar dengan baik.

6. Kesulitan dalam Mendeteksi Beberapa Emosi:

   Beberapa kelas emosi (seperti "disgust" dan "surprise") mungkin lebih sulit untuk diprediksi daripada yang lain. Hal ini tercermin dalam confusion matrix dan classification report, di mana akurasi untuk beberapa kelas sangat rendah.

7. Ketidaksesuaian Model:

   Model mungkin tidak sepenuhnya sesuai untuk tugas ini. Mungkin perlu dilakukan penelitian lebih lanjut untuk memilih atau merancang arsitektur model yang lebih sesuai dengan tugas pengenalan emosi dari gambar wajah.