EKSPRESI WAJAH MULTIKATEGORI: PENDEKATAN PENGENALAN DENGAN *TRANSFER LEARNING*

SKRIPSI

Diajukan untuk Memenuhi Tugas Akhir dan Melengkapi Syarat Guna
Memperoleh Gelar Sarjana Strata Satu(S-1) dalam
Teknologi Informasi

Diajukan oleh :

MUHAMMAD BIMO GINANTOKO
NIM : 2008096033

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI WALISONGO SEMARANG
TAHUN 2024

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Muhammad Bimo Ginantoko
NIM : 2008096033
Jurusan : Teknologi Informasi

Menyatakan bahwa skripsi yang berjudul:

Ekspresi Wajah Multikategori:Pendekatan Pengenalan Dengan *Transfer Learning*

Secara keseluruhan adalah penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 2 Mei 2024
Pembuat pernyataan

Muhammad Bimo Ginantoko
NIM. 2008096033

PENGESAHAN

KEMENTRIAN AGAMA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof. Dr. Hamka Ngaliyan Semarang
Telp.024-7601295 Fax.7615387

PENGESAHAN

Naskah skripsi berikut ini:

Judul : EKSPRESI WAJAH MULTIKATEGORI: PENDEKATAN PENGENALAN DENGAN *TRANSFER LEARNING*
Penulis : **Muhammad Bimo Ginantoko**
NIM : 2008096033
Jurusan : Teknologi Informasi

Telah diujikan dalam sidang tugas akhir oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam Teknologi Informasi.

Semarang, 29 April 2024

DEWAN PENGUJI

Penguji I

Nur Cahyo Hendrowibowo, S.T., M.T.
NIP. 197312222006041001

Penguji II

Dr. Khotibul Umam, M.Kom.
NIP.197908272011011007

Penguji III

Hery Mustofa, M.Kom.
NIP. 198703172019031007

Penguji IV

Adzhal Arwani Mahfudh, M.Kom.
NIP. 199107032019031006

Pembimbing I

Dr. Khotibul Umam, M.Kom.
NIP.197908272011011007

Pembimbing II

Dr. Masy Ari Ulinuha, ST., M.T.
NIP. 198108122011011007

NOTA DINAS

Semarang, 1 April 2024

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan :

Judul : PENGENALAN EKPSRESI WAJAH MULTIKATEGORI: PENDEKATAN PENGENALAN DENGAN TRANSFER LEARNING
Nama : **Muhammad Bimo Ginantoko**
NIM : 2008096033
Jurusan : Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. Wr. Wb.

Pembimbing I,

Dr. Khotibul Umam, ST., M.Kom
NIP. 19790827 201101 1007

NOTA DINAS

Semarang, 1 April 2024

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum. Wr. Wb.

Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan :

Judul : PENGENALAN EKPSRESI WAJAH MULTIKATEGORI: PENDEKATAN PENGENALAN DENGAN TRANSFER LEARNING
Nama : **Muhammad Bimo Ginantoko**
NIM : 2008096033
Jurusan : Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo Semarang untuk diujikan dalam Sidang Munaqosyah.

Wassalamu'alaikum. Wr. Wb.

Pembimbing II,

Dr. Masy Ari Ulinuha, M.T.
NIP. 198108122011011007

LEMBAR PERSEMBAHAN

Puji dan Syukur Penulis ucapkan kepada Allah SWT, laporan tugas akhir ini dapat terselesaikan dengan baik. Karya tulis ini penulis persembahkan untuk :

1. Bapak Sulthoni dan Ibu Widiastuti sebagai orangtua penullis.
2. Saudara dan saudari dari penulis.
3. Segenap dosen Jurusan Teknologi Informasi.
4. Teman – teman Teknologi Informasi 2020.
5. Almamater Universitas Islam Negeri Walisongo Semarang.

MOTTO

Sesungguhnya Allah beserta orang-orang yang bertakwa dan orang-orang yang berbuat kebaikan."

(QS. An-Nahl: 128)

ABSTRAK

Ekspresi wajah merupakan contoh bentuk karakteristik perilaku seseorang untuk mengetahui emosi apa yang sedang terjadi atau dirasakan oleh setiap individu. Adanya berbagai macam ekspresi wajah maka akan terjadi kesulitan dalam mendeteksinya maka diperlukan sebuah *Machine Learning* yang mampu digunakan dalam bidang ini yaitu itu *Transfer Learning.* Penggunaan metode Transfer Learning dalam pengenalan ekspresi wajah ini bertujuan agar tidak kesulitan ketika menemukan ekspresi-ekspresi baru yang belum ada pada databasenya.Proses awal yang dilakukan pada penelitian ini yaitu dengan proses data *collection* dimana data yang digunakan adalah data FER. Selanjutnya dilakukan proses data *preparation* yang meliputi normalisasi data, *split* data yang menggunakan perbandingan 80:20, dan data *augmentation*. Kemudian dilanjutkan dengan melakukan perancangan dan pelatihan model. Setelah berhasil dengan adanya performa model tersebut maka perlu dilakukan sebuah pengujian dan pengevaluasian model yang bertujuan untuk mengetahui ketepatan suatu sistem. Berdasarkan hasil penelitian yang telah dilakukan Penggunaan metode *Transfer Learning* dalam melakukan Pengenalan Ekspresi Wajah Multikategori dengan model arsitektur *MobileNetV2* menghasilkan performa yang cukup baik dengan rincian nilai *F1-score* sebesar 90%, nilai *recall* sebesar 89%, nilai *precision* sebesar 90%, dan untuk nilai akurasi yang dihasilkan sebesar 90%.

Kata Kunci : Ekspresi Wajah, *Transfer Learning, MobileNetV2*

KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Allah SWT yang melimpahkan rahmat kepada penulis sehingga dapat menyelesaikan tugas akhir ini dengan judul " **Ekspresi Wajah Multikategori: pendekatan Pengenalan Dengan *Transfer Learning***" yang digunakan sebagai syarat untuk kelulusan dalam menempuh Pendidikan di Program Studi S1 Teknologi Informasi Universitas Islam Negeri Walisongo Semarang.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada beberapa pihak yang telah banyak memberikan bantuan, dukungan dan juga nasehat kepada penulis sehingga terselesaikannya tugas akhir ini kepada :

1. Kedua orang tua penulis bapak H.M Sulthoni dan ibu Hj. Widiastuti, S.E. yang selalu memberikan do'a, dukungan dan support kepada penulis sehingga tugas akhir ini terselesaikan.
2. Ketua Program Studi Teknologi Informasi UIN Walisongo Semarang, Bapak Nur Cahyo Hendro Wibowo, S.T., M.Kom.
3. Dosen Pembimbing I sekaligus Dosen Wali, Bapak Dr.Khotibul Umam, ST., M.Kom, yang telah memberikan arahan dan bimbingan dalam tugas akhir ini.

4. Dosen Pembimbing II, Bapak Dr.Masy Ari Ulinuha, S.T., M.T, yang telah memberikan arahan dan bimbingan dalam tugas akhir ini.
5. Seluruh Dosen Program Studi Teknologi Informasi serta seluruh dosen dan Staff akademik di Universitas Islam Negeri Walisongo Semarang yang telah berkontribusi dan memberikan ilmu pengetahuan selama masa pendidikan.
6. Dan semua pihak yang tidak dapat disebutkan satu per satu sehingga penyusunan tugas akhir ini sehingga dapat selesai dengan baik.

Penulis berharap tugas akhir ini dapat memberikan manfaat bagi semua pihak dan bisa dijadikan sumber referensi bagi penelitian berikutnya.

Semarang, 5 April 2024

Penulis

DAFTAR ISI

PERNYATAAN KEASLIAN ..... **ii**
PENGESAHAN ..... **iii**
NOTA DINAS ..... **iv**
NOTA DINAS ..... **v**
LEMBAR PERSEMBAHAN ..... **vi**
MOTTO ..... **vii**
ABSTRAK ..... **viii**
KATA PENGANTAR ..... **ix**
DAFTAR ISI ..... **xi**
DAFTAR GAMBAR ..... **xiv**
DAFTAR TABEL ..... **xvi**
BAB I PENDAHULUAN **1**
A.Latar Belakang 1
B.Identifikasi Masalah 5
C.Rumusan Masalah 5
D.Tujuan Penelitian 6
E.Batasan Masalah 6
F.Manfaat Penelitian 7
BAB II LANDASAN PUSTAKA **8**
A.Kajian Teori 8
1.*Ekspresi Wajah* 8
2.*Citra* 10
3.Machine Learning 11

4.Transfer Learning 11
5.MobileNetV2 13
6.Tensorflow 15
7.Convolutional Neural Network (CNN) 16
a.Convolution Layer *18*
b.Stride *19*
c.Padding *20*
d.Activation Layer *21*
e.Pooling Layer *22*
f.Fully-Connected Layer *22*
8.Evaluasi model 23
a.Learning Curve *23*
b.Confusion Matrix *24*
B.Kajian Penelitian yang Relevan 28
BAB III METODOLOGI PENELITIAN 31
A.Jenis Penelitian 31
B.Jenis dan Sumber data 31
C.Metode 32
*1.*Data *preparation* 33
2.Modeling 35
3.Evaluasi model 40
BAB IV HASIL DAN PEMBAHASAN 41
A.Data *Collection* 41
B.Data *Preparation* 43
1.Inisialisasi *Array* Dan Normalisasi 43
2.*Split* Data 47

3.Data *Augmentation* 49
C.Modeling 51
1.Perancangan Model 51
2.Pelatihan Model 54
3.Pengujian Pada Nilai *True* 57
4.Pengujian Pada Nilai *False* 61
D.Evaluation 64
1.Learning Curve 64
2.Akurasi, Precision, Recall dan F1-Score 65
BAB V KESIMPULAN DAN SARAN 79
A.Kesimpulan 79
B.Saran 80
DAFTAR PUSTAKA 81
LAMPIRAN 85

DAFTAR GAMBAR

Gambar 2. 1 Contoh ekspresi wajah 10
Gambar 2. 2 Ekspresi wajah dalam bentuk citra 11
Gambar 2. 3 Arsitektur *Transfer Learning* 12
Gambar 2. 4 Arsitektur *MobileNetV2* 14
Gambar 2. 5 Arsitektur CNN 17
Gambar 2. 6 *Learning Curve* 24
Gambar 2. 7 *Confusion Matrix* 25
Gambar 3. 1 Alur Metode 32
Gambar 3. 2 *MobileNetV2* Model 35
Gambar 4. 1 Proses *Data Collection* 41
Gambar 4. 2 Hasil dari inisialisai variabel 42
Gambar 4. 3 Pendefinisian kelas 43
Gambar 4. 4 *Inisialisai Array* untuk gambar dan label 44
Gambar 4. 5 Hasil dari *inisialisasi array* 45
Gambar 4. 6 Proses Normalisasi data 45
Gambar 4. 7 Contoh Hasil Gambar yang Telah di Normalisasi 46
Gambar 4. 8 Proses *Split data* 47
Gambar 4. 9 Hasil proses *split data* 48
Gambar 4. 10 Proses *Augmentation data* 49
Gambar 4. 11 Contoh hasil Gambar setelah proses *Data Augmentation* 50
Gambar 4. 12 Pembuatan variabel parameter 51
Gambar 4. 13 Proses Perancangan model 52
Gambar 4. 14 Hasil Perancangan Model 53
Gambar 4. 15 Proses *Compile* 54
Gambar 4. 16 Proses Pelatihan Model *(Fit)* 55
Gambar 4. 17 Hasil Proses Pelatihan Model 56
Gambar 4. 18 Proses klasifikasi data yang benar 57
Gambar 4. 19 Contoh Gambar yang diprediksi benar pada kelas *Happy* 59

Gambar 4. 20 Contoh Gambar yang diprediksi benar pada kelas *Sad* 59
Gambar 4. 21 Contoh Gambar yang diprediksi benar pada kelas *Surprise* 60
Gambar 4. 22 Proses klasifikasi data yang salah 61
Gambar 4. 23 Contoh Gambar yang diprediksi salah pada kelas *Happy* 62
Gambar 4. 24 Contoh Gambar yang diprediksi salah pada kelas *Sad* 63
Gambar 4. 25 Contoh Gambar yang diprediksi salah pada kelas *Surprise* 64
Gambar 4. 26 Proses Visualisasi *Learning Curve* 65
Gambar 4. 27 Proses Visualisasi *Confusion Matrix* 66
Gambar 4. 28 Hasil Visualisasi *Confusion Matrix* 67
Gambar 4. 29 Hasil *Classification Report* 68

DAFTAR TABEL

Tabel 2. 1 Rujukan Penelitian....28
Tabel 3. 1 Jumlah Dataset....32
Tabel 4. 1 Tabel *Confusion Matrix*....69
Tabel 4. 2 Perhitungan Kelas *Happy*....70
Tabel 4. 3 Perhitungan Kelas *Sad*....72
Tabel 4. 4 Perhitungan Kelas *Surprise*....74
Tabel 4. 5 Hasil Perhitungan Performa Setiap Kelas....76
Tabel 4. 6 Perhitungan performa model....77

BAB I
PENDAHULUAN

A. Latar Belakang

Penggunaan ilmu pengetahuan dalam teknologi pengolahan citra digital telah mengalami pertumbuhan yang sangat cepat sejak diperkenalkannya komputer yang mampu menjalankan algoritma pada tahun 1960. (Arhandi *et al*., 2018). Dengan perkembangan teknologi pengolahan citra dalam berbagai bidang, semakin banyak permasalahan yang dapat diatasi. Allah Subhanahu Wa Ta'ala berfirman dalam QS. Ali 'Imran 3: Ayat 191 yang berbunyi:

رَبَّنَا مَا خَلَقْتَ هٰذَا بَا طِلًا ۚ سُبْحٰنَكَ فَقِنَا عَذَا بَ النَّا رِ

Artinya: *"Ya Tuhan kami, tidaklah Engkau menciptakan semua ini sia-sia; Maha Suci Engkau, lindungilah kami dari azab neraka."*

Berdasarkan firman Allah SWT diatas menjelaskan segala penciptaan-Nya merupakan sebuah tanda kekuasaan dari Allah SWT yang tidak lah sia-sia.

Sebagai contoh, teknologi absensi berbasis sidik jari yang banyak diterapkan oleh perusahaan dan instansi pemerintah. Keunggulan teknologi ini terletak pada kemudahan penggunaannya, menghilangkan kebutuhan untuk absensi manual. Namun, dari perspektif teknologi,

pengenalan dan pencocokan sidik jari dengan data yang ada memerlukan metode yang canggih dan efisien. Hal ini sangat penting sekali karena jika terjadi kesalahan baik dalam mengenali ataupun mencocokkan data maka akan terjadi masalah dan merugikan beberapa pihak. Oleh karena itu, saat ini bukan hanya teknologi citra saja yang berkembang tetapi metode – metode *recognition* yang melatar belakangi teknologi pengolahan citra juga berkembang (Mubarok, 2019).

Metode *recognition* ini sangatlah penting perannya ketika diaplikasikan sesuai dengan kebutuhannya. Karna pasti banyak sekali masalah yang akan terselesaikan ketika metode recognition ini berhasil diaplikasikan dan memiliki performa yang bagus (Chandra, 2019). Tentu saja, untuk menjalankan metode *recognition* dengan baik, diperlukan perangkat komputasi yang kuat. Sebagai contoh, dalam bidang robotika, pengembangan kemampuan robot untuk mengenali wajah seseorang adalah salah satu pencapaian penting. Kemampuan ini dapat sangat membantu berbagai aspek pekerjaan dan penggunaan robot. Di sisi lain, dalam konteks keamanan, CCTV telah membantu banyak orang dengan merekam aktivitas di sudut-sudut tertentu. Namun, pertanyaan yang muncul adalah bagaimana kita dapat mengembangkan CCTV agar tidak hanya merekam,

tetapi juga mampu mengenali objek, mendeteksi ancaman, serta mencegah tindak pencurian. Tentunya itu bukanlah hal mudah, membutuhkan kemampuan *machine learning* yang sangat bagus untuk mewujudkannya. Jika kebutuhan sistem sampai pada taraf untuk mengenali persepsi ancaman, maka yang dibutuhkan sebuah metode pengenalan yang tidak hanya mampu mengenali wajah saja, tetapi juga bisa mengenali ekspresi wajah dari seseorang (Mubarok, 2019).

Ekspresi wajah menunjukkan keadaan emosi seseorang yang dilihat melalui ekspresi dapat menjadi penunjang keputusan dalam suatu tindakan terhadap seseorang. Dan juga ekspresi seseorang bersifat universal karena perbedaan wajah dari bentuknya, ras suku, dan warna kulit. Ekman dan Friesen mengkonfirmasi adanya teori darwin dan mengklasifikasi pada akhir abad ke – 20 bahwasannya terdapat enam ekspresi secara umum yang terdiri dari bahagia, takut, terkejut, jijik, sedih, netral dan marah. Hal ini tentunya akan menjadi tantangan tersendiri, karena untuk mengenali wajah hingga taraf ekspresi tertentu membutuhkan sebuah algoritma yang mampu mengenali ciri suatu citra secara detail (Ab Wahab et al., 2021).

Bukan hanya itu saja, dikarenakan setiap orang memiliki ekspresi yang berbeda-beda, maka dengan itu dibutuhkanlah *dataset* besar yang berisi ekspresi berbeda beda pada setiap kategorinya. Agar algoritma yang digunakan nanti memiliki variasi dan perbandingan data yang cukup banyak sehingga algoritma tidak kesulitan ketika menemukan ekspresi-ekspresi baru yang belum ada pada databasenya. Disinilah tantangan yang harus dipecahkan oleh peneliti untuk menemukan sebuah algoritma yang dapat mengenali berbagai ekspresi wajah. Demi menjangkau itu semua dibutuhkan sebuah *machine learning* yang mampu mempelajari itu secara lebih dalam yaitu menggunakan *Transfer Learning.*

Transfer Learning adalah sebuah metode yang menggunakan model yang sudah ada dilatih sebelumnya *(pre-trained)* sebagai dasar untuk melatih model yang baru. Model ini cocok digunakan untuk mengenali berbagai ekspresi wajah seseorang (Abas *et al*., 2018).

Dengan demikian penelitian kali ini berfokus pada Bagaimana membangun sebuah model untuk pengenalan ekspresi wajah multikategori menggunakan metode *Transfer Learning.* Pemanfaatan model pengenalan ekspresi wajah multikategori ini berguna untuk mengidentifikasi emosi seseorang yang dimana jika model

ini berhasil maka bisa dikembangkan dibidang keamanan dan robotika.

B. Identifikasi Masalah

Berdasarkan latar belakang diatas, dapat diidentifikasi beberapa masalah yang menjadi fokus dari penelitan ini:

1. Adanya ekspresi yang bersifat *universal* sehingga membutuhkan model *machine learning* untuk mengenali wajah hingga taraf ekspresinya.
2. Dalam penerapan model *transfer learning* ini menggunakan dataset yang sudah dilatih dan diberi label sebelumnya sehingga mempermudah proses pengolahan data dibandingkan dengan penelitian terdahulu yang menggunakan metode lain.

C. Rumusan Masalah

Berdasarkan latar belakang diatas, maka peneliti mengambil rumusan masalah yaitu:

1. Bagaimana membangun model untuk pengenalan ekspresi wajah multikategori menggunakan metode *Transfer Learning*?
2. Bagaimana performa model pengenalan ekspresi wajah multikategori dengan menggunakan metode *Transfer Learning*?

D. Tujuan Penelitian

Adapun tujuan dari penelitian ini yaitu:

1. Membangun model pengenalan ekspresi wajah multikategori dengan menggunakan metode *Transfer Learning*.
2. Mengetahui performa model pengenalan ekspresi wajah multikategori dengan menggunakan metode *Transfer Learning*.

E. Batasan Masalah

Dalam penelitian ini menggunakan batasan masalah yang digunakan dalam proses penelitian agar dapat dilakukan secara jelas dan memiliki cakupan yang tidak meluas. Batasan masalah pada penelitian ini sebagai berikut:

1. Dalam penelitian ini menggunakan dataset yang diambil dari kaggle dengan jumlah 15057data yang memiliki format .jpg dan dengan ukuran pixel 48 x 48
2. Penelitian ini menggunakan Metode *Transfer Learning* serta menggunakan *library TensorFlow* dan model arsitektur *MobileNetV2.*

F. Manfaat Penelitian

1. Manfaat Praktis

   Menghasilkan perangkat lunak yang dapat membantu dalam mengidentifikasi emosi seseorang melalui ekspresi wajah.

2. Manfaat Teoritis

   Model pengenalan ekspresi wajah multikategori ini dapat dikembangkan kedalam CCTV dan dibidang robotika untuk mendeteksi emosi seseorang melalui ekspresi wajah manusia.

BAB II
LANDASAN PUSTAKA

A. Kajian Teori

1. Ekspresi Wajah

Ekspresi merupakan bentuk komunikasi yang bersifat nonverbal yang dihasilkan dari gerakan atau posisi otot yang terdapat pada wajah yang bertujuan untuk menyampaikan sebuah keadaan dari emosi seseorang terhadap orang yang sedang berinteraksi. Dalam mengungkapkan ekspresi wajah maka seseorang dapat menilai yang terlintas terhadap lawan interaksinya. Dengan adanya ekspresi wajah maka akan dapat mengetahui emosi apa yang sedang terjadi pada setiap individu (Abidin, 2011). Ekspresi wajah merupakan contoh bentuk karakteristik perilaku seseorang. Berikut merupakan contoh fitur dari masing – masing ekspresi wajah:

a. Ekspresi wajah dalam kategori Senang atau *Happy* maka ditunjukan dengan ekspresi wajah yang memberikan senyuman dan juga kadang memperlihatkan gigi.
b. Ekspresi wajah dalam kategori Marah atau *Angry* maka dapat diekspresikan dengan

keadaan mulut terbuka seperti dalam keadaan berteriak dan keadaan mata yang terlihat kaku seperti menatap dengan tajam terhadap lawan bicara.

c. Ekspresi wajah yang dikategorikan Takut atau *Fear* maka dapat ditunjukkan dengan keadaan bentuk mata yang terlihat takut dan ragu dalam menatap objek yang dilihat.
d. Ekspresi wajah dalam kategori Netral dapat digambarkan dengan keadaan tidak merasakan emosi apapun sehingga ekspresi yang diberikan merupakan ekspresi dalam bentuk muka yang datar.
e. Ekspresi wajah yang dikategorikan Sedih atau *Sad* maka dapat ditunjukkan dengan keadaan bentuk bibir sedikit tertarik kebawah dan mata dalam keadaan yang tidak fokus dalam menatap sebuah objek.
f. Ekspresi wajah dalam kategori Terkejut atau *Surprise* dapat ditunjukkan dengan keadaan tatapan mata tajam terhadap objek dan mulut menganga atau terbuka lebar.
g. Ekspresi wajah dalam kategori Jijik atau *Disguest* maka dapat ditunjukkan dengan

keadaan bentuk bibir yang ditarik kebawah dan mata mentap tajam objek yang membuat mereka jijik (Amaanullah et al., 2022).

Gambar 2. 1 *Contoh ekspresi wajah*

Ekspresi wajah merupakan cara efektif yang digunakan untuk mengenali emosi manusia. Dalam kehidupan sehari-hari penggunaan ekspresi wajah sangat penting unruk berkomunikasi karena dalam berkomunikasi seseorang dapat menyampaikan emosi dan juga perasaan yang non-verbal (Ab Wahab *et al.*, 2021).

2. Citra

Citra adalah salah satu bagian komponen dari multimedia yang memiliki peranan penting karena berisi informasi dalam bentuk visual. Citra ini

memiliki banyak informasi dalam bentuk tulisan (Abdi & Aisyah, 2011).

Gambar 2. 2 *Ekspresi wajah dalam bentuk citra*

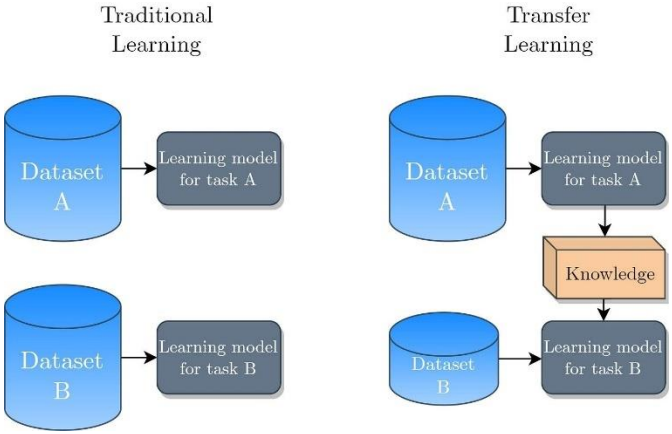
3. Machine Learning

Machine Learning merupakan bidang keilmuan yang berfokus untuk mengembangkan sebuah sistem yang dapat belajar sendiri menggunakan data tanpa harus memprogram berkali-kali. Sehingga program komputer bersifat dinamis bergantung dengan kelengkapan data yang dimiliki. *Machine learning* sangat berguna untuk menyelesaikan kasus yang berubah-ubah dalam waktu tertentu, dan salah satu contohnya adalah mengenali ekspresi wajah (AfrizaL *et al.*, 2022).

4. Transfer Learning

Di era digital seperti sekarang, data adalah salah satu aset yang sangat berharga. Sayangnya, terkadang kita tidak memiliki akses ke dataset berukuran besar. Keterbatasan data ini dapat

menghambat kemampuan kita untuk membangun model yang akurat seperti yang dapat dihasilkan oleh dataset yang besar. Namun, jika kita membandingkannya dengan cara manusia belajar, manusia tidak selalu belajar dari awal. Sebagai contoh, jika seseorang sudah tahu cara naik sepeda, belajar cara mengendarai sepeda motor tidak akan dimulai dari nol. Mereka sudah memiliki pengetahuan sebelumnya yang dapat mereka transfer dari pengalaman sepeda. Inilah yang memicu pengembangan teknik *Transfer Learning* (Hanif, 2022).



Gambar 2. 3 *Arsitektur Transfer Learning*

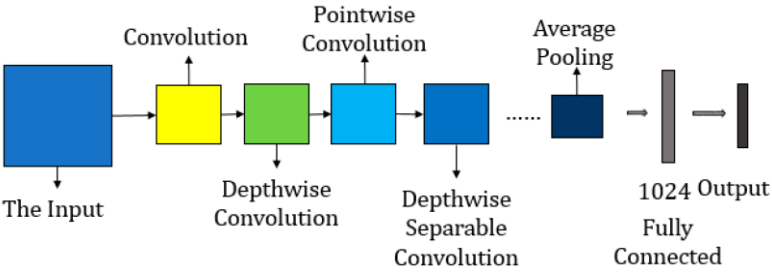
Transfer learning adalah pendekatan di mana kita menggunakan model yang telah dilatih sebelumnya untuk dataset tertentu dan kemudian

menerapkannya pada dataset yang berbeda. Model ini sering disebut sebagai *pre-trained* model karena telah dilatih dengan dataset yang lebih umum sebelumnya. Suatu *pre-trained* model adalah model yang sebelumnya telah dilatih pada suatu dataset dan telah memiliki bobot dan bias yang merepresentasikan fitur dari dataset yang digunakan untuk melatihnya. Fitur yang telah dipelajari tersebut dapat ditransfer ke data yang berbeda (Oumina *et al*., 2020).

5. MobileNetV2

MobileNetV2 adalah salah satu arsitektur *Convolutional Neural Network* (CNN) yang telah dirancang untuk menangani kebutuhan komputasi yang lebih efisien. *MobileNetV2* mengadopsi konsep *Depthwise Separable Convolution*, yang terdiri dari dua jenis konvolusi, yaitu *depthwise convolution* dan *pointwise convolution*. *Depthwise convolution* menggantikan konvolusi standar dengan melakukan konvolusi pada setiap saluran input secara terpisah. Kemudian, *pointwise convolution* digunakan untuk menggabungkan hasil dari *depthwise convolution*. Pendekatan ini mengurangi jumlah parameter dan operasi komputasi yang diperlukan, sehingga membuat jaringan menjadi lebih efisien. Selain itu, model *MobileNetV2* biasanya tersedia dalam berbagai

versi yang telah dilatih sebelumnya dengan dataset besar seperti *ImageNet.* Ini memungkinkan pengguna untuk menerapkan *transfer learning* dengan menggunakan model yang sudah memiliki pengetahuan tentang berbagai objek dan fitur visual (Zaelani & Miftahuddin, 2022).



Gambar 2. 4 *Arsitektur MobileNetV2*

MobileNetV2 dapat digunakan dalam berbagai tugas dalam bidang penglihatan komputer, seperti klasifikasi gambar, deteksi objek, pengenalan wajah, dan berbagai tugas lainnya. Hal ini membuatnya menjadi pilihan yang sangat fleksibel dan berguna dalam banyak aplikasi. Secara keseluruhan, *MobileNetV2* adalah arsitektur CNN yang efisien, dapat disesuaikan, dan sangat berguna untuk tugas pengolahan gambar dan penglihatan komputer pada perangkat dengan sumber daya terbatas (Sandler *et al*., 2018).

6. *Tensorflow*

TensorFlow adalah sebuah *library* perangkat lunak yang dikembangkan oleh tim di Google, dengan tujuan utama untuk mendukung riset dalam pembelajaran mesin dan jaringan syaraf. *TensorFlow* mengintegrasikan prinsip-prinsip aljabar komputasi dengan teknik optimasi kompilasi, yang mempermudah eksekusi komputasi matematika yang kompleks dan beragam(Ihsan *et al*., 2021). *TensorFlow* Memiliki fitur yang sangat kuat:

a. *Tensorflow* merupakan sebuah platform yang kuat untuk pemrograman yang mendukung pengembangan jaringan syaraf dan aplikasi *Machine Learning.*
b. *Tensorflow* didasarkan pada konsep tensor, yang merupakan array multidimensi. Inii memungkinkan peneliti untuk mendefinisikan, mengoptimalkan, dan menghitung secara matematis ekspresi yang melibatkan data dalam format tensor. Hal ini sangat berguna dalam konteks pengolahan data yang kompleks.
c. *Tensorflow* mampu memanfaatkan GPU *(Graphics Processing Unit*) secara efisien, yang mempercepat komputasi untuk

Machine Learning dan jaringan saraf yang berat secara komputasi. Platform ini juga otomatis mengelola dan mengoptimalkan alokasi memori yang sama terhadap data yang diproses.

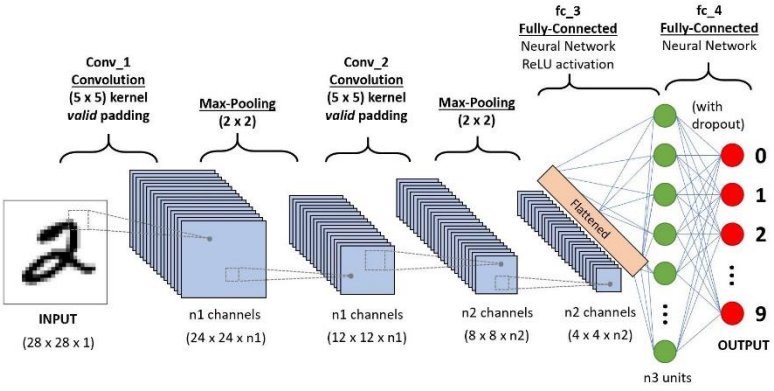
d. *Tensorflow* dirancang untuk menangani komputasi pada skala yang besar. Dengan begini peneliti dapat menggunakan *tensorflow* untuk mengelola dan menghitung data yang sangat besar pada beberapa mesin sekaligus.

7. Convolutional Neural Network (CNN)

Secara umum, *Convolutional Neural Network* (CNN) merupakan salah satu varian dari jaringan saraf yang sering digunakan untuk menganalisis data gambar. Jika kita membandingkannya dengan jaringan saraf biasa, CNN memiliki sejumlah perbedaan yang menonjol. CNN terdiri dari neuron dengan bobot *(weight)*, bias, dan fungsi aktivasi, mirip dengan jaringan saraf lainnya (Guo *et al.*, 2017).

Namun, yang membuat CNN unik adalah lapisan konvolusi, di mana neuron-neuron disusun sedemikian rupa sehingga membentuk filter dengan dimensi tertentu dalam gambar. Konsep utama di balik CNN adalah konvolusi, di mana sebuah kernel

konvolusi *(filter)* bergerak melintasi gambar, menghasilkan informasi representatif baru melalui perkalian titik-titik gambar dengan filter tersebut(Eka Putra, 2016).



***Gambar 2. 5** Arsitektur CNN*

Dalam proses ini, gambar di encode menjadi fitur-fitur yang terdiri dari angka-angka, yang memungkinkan representasi gambar dalam bentuk yang lebih kompak *(feature Extraction)*. Lapisan ekstraksi fitur ini terdiri dari dua komponen utama: Lapisan Konvolusi *(Convolutional Layer)* dan Lapisan Pemampatan *(Pooling Layer)* (Guo *et al.*, 2017).

CNN yang digambarkan dalam gambar ini terdiri dari dua bagian utama, Jaringan Neural Tiruan (JNT) dan *Convolutional Neural Network* (CNN). Gambar input dengan ukuran 28x28 piksel dan 1

channel *(grayscale)* dimasukkan ke dalam CNN. CNN ini terdiri dari dua tahap *convolutional* dan *pooling*, di mana setiap tahap terdiri dari lapisan *convolutional* dan lapisan *pooling*. Lapisan *convolutional* mengekstraksi fitur-fitur penting dari gambar, sedangkan lapisan *pooling* mereduksi dimensi data dan ringkas informasi penting. Hasil dari CNN kemudian diumpankan ke dalam JNT, yang terdiri dari dua lapisan *fully-connected*. JNT ini berfungsi untuk menggabungkan informasi dari fitur-fitur yang diekstrak dan menghasilkan klasifikasi akhir gambar input.

a. Convolution Layer

Lapisan Konvolusi *(Convolutional Layer)* dalam *Convolutional Neural Network* (CNN) adalah salah satu komponen penting dalam struktur jaringan tersebut. Lapisan ini bertanggung jawab untuk melakukan operasi konvolusi pada data gambar. Ini adalah langkah pertama dalam proses ekstraksi fitur dari gambar dan membantu CNN memahami pola dan fitur dalam gambar (Maulana & Rochmawati, 2020).

Dalam lapisan konvolusi, terdapat sejumlah kernel konvolusi *(filter)* yang digunakan untuk mengoperasikan gambar. Setiap kernel adalah

matriks bobot yang digunakan untuk mengidentifikasi pola atau fitur tertentu dalam gambar. Kernel ini dikonvolusi secara bertahap di seluruh gambar, dan operasi perkalian titik demi titik antara kernel dan bagian gambar yang sedang dianalisis dilakukan. Hasil dari operasi konvolusi ini menciptakan matriks baru yang bernama *feature map,* yang menunjukkan dimana dan sejauh mana fitur-fitur yang telah berhasil di ekstrak dari gambar asli (Wicaksono *et al.*, 2020).

b. Stride

Stride adalah salah satu parameter dalam operasi konvolusi yang digunakan dalam *Convolutional Neural Networks* (CNN) dan dalam pemrosesan gambar serta sinyal. Stride mengontrol sejauh berapa langkah kernel konvolusi bergerak saat melakukan konvolusi pada data input. Pada dasarnya, saat kernel konvolusi digeser di atas data input, *stride* menentukan sejauh berapa jumlah langkah yang diambil oleh kernel setiap kali operasi konvolusi diterapkan (Eka Putra, 2016).

Stride ini mempengaruhi ukuran peta fitur yang dihasilkan setelah operasi konvolusi. *Stride* memiliki dua opsi umum yaitu jika nilai *stride*

adalah 1, maka *convolusi filter* bergeser sebanyak 1 *pixel* pada setiap iterasi sedangkan jika nilai stride lebih besar dari 1 maka *convolusi filter* akan bergerak lebih cepat, dan beberapa *pixel* pada data input akan diabaikan dalam proses konvolusi ini. Hal ini mengakibatkan *feature map* yang dihasilkan lebih kecil ukuran nya. *Stride* sering digunakan untuk mengontrol *downsampling* pada lapisan konvolusi, seperti lapisan pooling dalam CNN(Hauzan, 2023).

c. Padding

Padding adalah salah satu konsep yang digunakan dalam pemrosesan gambar dan sinyal, terutama dalam *Convolutional Neural Networks* (CNN). Ini adalah teknik yang digunakan untuk mengelola dimensi data input saat melakukan operasi konvolusi, terutama untuk memastikan bahwa informasi di setiap sisi dari data input tetap diperlakukan secara tepat. Dalam konteks CNN, *padding* mengacu pada penambahan *pixel* di sekitar gambar atau data input sebelum melakukan operasi konvolusi. Ada dua jenis padding yang umum digunakan yaitu *No padding* dan *Zero padding*(Hauzan, 2023).

d. Activation Layer

Activation layer dalam jaringan saraf adalah salah satu komponen yang penting dalam arsitektur neural network, termasuk *Convolutional Neural Networks* (CNN). Lapisan aktivasi bertanggung jawab untuk mengenalkan non-linearitas ke dalam model, sehingga model dapat mempelajari representasi yang lebih kompleks dari data (Guo *et al.*, 2017).

Fungsi utama lapisan aktivasi adalah menerapkan *activation function* pada hasil dari lapisan sebelumnya, yang biasanya adalah lapisan konvolusi atau lapisan tersembunyi *(hidden layer)* pada jaringan saraf. Fungsi aktivasi mengubah nilai-nilai linier menjadi nilai-nilai non-linear, sehingga jaringan dapat memodelkan hubungan yang lebih kompleks dan tidak linier dalam data. Salah satu fungsi aktivasi yang paling sering digunakan adalah ReLU *(rectified Linear Unit)* karena sifatnya yang berfungsi lebih baik. Selain ReLU ada juga beberapa fungsi aktivasi yang digunakan dalam jaringan saraf seperti *Tanh, Sigmoid* dan *Softmax* (Wicaksono *et al*., 2020).

e. Pooling Layer

Pooling Layer adalah salah satu komponen utama dalam *Convolutional Neural Networks* (CNN). Tujuannya adalah untuk mengurangi dimensi dari *feature map* yang dihasilkan oleh lapisan konvolusi sebelumnya. Lapisan ini memainkan peran penting dalam mengurangi jumlah parameter yang dibutuhkan dalam jaringan, mengontrol *overfitting*, dan menjadikan representasi fitur lebih invarian terhadap pergeseran kecil dalam data masukan. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling* (Coskun *et al*., 2017).

f. Fully-Connected Layer

Fully-Connected Layer juga dikenal sebagai lapisan Dense, adalah salah satu komponen utama dalam arsitektur jaringan saraf buatan, termasuk Convolutional Neural Networks (CNN). *Lapisan Fully-connected* adalah lapisan dimana semua neuron aktivitas dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti hal nya jaringan syaraf tiruan bisa. Setiap aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan

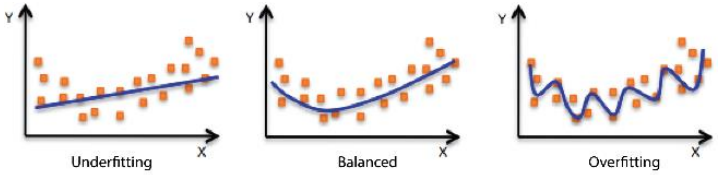
Fully-Connected (Suartika E. P, I Wayan, Wijaya Arya Yudhi, 2016).

Fully-Connected Layer sering ditempatkan di akhir jaringan saraf untuk menghasilkan output yang sesuai dengan tugas yang dikerjakan, seperti mengklasifikasikan gambar ke dalam kelas tertentu. Kombinasi *Convolutional Layer* untuk mengekstraksi fitur dan *Fully-Connected Layer* untuk mengambil keputusan berdasarkan fitur-fitur tersebut sering digunakan dalam CNN untuk tugas pengenalan gambar.

8. **Evaluasi model**

a. Learning Curve

Learning curve adalah grafik atau plot yang menggambarkan bagaimana kinerja model atau algoritma pembelajaran mesin berkembang seiring berjalannya waktu atau seiring dengan meningkatnya jumlah data latihan yang digunakan. *Learning curve* sangat berguna untuk memahami bagaimana model mengatasi tugas pembelajaran dan sejauh mana model tersebut bisa memahami dan menggeneralisasi data dan berguna juga untuk mengidentifikasi data yang mengalami *overfitting* ataupun *underfitting.*

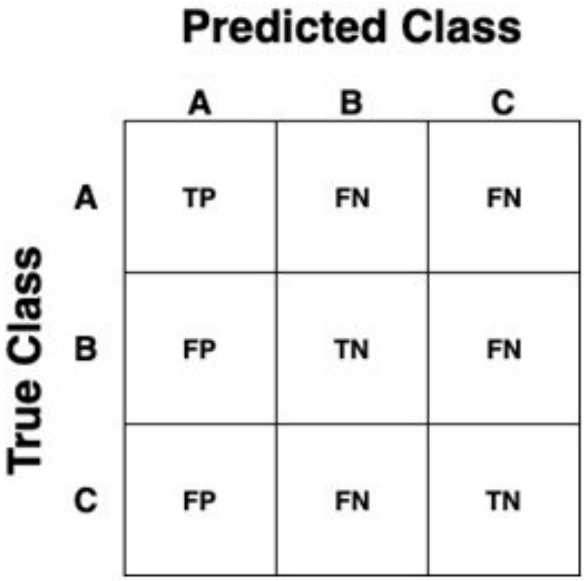


Gambar 2. 6 *Learning curve*

Underfitting merupakan kondisi saat nilai akurasi data training rendah diikuti dengan nilai akurasi data testing yang rendah. Sedangkan *overfitting* adalah kondisi dimana nilai akurasi pada data *training* tinggi dengan nilai akurasi data *testing* rendah(Sudalto, 2022).

b. Confusion Matrix

Confusion Matrix adalah alat evaluasi yang umum digunakan dalam pengujian klasifikasi dan identifikasi untuk mengukur kinerja model. Ini memberikan gambaran tentang sejauh mana model mampu memprediksi dengan benar kelas-kelas yang ada dalam dataset. *Confusion Matrix* biasanya digunakan dalam konteks masalah klasifikasi biner (dua kelas) atau klasifikasi multi-kelas (lebih dari dua kelas). Terdapat 4 istilah representasi hasil proses klasifikasi pada *confusion matrix.* Keempat istilah tersebut adalah *True positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN).



Gambar 2. 7 *Confusion Matrix*

True Positive (TP) adalah jumlah data yang sebenarnya termasuk dalam kelas positif dan telah diprediksi dengan benar sebagai kelas positif oleh model. *True Negative* (TN) adalah jumlah data yang sebenarnya termasuk dalam kelas negatif dan telah diprediksi dengan benar sebagai kelas negatif. *False Positive* (FP) Juga dikenal sebagai *Type I Error*, ini adalah jumlah data yang sebenarnya termasuk dalam kelas negatif, tetapi telah salah diprediksi sebagai kelas positif oleh model. *False Negative* (FN): Juga dikenal sebagai *Type II Error*, ini adalah jumlah data yang

sebenarnya termasuk dalam kelas positif, tetapi telah salah diprediksi sebagai kelas negatif oleh model.(Sudalto,2022) *Confusion matrix* membantu dalam menghitung berbagai metrik evaluasi performa model seperti *Accuracy, Precision, Recall, F1-Score*.

Accuracy adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar dari total prediksi yang dilakukan. Dalam konteks klasifikasi, akurasi memberikan gambaran mengenai seberapa sering model memprediksi kelas yang benar, baik itu kelas positif maupun negatif(Sudalto, 2022). Berikut ini rumus menetukan nilai akurasi berdasarkan persamaan 2.1.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \qquad (2.1)$$

Precision adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar untuk kelas positif dari total prediksi positif yang dilakukan. Dalam konteks klasifikasi, presisi memberikan gambaran mengenai seberapa sering model memprediksi kelas positif dengan benar, di antara semua

prediksi positif yang dibuat oleh model(Sudalto, 2022). Berikut ini rumus menetukan nilai *precision* berdasarkan persamaan 2.2.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

Recall adalah metrik evaluasi yang menggambarkan seberapa baik suatu model dalam mengidentifikasi kelas positif dengan benar(Sudalto, 2022). Berikut ini rumus menetukan nilai *recall* berdasarkan persamaan 2.3.

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

F1-Score merupakan metrik evaluasi yang mencerminkan keseimbangan antara *Precision* dan *Recall*. Nilai *F1-Score* akan memberikan informasi tentang seberapa baik model dalam menggabungkan kemampuan *Precision* dan *Recall*, sehingga bisa memahami seberapa efektif model dalam mengklasifikasikan (Sudalto, 2022). Berikut ini rumus menetukan nilai *F1-Score* berdasarkan persamaan 2.4.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.4)$$

B. Kajian Penelitian yang Relevan

Berikut ini adalah tabel penelitian terkait yang berisi kumpulan jurnal tentang penelitian terdahulu yang berkaitan dengan penelitian pada tugas akhir kali ini pada tabel 2.1.

Tabel 2. 1 Rujukan Penelitian

NO	Judul	Nama & Tahun Penelitian	Metode	Hasil
1	Deteksi Ekspresi Wajah Menggunakan *Deep Learning*	Garry Agustinus safiro (2020)	*Deep Learning* dengan *Local Binary Pattern*	Hasil dari penelitian ini medapatkan akurasi sebesar 85% sedangkan di kondisi redup sebesar 70%.
2	Pengenalan Citra Ekspresi Wajah Menggunakan	Andy Rizki Wiyono dan Elly Matul Imah	*Principal Component Analysis* (PCA) dan	Hasil terbaik dari penelitian ini mendapatkan

	Algoritma *Principal Component Analysis* dan *Extreme Learning Machine*	(2018)	*Extreme Learning Machine* (ELM)	akurasi *testing* sebesar 71,5% dan akurasi *training* sebesar 93,1%
3	Pengenalan Ekspresi Wajah menggunakan LGBP dan SVM	Erwin Yulizar Fardani, Anditya Arifianto, dan Kurniawan Nur Ramadhani (2018)	*Local Gabor Binary Pattern* (LGBP) dan *Support Vector Machine* (SVM)	Hasil terbaik dari Penelitian ini memperoleh akurasi sebesar 69%.
4	Penerapan *Principal Component Analysis* dan *Random Forest* untuk pengenalan	Panji Bintoro, Ratnasari, dan Panggah widiandana (2023)	*Principal Component Analysis (PCA)* dan *Random Forest*	Hasil dari penelitian ini mendapatkan akurasi pengujian sebesar 91,37% dan

	Ekspresi Wajah			akurasi klasifikasi sebesar 91%

Penelitian mengenai pengenalan ekspresi wajah multikategori dengan menggunakan metode *Transfer learning* dan model arsitektur *MobileNetV2* dan *framework Tensorflow*. Penelitian ini memanfaatkan metode *Transfer learning* dimana model yang telah dilatih sebelumnya digunakan sebagai dasar untuk melatih model baru, dengan kata lain penelitian ini menghemat waktu dan sumber daya yang diperlukan untuk melatih model dari awal. Model arsitektur *MobileNetV2* dikenal karena keefisienan dan kecepatannya. Penggunaan *framework tensorflow* memberikan fleksibilitas dan mendukung penggunaan konsep graf komputasi pada berbagai *platform* termasuk perangkat CPU, GPU, dan TPU, hal ini sangat membantu untuk memaksimalkan efisiensi komputasi.

BAB III
METODOLOGI PENELITIAN

A. Jenis Penelitian

Penelitian ini adalah studi eksperimental dengan pendekatan kuantitatif yang bertujuan untuk merancang dan menguji sebuah model *Machine Learning* yang berfokus pada metode *Transfer Learning*. Model ini akan digunakan untuk melakukan deteksi ekspresi wajah multikategori.

B. Jenis dan Sumber data

Dalam tahap awal ini, langkah pertama adalah mengumpulkan data ekspresi wajah yang akan menjadi input dalam penelitian. Data ini diperoleh melalui pencarian sumber data publik atau menggunakan sumber data sekunder yang sudah tersedia di platform Kaggle, yang dapat diakses melalui tautan URL berikut:
https://www.kaggle.com/datasets/msambare/fer2013

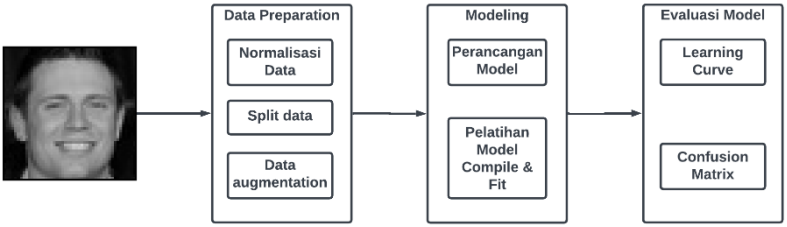
Dalam proses pengumpulan data ini, peneliti menggunakan 15057 gambar yang kemudian dikelompokkan menjadi 3 kelas utama. Gambar-gambar ini memiliki dimensi 48 x 48 *pixel* dan diformat dalam tipe citra *grayscale* dengan ekstensi .jpg.

Informasi lebih rinci terkait jumlah data dapat dilihat pada Tabel 3.1 berikut:

Tabel 3. 1 Jumlah Dataset

Kelas	Jumlah Data
Senang	7096
Sedih	4830
Terkejut	3131
Jumlah Total	**15057**

C. Metode



Gambar 3. 1 *Alur Metode*

Dalam melakukan penelitian ini, metode menjadi pedoman utama yang mengatur alur dari setiap tahap proses penelitian. Setiap tahap penelitian dirancang secara terstruktur, dimulai dari tahap awal hingga tahap penyelesaian. Metode analisis data menjadi landasan yang memberikan arahan untuk semua prosedur yang terlibat dalam penelitian ini. Dalam penelitian ini, metode yang diterapkan dengan menggunakan teknologi *Machine Learning* yang memanfaatkan *Convolutional Neural Network* (CNN) sebagai metode utama dengan model arsitektur *MobileNetV2*, dan menggunakan metode

Transfer Learning serta *framework TensorFlow*. Berikut adalah alur tahapan metode yang akan digunakan:

1. **Data *preparation***

    a. Normalisasi Data

        Sebelum memasuki tahap normalisasi maka diperlukan proses inisialisasi array yang berfungsi untuk menyimpan data dari gambar dan juga label yang sesuai. Kemudian dilanjutkan dengan proses iterasi melalui sebuah direktori yang di dalamnya memuat gambar dari ekspresi wajah yang bertujuan untuk memeriksa kembali bahwa dari setiap gambar akan dimuat dan labelnya ditentukan sesuai berdasarkan jenis dari ekspresi yang direpresentasikan. Apabila gambar telah selesai dimuat lalu akan dilakukan proses normalisasi yang berfungsi untuk memeriksa kembali ukuran piksel pada setiap gambar dan menskalakan data ke rentang yang sama.

    b. *Split* Data

        Pada tahap ini dataset akan dibagi dengan perbandingan 80:20. Dalam pembagian ini 80% dari data akan digunakan sebagai data latih *(Training),* dan sisanya 20% akan digunakan menjadi data validasi.

c. Data *Augmentation*

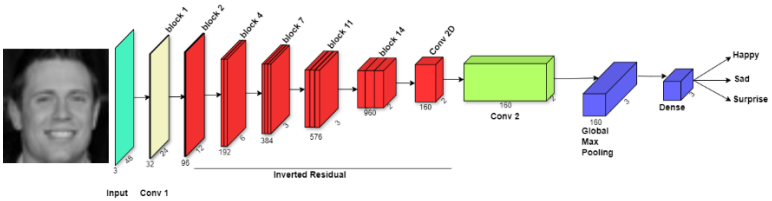
Dalam proses *augmentation* dataset akan dimodifikasi yang bertujuan untuk meningkatkan jumlah dan variasi data. Teknik ini menghasilkan data baru dari data asli sehingga model memiliki banyak contoh untuk mempelajari dan meningkatkan performa. Beberapa teknik *augmentation* di antaranya yaitu:

1) *rotation_range* merupakan proses augmentasi yang akan mengubah rotasi pada gambar secara acak dalam derajat.
2) *zoom_range* merupakan proses augmentasi berupa memperbesar pada gambar.
3) *shear_range* merupakan proses yang akan mengendalikan sejauh mana gambar mengalami pergeseran secara diagonal.
4) *width_shift_range* merupakan proses pergeseran pada gambar secara horizontal (ke kanan ataupun ke kiri).
5) *height_shift_range* merupakan proses pergeseran gambar secara vertikal (ke atas ataupun ke bawah).

2. Modeling

a. Perancangan Model

Dalam tahap perancangan model, peneliti akan menerapkan metode *Transfer Learning* dengan memanfaatkan arsitektur CNN terkemuka, yaitu *MobileNetV2*. Kelebihan dari penggunaan *MobileNetV2* ini adalah kemampuannya untuk mengatasi keterbatasan sumber daya komputasi yang berlebihan. Peneliti juga akan menggunakan framework *TensorFlow* untuk mempermudah proses pemrosesan data.



Gambar 3. 2 *MobileNetV2 Model*

1. Sebelum gambar di *input*, gambar akan diaugmentasi seperti yang sudah dijelaskan pada bagian data *Augmentation.*
2. Hasil *augmentation* gambar kemudian dipanggil untuk dijadikan *input* pada Model *MobileNetV2* ini. Input gambar yang akan digunakan berukuran 48x48x3.

3. Setelah itu akan diproses pada konvolusi pertama. Pada proses konvolusi pertama ini *feature map* yang dihasilkan adalah 24x24x32.
4. Pada *Inverted Residual* terdapat 3 lapisan konvolusi yaitu *Expansion Layer*, *Depthwise Convolution,* dan *Projection Layer. Expansion Layer* merupakan konvolusi yang bertujuan untuk memperluas jumlah saluran data sebelum masuk ke *Depthwise Convolution* sehingga *Expansion Layer* memiliki lebih banyak saluran keluaran daripada saluran masukan, seberapa banyak data yang diperluas diberikan oleh *Factor Expansion*. *Depthwise Convolution* melakukan penyaringan apapun yang penting pada tahap jaringan ini. *Projection Layer* mengembalikan data menjadi semula atau lebih kecil.
5. Resolusi *Output* dari jaringan dasar berupa citra 3x3x160. Ukuran citra ini akan masuk kedalam pengklasifikasi pertama menggunakan *Global Max Pooling.*
6. Pada lapisan *Dense* dengan *Activation Softmax* akan melakukan klasifikasi 3 kelas ekspresi wajah yaitu *Happy, Sad*, dan *Surprise*.

b. Pelatihan Model

Setelah perancangan model dengan menggunakan metode *Transfer Learning*, tahap berikutnya adalah melakukan proses pelatihan. Proses pelatihan ini terdiri dari dua tahap utama yang perlu dilakukan, yaitu tahap kompilasi (compile) dan tahap penyesuaian (fit).

1) Compile

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

Model.compile() adalah metode pada model jaringan saraf tiruan yang digunakan untuk mengkonfigurasi model sebelum pelatihan. Terdapat beberapa parameter atau argumen yang memiliki fungsinya masing-masing. Pertama *optimizer='adam"* memiliki peran penting dalam menentukan algoritma optimisasi yang akan mengatur bagaimana bobot model *neural network* dilakukan berdasarkan perhitungan loss.

Optimizer Adam dikenal karena kemampuannya yang kuat dalam mengoptimalkan model *neural network* dengan efisien. Selanjutnya

loss='categorical_crossentropy' yang mengatur fungsi kerugian (*loss function*) yang digunakan untuk mengukur sejauh mana model benar atau salah dalam melakukan prediksi. Argumen ini juga sering digunakan dalam masalah klasifikasi dan identifikasi multikelas. Dan yang terakhir *metrics=['accuracy']* berfungsi menentukan matrik evaluasi yang digunakan untuk menilai perfoma model serta menjadi indikator sejauh mana model berhasil dalam mengidentifikasi dan memprediksi data dengan benar.

2) *Fit*

```
history = model.fit(
train_datagen.flow(X_train,
y_train, batch_size=batch_size),
validation_data=(X_test, y_test),
steps_per_epoch=len(X_train)/batc
h_size,epochs=epochs,
callbacks=callbacks)
```

Fungsi ini digunakan untuk melatih model jaringan saraf Anda dengan data pelatihan dan data validasi. Ada beberapa argumen yang memiliki peran penting, salah satunya adalah *epoch* yang berperan

menentukan berapa kali model akan melihat seluruh dataset pelatihan selama satu iterasi. Semakin besar nilai *epoch*nya maka semakin tinggi juga tingkat akurasi karena model memiliki lebih banyak kesempatan untuk memahami data. Namun, perlu diingat bahwa peningkatan nilai *epoch* juga berarti waktu pelatihan yang lebih lama. Dengan kata lain, semakin besar nilai *epoch*, semakin lama waktu yang diperlukan untuk menyelesaikan pelatihan model.

Train_data merupakan data latih yang akan digunakan untuk melatih model. Data ini berupa data input gambar dan model akan mempelajari pola dan korelasi dalam data ini selama proses pelatihan. *Validation_data* adalah data validasi yang digunakan untuk mengukur kinerja model selama pelatihan. Data validasi ini digunakan untuk menghindari *overfitting* dan memastikan bahwa model dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat. *Callback* digunakan untuk mengendalikan perilaku pelatihan

seperti menghentikan pelatihan jika tidak ada peningkatan dalam kinerja model.

3. **Evaluasi model**
   a. Evaluasi performa model dilakukan dengan mengamati *Learning Curve*, yang memberikan pengetahuan tentang bagaimana model mengalami perkembangan selama proses pelatihan seiring berjalannya waktu. *Learning Curve* membantu kita untuk memahami apakah model mengalami *overfitting* atau *underfitting*, serta sejauh mana kemampuan model dalam memahami dan memanfaatkan informasi dari data pelatihan.
   b. Dalam proses evaluasi model, peneliti juga akan memanfaatkan metode *Confusion Matrix*, yang memberikan gambaran yang sangat detail tentang performa model. *Confusion Matrix* membantu kita untuk mengidentifikasi sejauh mana model dapat membedakan dengan benar antara berbagai kelas, serta mengukur tingkat akurasi, *precision*, *recall*, dan nilai *F1-score* yang menggambarkan kualitas prediksi model.

BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan menjelaskan lebih lanjut terkait dengan hasil dan pembahasan dari sistem yang sudah dibangun. Penelitian ini akan melalui berbagai tahapan dimulai dari pengambilan data, proses membaca data, *preparation* data, *modeling* dan tahap evaluasi. Penelitian terkait dengan ekspresi wajah multikategori ini menggunakan subjek berupa data ekspresi wajah yang diambil melalui kaggle.

A. Data *Collection*

Tahap pertama yang dilakukan oleh peneliti yaitu mencari dataset yang digunakan dalam penelitian ini. Penelitian ini menggunakan gambar ekspresi wajah yang bertipe citra *grayscale* yang berasal dari website kaggle yaitu berupa data FER (*Facial Expression Recognition*) yang berukuran 48 x 48.

```
# DATASET
Data = "Ekspresi/"

# Inisialisasi variabel
total_images = 0
for dir in os.listdir(Data):
    count = 0
    for f in os.listdir(Data + dir + "/"):
        count += 1
        total_images += 1
    print(f"{dir} has {count} number of images")

print(f"\ntotal images are {total_images}")
```

Gambar 4. 1 *Proses Data Collection*

Langkah pertama load terlebih dahulu dataset yang sudah diambil dari website kaggle dan letakkan di direktori. Kemudian variable '*total_images*' akan diinisialisasi untuk mengecek ada berapa jumlah gambar didalam direktori tersebut dengan cara menggunakan fungsi *loop*. Tujuan dari '*for dir in os.listdir(data)*'adalah untuk melakukan iterasi melalui setiap direktori dari daftar direktori yang ditemukan dalam path 'Data' yang dapat digunakan untuk melakukan operasi tertentu pada setiap direktori tersebut seperti menghitung jumlah file. '*for f in os.listdir(Data + dir + "/")*' digunakan untuk melakukan iterasi didalam sub-direktori yang ada dalam direktori "dir". Fungsi variable 'count' dan 'total_image' adalah setiap kali menemukan sebuah file, maka akan bertambah satu. Setelah tidak ada lagi file yang ditemukan, maka akan ditampilkan ada berapa jumlah file didalam sub direktori dan jumlah file keseluruhan dari direktori tersebut yang ditunjukkan pada gambar 4.2

```
happy has 7096 number of images
sad has 4830 number of images
surprise has 3131 number of images

total images are 15057
```

Gambar 4. 2 *Hasil dari inisialisai variabel*

Tahap selanjutnya mendefinisikan pengambilan kelas yang akan diambil. Tiga kelas yang akan diambil yaitu data yang berupa ekspresi “happy”, “sad”, dan “surprise” ditunjukkan pada gambar 4.3

```
# Daftar ekspresi yang ingin diambil
Ekspresi_Multi = ["happy", "sad", "surprise"]
total_images
```

Gambar 4. 3 *Pendefinisian kelas*

B. Data *Preparation*

Apabila telah mendapat dataset pada proses sebelumnya maka selanjutnya yaitu mempersiapkan dataset yang akan digunakan untuk proses pelatihan model.

1. Inisialisasi *Array* Dan Normalisasi

a. Inisialisasi array untuk gambar dan label

Tujuan nya adalah untuk memuat gambar-gambar dari setiap direktori yang sesuai dengan ekspresi yang diinginkan ke dalam *array numpy ‘img_arr’*, serta untuk menetapkan label *numerik* ke setiap gambar yang dimuat ke dalam *array ‘img_label’*. Dengan memuat gambar kedalam array dan menetapkan label numerik ke setiap gambar, hal ini memungkinkan untuk mempersiapkan data yang diperlukan untuk melatih dan menguji model.

```python
# Inisialisasi array untuk gambar dan label
img_arr = np.empty(shape=(total_images,48,48,3))
img_label = np.empty(shape=(total_images))
label_to_text = {}

i = 0
e = 0
for dir in os.listdir(Data):
    if dir in Ekspresi_Multi:
        label_to_text[e] = dir
        for f in os.listdir(Data + dir + "/"):
            img_arr[i] = cv2.imread(Data + dir + "/" + f)
            img_label[i] = e
            i += 1
        print(f"loaded all {dir} images to numpy arrays")
        e += 1

img_arr.shape, img_label
```

Gambar 4. 4 *Inisialisai Array untuk gambar dan label*

Pada gambar 4.4, ada dua variabel yang diinisialisasi sebelumnya, yaitu `` `i` `` dan `` `e` ``, yang digunakan sebagai indeks dan label, secara berturut-turut. Melalui *loop* `` `for` ``, setiap direktori dalam path yang ditentukan oleh `` `os.listdir(Data)` `` diperiksa. Jika direktori tersebut terdapat dalam daftar `` `Ekspresi_Multi` ``, maka label (ekspresi) yang sesuai diberikan ke variable `` `label_to_text` ``. Selanjutnya, setiap file di dalam direktori tersebut diproses. Gambar dibaca menggunakan *OpenCV* `` (`cv2.imread`) `` dan dimasukkan ke dalam array `` `img_arr` ``, sementara label gambar ditetapkan sesuai dengan variabel `` `e` `` ke dalam array `` `img_label` ``.

Selama proses ini, variabel `i` digunakan untuk memastikan setiap gambar dan label disimpan pada indeks yang tepat dalam *array*. Setelah semua gambar dalam suatu direktori diproses, pesan cetak menunjukkan bahwa semua gambar telah dimuat ke dalam array numpy. Variabel `e` juga ditingkatkan, sehingga label selanjutnya akan sesuai dengan ekspresi wajah yang berbeda. Dibawah ini akan ditampilkan hasil dari proses inisialisasi array ditunjukan pada gambar 4.5

```
loaded all happy images to numpy arrays
loaded all sad images to numpy arrays
loaded all surprise images to numpy arrays

((15057, 48, 48, 3), array([0., 0., 0., ..., 2., 2., 2.]))
```

Gambar 4. 5 *Hasil dari inisialisasi array*

b. Normalisasi Data

```
1  img_arr = img_arr / 255.
```

Gambar 4. 6 *Proses Normalisasi data*

Pada gambar 4.6 merupakan langkah normalisasi dalam pemrosesan gambar. Normalisasi adalah proses yang umumnya dilakukan pada data untuk mengubah rentang nilainya menjadi lebih sesuai untuk diproses oleh

algoritma pembelajaran mesin atau jaringan saraf. Dalam konteks ini, nilai piksel dalam gambar biasanya memiliki rentang antara 0 hingga 255, dengan 0 mewakili warna hitam dan 255 mewakili warna putih. Dengan membagi setiap nilai piksel dalam array '*img_arr*' dengan 255, kita mengubah rentang nilai tersebut menjadi antara 0 dan 1. Hal ini membuat data lebih mudah diinterpretasikan oleh model dan membantu proses pembelajaran dengan membuat gradient lebih stabil dan konvergen lebih cepat(Sudeep & Pal, 2017). Dengan demikian, normalisasi adalah langkah penting dalam pra-pemrosesan gambar sebelum dilakukan pelatihan model.

Gambar 4. 7 *Contoh Hasil Gambar yang Telah di Normalisasi*

2. *Split* Data

```
X_train, X_test, y_train, y_test = train_test_split(img_arr, img_label,
                                                    shuffle=True, stratify=img_label,
                                                    test_size=0.2, random_state=42)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Gambar 4. 8 *Proses Split data*

Pada gambar 4.7, penggunaan fungsi `train_test_split` dari modul `sklearn.model_selection` memungkinkan kita untuk membagi dataset menjadi empat bagian: data latih (*X_train* dan *y_train*) dan data uji (*X_test* dan *y_test*). Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji kinerja model yang telah dilatih.

Dalam pemanggilan fungsi `train_test_split`, parameter `img_arr` digunakan sebagai fitur atau atribut dari dataset, yang dalam konteks ini adalah array gambar yang telah dinormalisasi. Sedangkan parameter `img_label` digunakan sebagai label dari dataset, yang merupakan array label gambar. Selanjutnya, parameter `shuffle=True` memastikan bahwa dataset diacak sebelum pembagian dilakukan, sehingga menghindari bias dalam pembagian. Parameter `stratify=img_label` memastikan bahwa proporsi kelas yang sama dipertahankan di kedua

dataset, sehingga meminimalkan kemungkinan overfitting atau underfitting.

Selain itu, parameter `test_size=0.2` menentukan bahwa 20% dari dataset akan dialokasikan untuk data uji, sementara 80% sisanya akan digunakan sebagai data latih. Terakhir, parameter `random_state=42` menentukan seed untuk pengacakan data, sehingga memastikan hasil pembagian dataset yang konsisten saat kode dieksekusi ulang. Dengan memperoleh dataset yang terbagi dengan baik, kita dapat melatih model menggunakan data latih dan menguji kinerjanya menggunakan data uji. Proporsi yang tepat antara data latih dan data uji serta konsistensi dalam pembagian dataset adalah kunci untuk membangun model pembelajaran mesin yang handal dan dapat diandalkan. Berikut pada gambar 4.8 merupakan hasil dari proses split data

```
((12045, 48, 48, 3), (3012, 48, 48, 3), (12045, 3), (3012, 3))
```

Gambar 4. 9 *Hasil proses split data*

Data terbagi 80% untuk data latih yang berjumlah 12045 data dan 20% untuk data test yang berjumlah 3012 data.

3. Data *Augmentation*

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.15,
    height_shift_range=0.15,
    shear_range=0.15,
    zoom_range=0.15 )

train_datagen.fit(X_train)
```

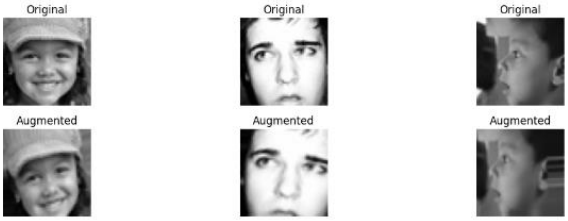
Gambar 4. 10 *Proses Augmentation data*

Pada gambar 4.9 adalah bagian dari proses augmentasi data gambar menggunakan modul `` `ImageDataGenerator` `` dari *Keras.* Augmentasi data merupakan strategi yang berguna dalam melatih model jaringan saraf tiruan, terutama pada dataset gambar, untuk mencegah *overfitting* dan meningkatkan generalisasi model.

Dalam inisialisasi `` `ImageDataGenerator` ``, berbagai jenis transformasi gambar ditentukan dengan parameter seperti `` `rotation_range` ``, `` `width_shift_range` ``, `` `height_shift_range` ``, `` `shear_range` ``, dan `` `zoom_range` ``. Ini mengizinkan variasi kecil pada gambar asli, seperti rotasi, pergeseran, pemotongan, dan perbesaran, yang berguna dalam mengenalkan variasi pada dataset pelatihan.

Setelah konfigurasi transformasi ditentukan, metode `fit` dari `ImageDataGenerator` digunakan dengan data latih *(`X_train`)* sebagai input. Ini memungkinkan `ImageDataGenerator` untuk menghitung statistik yang diperlukan, seperti rata-rata dan deviasi standar, dari data latih. Statistik ini diperlukan untuk mengonfigurasi transformasi yang tepat untuk augmentasi data.

Dengan melakukan augmentasi data ini, kita dapat meningkatkan jumlah sampel dalam dataset pelatihan dengan variasi yang realistis, yang dapat membantu meningkatkan kinerja dan generalisasi model. Proses ini membantu model untuk belajar dengan lebih baik dari data yang tersedia dan meningkatkan kemampuannya dalam mengenali pola-pola yang penting. Sehingga, `ImageDataGenerator` dengan augmentasi data merupakan alat yang sangat berguna dalam pengembangan model jaringan saraf untuk tugas-tugas klasifikasi gambar.



Gambar 4. 11 *Contoh hasil Gambar setelah proses Data Augmentation*

C. *Modeling*

1. Perancangan Model

Sebelum melakukan perancangan model, buatlah variabel seperti pada gambar 4.10, variabel ini akan digunakan untuk paramater saat perancangan model nanti.

```
1 img_width = X_train.shape[1]
2 img_height = X_train.shape[2]
3 img_depth = X_train.shape[3]
4 num_classes = y_train.shape[1]
```

Gambar 4. 12 *Pembuatan variabel parameter*

Tahap selanjutnya yaitu perancangan model. Base model yang digunakan pada penelitian ini menggunakan model *MobileNetV2.* model *MobileNetV2* diinisialisasi dengan menyediakan beberapa parameter penting termasuk yang telah ditentukan sebelumnya yaitu *'img_width'*, *'img_height'*, dan *'img_depth'*, serta jumlah kelas yang akan diprediksi oleh model, yang ditentukan oleh *'num_classes'*. Dengan menetapkan *'include_top'* sebagai *'False'*, kita menghapus lapisan akhir dari model, karena kita akan

menambahkan lapisan kustom di atasnya.

```
base_model = MobileNetV2(
    input_shape = (img_width, img_height, img_depth),
    include_top = False,
    weights = "imagenet",
    classes = num_classes)

model = base_model.layers[-14].output
model = GlobalMaxPool2D(name="global_pool")(model)
model = Dense(num_classes, activation="softmax", name="out_layer")(model)
model = Model(inputs=base_model.input, outputs=model)

model.summary()
```

Gambar 4. 13 *Proses Perancangan model*

Selanjutnya, output dari lapisan ke-14 dari *MobileNetV2* diambil untuk digunakan sebagai dasar untuk pembangunan model. Di atas output ini, ditambahkan lapisan Global Max Pooling, yang bertujuan untuk mereduksi dimensi spasial dari fitur menjadi vektor fitur tunggal. Kemudian, sebuah lapisan Dense ditambahkan untuk melakukan klasifikasi, dengan output sebanyak jumlah kelas yang diinginkan, dan menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas prediksi untuk setiap kelas.

Terakhir, model akhir dibuat dengan menyediakan input dari model *MobileNetV2* dan output dari lapisan *Dense* yang telah ditambahkan sebelumnya. Dengan menggunakan metode '`summary()`', ringkasan arsitektur model ditampilkan, memungkinkan untuk mengevaluasi

struktur dan parameter yang terlibat dalam model. Berikut adalah hasil ringkasan model yang ditunjukkan pada gambar 4.12.

```
block_14_project_BN (Batch  (None, 2, 2, 160)          640       ['block_14_project[0][0]']
Normalization)

block_14_add (Add)          (None, 2, 2, 160)          0         ['block_13_project_BN[0][0]',
                                                                  'block_14_project_BN[0][0]']

block_15_expand (Conv2D)    (None, 2, 2, 960)          153600    ['block_14_add[0][0]']

block_15_expand_BN (BatchN  (None, 2, 2, 960)          3840      ['block_15_expand[0][0]']
ormalization)

block_15_expand_relu (ReLU  (None, 2, 2, 960)          0         ['block_15_expand_BN[0][0]']
)

block_15_depthwise (Depthw  (None, 2, 2, 960)          8640      ['block_15_expand_relu[0][0]']
iseConv2D)

block_15_depthwise_BN (Bat  (None, 2, 2, 960)          3840      ['block_15_depthwise[0][0]']
chNormalization)

block_15_depthwise_relu (R  (None, 2, 2, 960)          0         ['block_15_depthwise_BN[0][0]'
eLU)                                                             ]

block_15_project (Conv2D)   (None, 2, 2, 160)          153600    ['block_15_depthwise_relu[0][0
                                                                 ]']

global_pool (GlobalMaxPool  (None, 160)                0         ['block_15_project[0][0]']
ing2D)

out_layer (Dense)           (None, 3)                  483       ['global_pool[0][0]']

==================================================================================================
Total params: 1364707 (5.21 MB)
Trainable params: 1337955 (5.10 MB)
Non-trainable params: 26752 (104.50 KB)
```

Gambar 4. 14 *Hasil Perancangan Model*

Dapat dilihat pada gambar 4.14, total paramater yang digunakan dalam model ini adalah 1.364.707, dimana sebanyak 1.337.955 dianggap sebagai paramater Trainable params atau paramater yang dapat dilatih, sementara 26.752 dianggap sebagai Non-trainable param.

2. Pelatihan Model

a. Compile

```
batch_size = 16
epochs = 50

optims = optimizers.Adam(learning_rate=0.001)
```

```
model.compile(loss='categorical_crossentropy',
              optimizer=optims,
              metrics=['accuracy'])
```

***Gambar 4. 15** Proses Compile*

Pada tahap kompilasi model, Ada beberapa parameter penting yang memengaruhi proses pelatihan. Pertama, *`categorical_crossentropy`* sebagai fungsi kerugian. Fungsi ini merupakan pilihan umum dalam kasus klasifikasi multikelas, karena mampu menilai perbedaan antara distribusi probabilitas prediksi dan distribusi probabilitas target dengan baik. Selanjutnya, menetapkan pengoptimal yang akan digunakan selama pelatihan, dalam model ini menggunakan algoritma Adam yang telah definisikan sebelumnya dengan *'learning_rate=0.001'.* Pengoptimal bertanggung jawab untuk memperbarui bobot model berdasarkan gradien dari fungsi kerugian selama pelatihan. Terakhir, metrik evaluasi `accuracy` untuk memantau kinerja model selama pelatihan dan evaluasi.

Metrik ini memberikan informasi tentang seberapa sering model membuat prediksi yang benar.

b. Fit

```
history = model.fit(
    train_datagen.flow(X_train, y_train, batch_size=batch_size),
    validation_data=(X_test, y_test),
    steps_per_epoch=len(X_train) / batch_size,
    epochs=epochs,
    callbacks=callbacks
)
```

Gambar 4. 16 *Proses Pelatihan Model (Fit)*

Proses ini merupakan proses pelatihan model. Metode *`fit()`* digunakan untuk melakukan iterasi melalui dataset pelatihan dalam batch yang telah ditentukan. Pertama, *`train_datagen.flow()`* untuk memasok data pelatihan dalam bentuk batch yang akan disesuaikan secara real-time menggunakan augmentasi data yang telah diproses sebelumnya. Kemudian, menyediakan data validasi sebagai argumen *`validation_data`,* yang akan digunakan untuk mengevaluasi kinerja model setelah setiap epoch pelatihan.

Selain itu, menentukan *`steps_per_epoch`* sebagai jumlah langkah yang akan dijalankan oleh model setiap epoch. Langkah ini dihitung berdasarkan jumlah total sampel dalam data pelatihan dibagi dengan ukuran batch,

memastikan bahwa model melihat seluruh dataset pelatihan setiap epoch. Selama proses pelatihan, model Anda akan belajar untuk meminimalkan nilai fungsi kerugian dan meningkatkan akurasi klasifikasi gambar. *Callbacks* opsional juga dapat ditambahkan untuk memantau kinerja model atau menyesuaikan pelatihan. Dengan cara ini, proses pelatihan akan berjalan selama jumlah *epoch* yang telah ditentukan, dan kinerja model akan dievaluasi secara berkala menggunakan data validasi. Berikut dibawah ini merupakan hasil dari proses pelatihan data yang ditunjukkan pada gambar 4.15.

```
Epoch 27: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
752/752 [==============================] - 30s 40ms/step - loss: 0.2231 - accuracy: 0.9169 - val_los
s: 0.2907 - val_accuracy: 0.8981 - lr: 2.5000e-04
Epoch 28/50
752/752 [==============================] - 31s 41ms/step - loss: 0.2008 - accuracy: 0.9279 - val_los
s: 0.2859 - val_accuracy: 0.9024 - lr: 6.2500e-05
Epoch 29/50
752/752 [==============================] - 30s 40ms/step - loss: 0.1934 - accuracy: 0.9289 - val_los
s: 0.2961 - val_accuracy: 0.8994 - lr: 6.2500e-05
Epoch 30/50
752/752 [==============================] - 30s 40ms/step - loss: 0.1901 - accuracy: 0.9281 - val_los
s: 0.2953 - val_accuracy: 0.9004 - lr: 6.2500e-05
 Epoch 37/50
 752/752 [==============================] - 29s 39ms/step - loss: 0.1755 - accuracy: 0.9378 - val_los
 s: 0.2951 - val_accuracy: 0.9011 - lr: 3.9063e-06
 Epoch 38/50
 752/752 [==============================] - 29s 39ms/step - loss: 0.1801 - accuracy: 0.9359 - val_los
 s: 0.2952 - val_accuracy: 0.9001 - lr: 3.9063e-06
 Epoch 39/50
 753/752 [==============================] - ETA: 0s - loss: 0.1803 - accuracy: 0.9347Restoring model w
 eights from the end of the best epoch: 28.
 752/752 [==============================] - 30s 40ms/step - loss: 0.1803 - accuracy: 0.9347 - val_los
 s: 0.2953 - val_accuracy: 0.9007 - lr: 3.9063e-06
 Epoch 39: early stopping
```

Gambar 4. 17 *Hasil Proses Pelatihan Model*

Pada gambar 4.15, pada *epoch* 39 terdapat *early stopping* karena model sudah tidak ada kenaikan dan penurunan akurasi yang signifikan, waktu untuk menyelesaikan satu *epoch* sekitar 27-30 detik. Ditunjukan bahwa nilai akurasi setiap *epoch* stabil di 89-90% dan nilai tertinggi nya pada *epoch* ke-28 dengan nilai *val_accuracy* sebesar 0.9024 dan *val_loss* sebesar 0.2895.

3. Pengujian Pada Nilai True

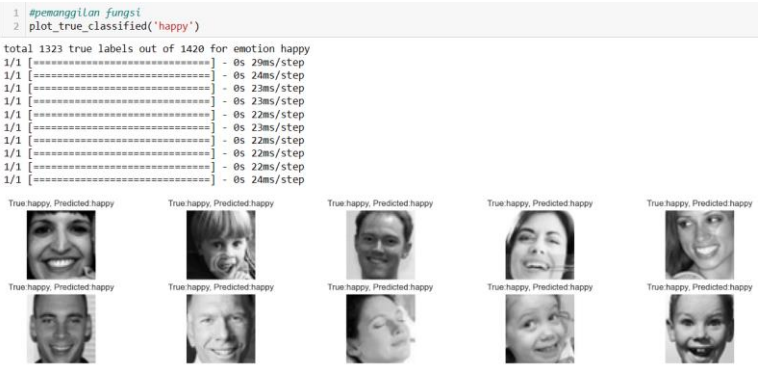
```python
def plot_true_classified(ekspresi):
    true_indices = np.where((ytest_ == predict_test) & (ytest_==label[ekspresi]))[0]
    print(f"total {len(true_indices)} true labels out of {len(np.where(ytest_==label[ekspresi])[0])} for ekspresi {ekspresi}")

    # Tampilkan hanya 10 gambar
    true_indices = true_indices[:10]
    cols = 5
    rows = 2
    fig = plt.figure(1, (20, rows * 2))

    for i, idx in enumerate(true_indices):
        sample_img = X_test[idx,:,:,:]
        sample_img = sample_img.reshape(1,*sample_img.shape)
        true_label = label_to_text[ytest_[idx]]  # Mendapatkan Kelas yang benar dari ytest_
        pred = label_to_text[np.argmax(model.predict(sample_img), axis=1)[0]]

        ax = plt.subplot(rows, cols, i+1)
        ax.imshow(sample_img[0,:,:,0], cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(f"True:{true_label}, Predicted:{pred}")
```

Gambar 4. 18 *Proses klasifikasi data yang benar*

Pada gambar 4.16, fungsi `*plot_true_classified (ekspresi)*` bertujuan untuk membantu dalam mengevaluasi kinerja model klasifikasi. Pertama, fungsi ini mencari indeks dari sampel-sampel yang berhasil diklasifikasikan dengan benar oleh model. Proses ini dilakukan dengan menggunakan kondisi yang membandingkan label yang diprediksi oleh model dengan label yang sebenarnya, serta memastikan bahwa label

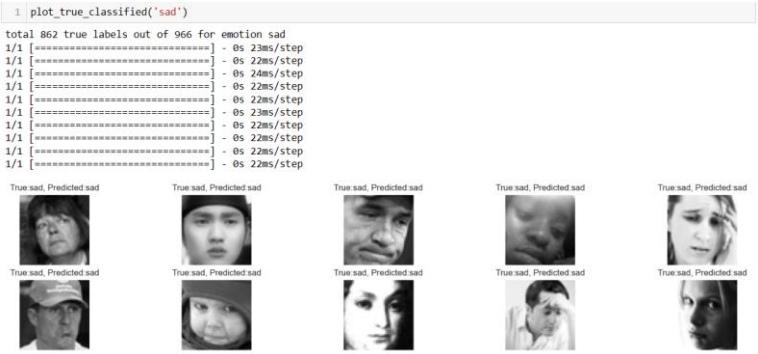
tersebut sesuai dengan kelas yang ditentukan oleh argumen `*ekspresi*`. Setelah itu, informasi tentang jumlah total sampel yang berhasil diklasifikasikan dengan benar dari total sampel yang memiliki label yang sesuai dengan `*ekspresi*` dicetak untuk memberikan pemahaman yang lebih baik tentang kinerja model. Untuk menghindari visualisasi yang berlebihan, jumlah sampel yang ditampilkan dibatasi hingga maksimal 10 gambar. Dengan tata letak gambar yang telah disiapkan, setiap gambar menampilkan sampel gambar asli dari dataset bersama dengan label yang benar dan label yang diprediksi oleh model.

Melalui visualisasi ini, peneliti dapat secara langsung melihat bagaimana model mengklasifikasikan sampel-sampel yang berhasil diklasifikasikan dengan benar untuk kelas yang ditentukan. Dengan demikian, fungsi ini berguna untuk mengevaluasi dan memahami kinerja model dalam konteks klasifikasi yang diberikan. Berikut dibawah ini contoh hasil sampel yang berhasil diklasifikasikan dengan benar dapat dilihat pada gambar 4.17, gambar 4.18 dan gambar 4.19.



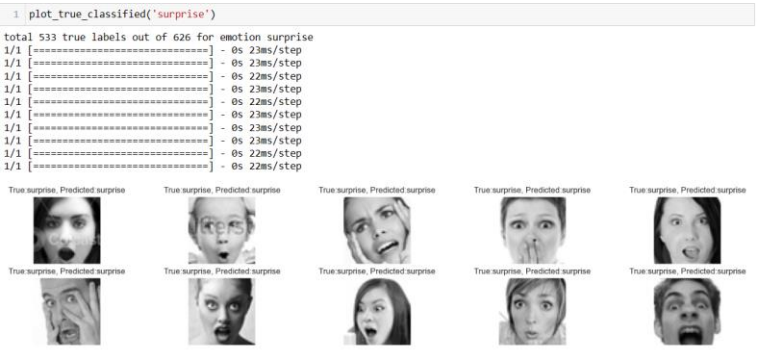
Gambar 4. 19 *Contoh Gambar yang diprediksi benar pada kelas Happy*

Pada gambar 4.17, dapat dilihat bahwa model telah berhasil memprediksi dengan benar sesuai dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Happy* dan model memprediksinya sebagai kelas *Happy*.



Gambar 4. 20 *Contoh Gambar yang diprediksi benar pada kelas Sad*

Pada gambar 4.18, dapat dilihat bahwa model telah berhasil memprediksi dengan benar sesuai dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Sad* dan model memprediksinya sebagai kelas *Sad*.



Gambar 4. 21 *Contoh Gambar yang diprediksi benar pada kelas Surprise*

Pada gambar 4.19, dapat dilihat bahwa model telah berhasil memprediksi dengan benar sesuai dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Surprise* dan model memprediksinya sebagai kelas *Surprise*.

4. Pengujian Pada Nilai False

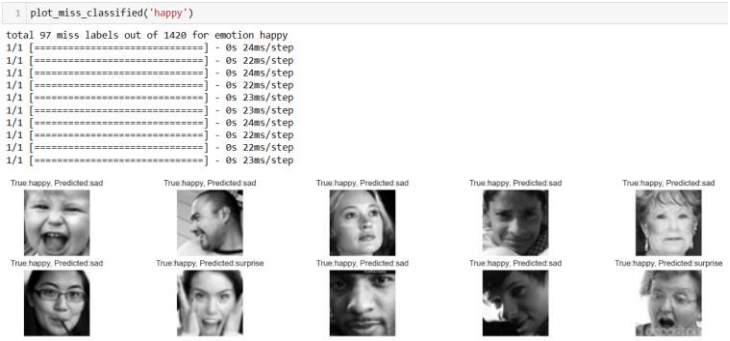
```python
def plot_miss_classified(ekspresi):
    miss_indices = np.where((ytest_ != predict_test) & (ytest_==label[ekspresi]))[0]
    print(f"total {len(miss_indices)} miss labels out of {len(np.where(ytest_==label[ekspresi])[0])} for emotion {ekspresi}")

    # Tampilkan hanya 10 gambar
    miss_indices = miss_indices[:10]
    cols = 5
    rows = 2
    fig = plt.figure(1, (20, rows * 2))

    for i,idx in enumerate(miss_indices):
        sample_img = X_test[idx,:,:,:]
        sample_img = sample_img.reshape(1,*sample_img.shape)
        true_label = label_to_text[ytest_[idx]]  # Mendapatkan label yang benar dari ytest_
        pred = label_to_text[np.argmax(model.predict(sample_img), axis=1)[0]]

        ax = plt.subplot(rows,cols,i+1)
        ax.imshow(sample_img[0,:,:,0], cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(f"p:{pred}")
        ax.set_title(f"True:{true_label}, Predicted:{pred}")
```

Gambar 4. 22 *Proses klasifikasi data yang salah*

Pada gambar 4.20, fungsi `*plot_miss_classified (ekspresi)*` bertujuan untuk membantu dalam mengevaluasi kinerja model klasifikasi dengan memvisualisasikan sampel-sampel yang salah diklasifikasikan. Pertama, fungsi ini mencari indeks dari sampel-sampel yang salah diklasifikasikan oleh model. Hal ini dilakukan dengan mencari sampel-sampel di mana label yang diprediksi oleh model tidak sama dengan label yang sebenarnya, tetapi sesuai dengan kelas yang ditentukan oleh argumen `*ekspresi*`. Setelah itu, informasi tentang jumlah total sampel yang salah diklasifikasikan dari total sampel yang memiliki label yang sesuai dengan `*ekspresi*` dicetak untuk memberikan pemahaman yang lebih baik tentang kinerja model. Untuk

menghindari visualisasi yang berlebihan, jumlah sampel yang ditampilkan dibatasi hingga maksimal 10 gambar. Dengan tata letak gambar yang telah disiapkan, setiap gambar menampilkan sampel gambar asli dari dataset bersama dengan label yang benar dan label yang diprediksi oleh model.

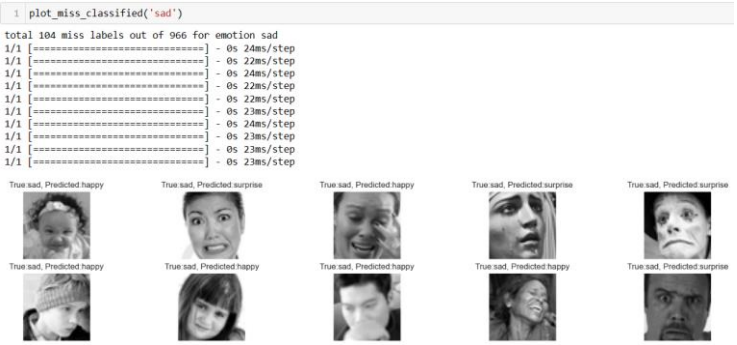
Melalui visualisasi ini, peneliti dapat melihat secara langsung sampel-sampel yang salah diklasifikasikan oleh model untuk kelas yang ditentukan, membantu dalam pemahaman tentang kelemahan atau area yang perlu diperbaiki dari model klasifikasi tersebut.



Gambar 4. 23 *Contoh Gambar yang diprediksi salah pada kelas Happy*

Pada gambar 4.21, dapat dilihat bahwa model belum berhasil memprediksi dengan benar sesuai

dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Happy* namun model memprediksinya sebagai kelas *Sad atau Surprise*.



Gambar 4. 24 *Contoh Gambar yang diprediksi salah pada kelas Sad*

Pada gambar 4.22, dapat dilihat bahwa model belum berhasil memprediksi dengan benar sesuai dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Sad* namun model memprediksinya sebagai kelas *Happy atau Surprise*.

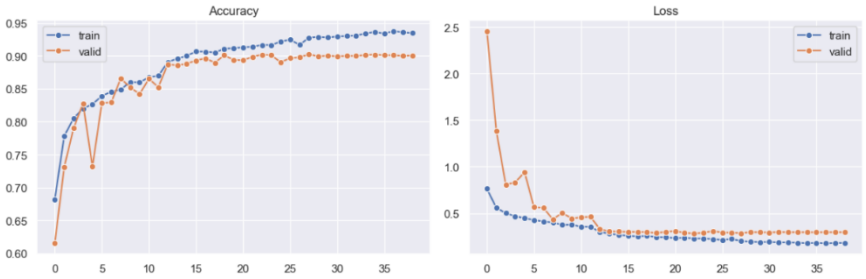
Gambar 4. 25 *Contoh Gambar yang diprediksi salah pada kelas Surprise*

Pada gambar 4.23 dapat dilihat bahwa model belum berhasil memprediksi dengan benar sesuai dengan kelas nya, dimana data asli dari gambar tersebut termasuk dalam kategori kelas *Surprise* namun model memprediksinya sebagai kelas *Happy atau Sad*.

D. Evaluation

1. Learning Curve

```
# Learning Curve
sns.set()
fig = plt.figure(figsize=(12, 4))

#Learning curve Accuracy
ax = plt.subplot(1, 2, 1)
sns.lineplot(x=history.epoch, y=history.history['accuracy'], label='train', marker='o',)
sns.lineplot(x=history.epoch, y=history.history['val_accuracy'], label='valid', marker='o')
plt.title('Accuracy')
plt.tight_layout()

#Learning curve Loss
ax = plt.subplot(1, 2, 2)
sns.lineplot(x=history.epoch, y=history.history['loss'], label='train', marker='o')
sns.lineplot(x=history.epoch, y=history.history['val_loss'], label='valid', marker='o')
plt.title('Loss')
plt.tight_layout()

plt.show()
```



Gambar 4. 26 *Proses Visualisasi Learning Curve*

Berdasarkan pada gambar 4.16, Dalam proses pelatihan model, teramati bahwa akurasi pada data pelatihan dan validasi terus meningkat seiring berjalannya *epoch*, sementara nilai *loss* pada kedua dataset tersebut terus menurun. Peningkatan akurasi dan penurunan *loss* pada kedua dataset tersebut kemudian mencapai konvergensi pada *epoch* ke-13. Fenomena ini mengindikasikan performa yang sangat baik dari model, dengan nilai yang stabil. Akurasi terbaik mencapai 0.90 atau 90%, sedangkan *loss* hanya mencapai 0.28 atau 28%, menunjukkan kinerja model yang sangat memuaskan. Dengan ini jelas menunjukkan bahwa model ini tidak mengalami *Overfitting* ataupun *Underfitting*.

2. Akurasi, Precision, Recall dan F1-Score

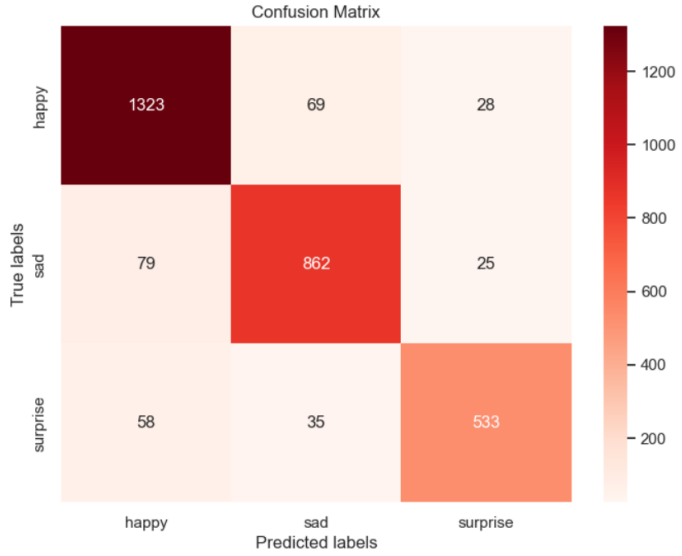
Setelah tahap evaluasi model menggunakan *Learning Curve*, tahap selanjutnya adalah evaluasi

menggunakan *Confusion Matrix* untuk menggambarkan performa model dengan membandingkan prediksi yang dilakukan model terhadap nilai sebenernya dari data uji. *Confusion matrix* memungkinkan untuk menghitung matrik evaluasi seperti Akurasi, *Precision*, *Recall* dan *F1-score.*

```
#evaluasi model test
predict_test = np.argmax(model.predict(X_test), axis=1)
ytest_ = np.argmax(y_test, axis=1)


# Menghitung confusion matrix
conf_matrix = confusion_matrix(ytest_, predict_test)
test_accu = np.sum(ytest_ == predict_test) / len(ytest_) * 100

#menampilkan hasil akurasi dan classification report
print(classification_report(ytest_, predict_test))
print(f"Test Accuracy: {round(test_accu, 4)} %\n\n")


# Membuat plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Reds', cbar=True)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

Gambar 4. 27 *Proses Visualisasi Confusion Matrix*

Pertama, dilakukan prediksi terhadap set data uji menggunakan model yang telah dilatih sebelumnya. Prediksi dilakukan dengan mengambil kelas dengan probabilitas tertinggi dari output model. Selanjutnya, *Confusion Matrix* dihitung untuk menunjukkan seberapa baik model dalam memprediksi kelas yang benar dan yang salah.



Gambar 4. 28 *Hasil Visualisasi Confusion Matrix*

Pada gambar 4.18, terdapat 3 kelas yang diprediksi, pada kelas *Happy* memprediksi sebanyak 1420 data yang dimana 1323 diprediksi benar sebagai kelas *Happy*, 69 data diprediksi sebagai *Sad* dan 28 data diprediksi sebagai *Surprise*.

Pada kelas *Sad* memprediksi 966 data yang dimana 862 data diprediksi benar sebagai kelas *Sad*, 79 data diprediksi sebagai kelas *Happy* dan 25 data diprediksi sebagai kelas *Surprise*.

Pada kelas *Surprise* memprediksi 626 data yang dimana 533 data diprediksi benar sebagai kelas

Surprise, 58 data diprediksi sebagai kelas *Happy* dan 35 data diprediksi sebagai kelas *Sad.*

```
95/95 [==============================] - 2s 14ms/step
              precision    recall  f1-score   support

           0       0.91      0.93      0.92      1420
           1       0.89      0.89      0.89       966
           2       0.91      0.85      0.88       626

    accuracy                           0.90      3012
   macro avg       0.90      0.89      0.90      3012
weighted avg       0.90      0.90      0.90      3012

Test Accuracy: 90.239 %
```

Gambar 4. 29 *Hasil Classification Report*

Berdasarkan gambar 4.19, *Precision* model untuk setiap kelas nya menunjukan hasil yang cukup baik, untuk *precision* kelas *Happy* sebesar 0.91 atau 91%, untuk *precision* kelas *Sad* sebesar 0.89 atau 89%, dan untuk *precision* kelas *Surprise* sebesar 0.91 atau 91%.

Pada *recall*, model ini menunjukan hasil yang baik pada setiap kelas nya. Untuk *recall* pada kelas *Happy* sebesar 0.93 atau 93%, untuk *recall* pada kelas *Sad* sebesar 0.89 atau 89%, dan untuk *recall* pada kelas *Surprise* sebesar 0.85 atau 85%.

Untuk *F1-Score* juga menunjukan hasil yang cukup baik dimana nilai *F1-score* pada kelas *Happy* sebesar 0.92 atau 92%, untuk kelas *Sad* sebesar 0.89

atau 89% dan untuk kelas *Surprise* sebesar 0.88 atau 88%. Secara keseluruhan menunjukan bahwa model memiliki perfoma yang cukup baik dengan akurasi yang mencapai 0.90 atau 90%.

Setelah langkah-langkah tersebut, peneliti melanjutkan dengan melakukan perhitungan performa secara manual dengan mempertimbangkan metrik-metrik seperti akurasi, presisi, recall, dan skor F1 berdasarkan matriks kebingungan. Penentuan nilai TP (True Positive), TN (True Negative), FP (False Positive), dan FN (False Negative) telah dijelaskan sebelumnya dalam bab sebelumnya, yaitu pada Gambar 2.7.

Tabel 4. 1 *Tabel Confusion Matrix*

		Predicted Class		
		Happy	***Sad***	***Surprise***
True class	***Happy***	1323	69	28
	Sad	79	862	25
	Surprise	58	35	533

Berdasarkan Tabel 4.1, telah didapatkan nilai *Confusion Matrix* untuk setiap kelas, langkah selanjutnya adalah melakukan perhitungan performa

model secara manual. Proses perhitungan akurasi akan dijelaskan pada tahap di bawah ini.

$$Accuracy = \frac{TP_{Happy} + TN_{Sad} + TN_{Surprise}}{Jumlah\ Keseluruhan\ Data} \times 100$$

$$Accuracy = \frac{1323 + 862 + 533}{3012} \times 100$$

$$Accuracy = \frac{2718}{3012} \times 100$$

$$Accuracy = 0.902 \times 100$$

$$Accuracy = 90\ \%$$

Setelah mengetahui nilai akurasi, langkah selanjutnya adalah melakukan perhitungan untuk metrik kinerja lainnya seperti presisi, recall, dan skor F1 berdasarkan nilai *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN). Mengingat bahwa menentukan nilai TP, FP, dan FN pada *Confusion Matrix* 3x3 bisa menjadi rumit, solusi yang diusulkan adalah dengan membagi matriks tersebut menjadi matriks 2x2 terlebih dahulu, seperti yang ditunjukkan dalam tabel di bawah ini

a. Perhitungan pada Kelas Happy

Tabel 4. 2 *Perhitungan Kelas Happy*

True/Predic	Happy	Bukan Happy
Happy	TP = 1323	FN = 69+28 = 97
Bukan Happy	FP=79+58 = 137	TN=862+25+35+533 = 1455

Berdasarkan Tabel 4.2, telah didapatkan nilai TP=1323, nilai FP=137, nilai FN=97, dan nilsi TN=1455. Setelah itu, mulai menghitung nilai precission, recall, dan f1-score.

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini:

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{\boldsymbol{TP+FP}}$$

$$Precision\ Happy = \frac{1323}{1323 + 137}$$

$$Precision\ Happy = \frac{1323}{1460}$$

$$Precision\ Happy = 0.906 = 91\%$$

Untuk perhitungan Recall menggunakan persamaan di bawah ini:

$$\boldsymbol{Recall} = \frac{\mathbf{TP}}{\boldsymbol{TP+FN}}$$

$$Recall\ Happy = \frac{1323}{1323 + 97}$$

$$Recall\ Happy = \frac{1323}{1420}$$

$$Recall\ Happy = 0.931 = 93\%$$

Untuk perhitungan F1-score menggunakan persamaan di bawah ini:

$$\boldsymbol{F1 - Score = 2x \frac{Precision\ x\ Recal}{Precision + Recall}}$$

$$F1 - Happy = 2x \frac{0.906\ x\ 0.931}{0.906 + 0.931}$$

$$F1 - Happy = 2x \frac{0.843}{1.837}$$

$$F1 - Happy = 2x\ 0.4589$$

$$F1 - Happy = 0.917 = 92\%$$

b. Perhitungan pada Kelas Sad

Tabel 4. 3 *Perhitungan Kelas Sad*

True/Predic	Sad	Bukan Sad
Sad	TP = 862	FN = 79+25 = 104
Bukan Sad	FP=69+35 = 104	TN=1323+28+58+533 = 1942

Berdasarkan Tabel 4.3, telah didapatkan nilai TP=862, nilai FP=104, nilai FN=104, dan nilai TN=1942. Setelah

itu, mulai menghitung nilai precission, recall, dan f1-score.

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini:

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{\boldsymbol{TP+FP}}$$

$$Precision\ Sad = \frac{862}{862 + 104}$$

$$Precision\ Sad = \frac{862}{966}$$

$$Precision\ Sad = 0.892 = 89\%$$

Untuk perhitungan Recall menggunakan persamaan di bawah ini:

$$\boldsymbol{Recall} = \frac{\mathbf{TP}}{\boldsymbol{TP+FN}}$$

$$Recall\ Sad = \frac{862}{862 + 104}$$

$$Recall\ Sad = \frac{862}{966}$$

$$Recall\ Sad = 0.892 = 89\%$$

Untuk perhitungan F1-score menggunakan persamaan di bawah ini:

$$\boldsymbol{F1 - Score = 2x \frac{Precision\ x\ Recall}{Precision + Recall}}$$

$$F1 - Sad = 2x \frac{0.892\ x\ 0.892}{0.892 + 0.892}$$

$$F1 - Sad = 2x \frac{0.795}{1.784}$$

$$F1 - Sad = 2x\ 0.4456$$

$$F1 - Sad = 0.891 = 89\%$$

c. Perhitungan pada Kelas Surprise

Tabel 4. 4 *Perhitungan Kelas Surprise*

True/Predic	Surprise	Bukan Surprise
Surprise	TP = 533	FN = 58+35 = 93
Bukan Surprise	FP=28+25 = 53	TN=1323+79+69+862 = 2333

Berdasarkan Tabel 4.4, telah didapatkan nilai TP=533, nilai FP=53, nilai FN=93, dan nilai TN=2333. Setelah itu, mulai menghitung nilai precission, recall, dan f1-score.

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini:

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{\boldsymbol{TP+FP}}$$

$$Precision\ Surprise = \frac{533}{533 + 53}$$

$$Precision\ Surprise = \frac{533}{586}$$

$$Precision\ Surprise = 0.909 = 91\%$$

Untuk perhitungan Recall menggunakan persamaan di bawah ini:

$$\boldsymbol{Recall} = \frac{\mathbf{TP}}{\boldsymbol{TP+FN}}$$

$$Recall\ Surprise = \frac{533}{533 + 93}$$

$$Recall\ Surprise = \frac{533}{626}$$

$$Recall\ Surprise = 0.851 = 85\%$$

Untuk perhitungan F1-score menggunakan persamaan di bawah ini:

$$\boldsymbol{F1 - Score} = \mathbf{2x}\frac{\mathbf{Precision\ x\ Recall}}{\boldsymbol{Precision + Recall}}$$

$$F1 - Surprise = 2\text{x}\frac{0.909\ \text{x}\ 0.851}{0.909 + 0.851}$$

$$F1 - Surprise = 2\text{x}\frac{0.7735}{1.7600}$$

$$F1 - Surprise = 2\text{x}\ 0.4392$$

$$F1 - Surprise = 0.878 = 88\%$$

Hasil ringkasan dari perhitungan diatas dapat dilihat pada tabel 4.5 di bawah ini

Tabel 4. 5 *Hasil Perhitungan Performa Setiap Kelas*

Kelas	***Precision***	***Recall***	***F1-Score***
Happy	0.91	0.93	0.92
Sad	0.89	0.89	0.89
Surprise	0.91	0.85	0.88

Berdasarkan Tabel 4.5, untuk *precision* kelas *Happy* sebesar 91%, *precision* kelas *Sad* sebesar 89%, dan untuk *precision* kelas *Surprise* sebesar 91%. Untuk *recall* pada kelas *Happy* 93%, *recall* pada kelas *Sad* sebesar 89%, dan untuk *recall* pada kelas *Surprise* sebesar 85%. Nilai *F1-score* pada kelas *Happy* sebesar 92%, kelas *Sad* sebesar 89% dan untuk kelas *Surprise* sebesar 88%.

Tabel 4. 6 *Perhitungan performa model*

Kelas	**TP**	**FP**	**FN**	***Precision***	***Recall***	***F1-Score***
Happy	1323	137	97	0.91	0.93	0.92
Sad	862	104	104	0.89	0.89	0.89
Surprise	533	53	93	0.91	0.85	0.88

Berdasarkan Tabel 4.6, lakukan perhitungan manual untuk *Precision*, *Recall* dan F1-Score nya. Hasil perhitungan yang dilakukan secara manual akan dijelaskan pada proses dibawah ini.

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini:

$$\boldsymbol{Precision} = \frac{\mathbf{TP}}{\boldsymbol{TP+FP}}$$

$$Precision = \frac{1323 + 862 + 533}{1323 + 862 + 533 + 137 + 104 + 53}$$

$$Precision = \frac{2718}{3012}$$

$$Precision = 0.902 = 90\%$$

Untuk perhitungan *Precision* menggunakan persamaan di bawah ini:

$$\boldsymbol{Recal} = \frac{\mathbf{TP}}{\boldsymbol{TP+FN}}$$

$$Recall = \frac{1323 + 862 + 533}{1323 + 862 + 533 + 97 + 104 + 93}$$

$$Recall = \frac{2718}{3012}$$

$$Recall = 0.902 = 90\%$$

Untuk perhitungan F1-score menggunakan persamaan di bawah ini:

$$\boldsymbol{F1 - Score} = \mathbf{2x}\frac{\mathbf{Precision\ x\ Recall}}{\boldsymbol{Precision + Recall}}$$

$$F1 - Score = 2\text{x}\frac{0.902\ \text{x}\ 0.902}{0.902 + 0.902}$$

$$F1 - Score = 2\text{x}\frac{0.813}{1.804}$$

$$F1 - Score = 2\text{x}\ 0.4506$$

$$F1 - Score = 0.901 = 90\%$$

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil pembahasan dari penelitian yang dilakukan oleh peneliti dapat disimpulkan sebagai berikut:

1. Dengan menerapkan metode *Transfer Learning* dan memanfaatkan model arsitektur *MobileNetV2*, berhasil dibangun sebuah model *Machine Learning* yang mampu mengidentifikasi Ekspresi Wajah dalam berbagai kategori.
2. Berdasarkan hasil analisis *Learning Curve,* pada epoch ke-5 model sudah cukup baik untuk mempelajari pola data, kemudian model mencapai titik konvergensi sekitar epoch ke-13, menandakan bahwa proses pelatihan telah stabil dan model telah berhasil mengenali pola yang signifikan dari data latih. dengan ini mengonfirmasi bahwa efektivitas dan kecepatan arsitektur *MobileNetV2* mampu dalam menangani tugas pengenalan ekspresi wajah.
3. Berdasarkan hasil *Confusion Matrix* Penggunaan metode *Transfer Learning* dalam melakukan Pengenalan Ekspresi Wajah Multikategori dengan model arsitektur *MobileNetV2* menghasilkan

performa yang cukup baik. Nilai *F1-score* sebesar 90%, Nilai *Recall* sebesar 89%, Nilai *Precision* sebesar 90%, dan untuk nilai akurasi yang dihasilkan sebesar 90%.

B. Saran

Berikut adalah beberapa saran yang dapat dipertimbangkan untuk penelitian mendatang:

1. Penelitian selanjutnya dapat mencoba model arsitektur yang berbeda selain MobileNetV2, seperti VGGNet, Inception, ResNet50, dan sebagainya. Membandingkan kinerja berbagai model ini dapat memberikan pemahaman lebih mendalam tentang model mana yang paling efektif dalam mengenali ekspresi wajah.
2. Peneliti menyarankan untuk mencoba menggunakan teknik evaluasi dan validasi model yang berupa *K-fold Cross Validation*. Membandingkan teknik pelatihan model dapat memberikan pemahaman lebih mendalam tentang model mana yang paling efektif dalam mengenali ekspresi wajah.

DAFTAR PUSTAKA

Ab Wahab, M. N., Nazir, A., Ren, A. T. Z., Noor, M. H. M., Akbar, M. F., & Mohamed, A. S. A. (2021). Efficientnet-Lite and Hybrid CNN-KNN Implementation for Facial Expression Recognition on Raspberry Pi. *IEEE Access*, *9*, 134065–134080. https://doi.org/10.1109/ACCESS.2021.3113337

Abas, M. A. H., Ismail, N., Yassin, A. I. M., & Taib, M. N. (2018). VGG16 for plant image classification with transfer learning and data augmentation. *International Journal of Engineering and Technology(UAE)*, *7*(4), 90–94. https://doi.org/10.14419/ijet.v7i4.11.20781

Abdi, N. M., & Aisyah, S. (2011). Peningkatan Kualitas Citra Digital Menggunakan Metode Super Resolusi Pada Domain Spasial. *Jurnal Rekayasa Elektrika*, *9*(3), 137–142.

Abidin, Z. (2011). Pengembangan Sistem Pengenalan Ekspresi Wajah menggunakan Jaringan Syaraf Tiruan Backpropagation (Studi Kasus pada Database MUG). *Jurnal Matematika Murni Dan Terapan*, *5*(1), 21–30.

AfrizaL, Ismail, S. J. I., & Satrya, G. B. (2022). Perancangan Sistem Keamanan Rumah Menggunakan Deteksi Wajah Berbasis Machine Learning Menggunakan Tensorflow. *E-Proceeding of Applied Science*, *8*(1), 9–21.

Amaanullah, R. R., Pascifa, G. R., Nugraha, S. A., Zein, M. R., & Adhinata, F. D. (2022). *Implementasi CNN untuk deteksi emosi melalui wajah.pdf* (pp. 236–244).

Arhandi, P. P., Rosiani, U. D., Prasetyawati, A., & Choirina, P. (2018). *Sistem Pengenalan Wajah Untuk Keamanan Folder*. *4*, 268–273.

Chandra, H. (2019). *Jurnal Ilmu Komputer dan Sistem Informasi PENGENALAN WAJAH PEGAWAI KANTOR DENGAN MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK BERBASIS ANDROID*. 1–6.

Coskun, M., Ucar, A., & Demir, Y. (2017). Face Recognition Based on Convolutional Neural Network. *International Conference on Modern Electrical and Energy (MEES)*, *54*(5), 1159–1165. https://doi.org/10.1109/MEES.2017.8248937

Eka Putra, W. S. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, *5*(1). https://doi.org/10.12962/j23373539.v5i1.15696

Guo, T., Dong, J., Li, H., & Gao, Y. (2017). Simple convolutional neural network on image classification. *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, 721–724. https://doi.org/10.1109/ICBDA.2017.8078730

Hanif, A. (2022). *Identifikasi Masker Wajah Menggunakan Transfer Learning*. 1–23. https://eprints.utdi.ac.id/9547/3/3_155410154_BAB_II - Agung N. Hanif.pdf

Hauzan, S. A. (2023). Penerapan Convolutional Neural Network dalam Pengklasifikasian Citra Gambar Jamur Beracun. *Repository.Uinjkt.Ac.Id*. https://repository.uinjkt.ac.id/dspace/handle/123456789/72929%0Ahttps://repository.uinjkt.ac.id/dspace/bitstream/123456789/72929/1/SHIDQI AKRAM HAUZAN-FST.pdf

Ihsan, M., Niswatin, R. K., & Swanjaya, D. (2021). Deteksi Ekspresi Wajah Menggunakan Tensorflow. *Joutica*, *6*(1), 428. https://doi.org/10.30736/jti.v6i1.554

Maulana, F. F., & Rochmawati, N. (2020). Klasifikasi Citra Buah Menggunakan Convolutional Neural Network. *Journal of Informatics and Computer Science (JINACS)*, *1*(02), 104–108. https://doi.org/10.26740/jinacs.v1n02.p104-108

Mubarok, H. (2019). Identifikasi Ekspresi Wajah Berbasis Citra Menggunakan Algoritma Convolutional Neural Network (CNN). *Universitas Islam Negeri Maulana Malik Ibrahim Malang*, *3*(1), 10–12.

Oumina, A., El Makhfi, N., & Hamdi, M. (2020). Control the COVID-19 Pandemic: Face Mask Detection Using Transfer Learning. *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science, ICECOCS 2020*. https://doi.org/10.1109/ICECOCS50124.2020.9314511

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Suartika E. P, I Wayan, Wijaya Arya Yudhi, S. R. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) Pada Caltech 101. *Jurnal Teknik ITS*, *5*(1), 76. http://repository.its.ac.id/48842/

Sudalto. (2022). Implementasi Convolutional Neural Network untuk Klasifikasi Citra Soil Nikel dan Non Nikel. *G-Tech: Jurnal Teknologi Terapan*, *6*(1), 85–90. https://doi.org/10.33379/gtech.v6i1.1273

Sudeep, K. S., & Pal, K. K. (2017). Preprocessing for image classification by convolutional neural networks. *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings*, 1778–1781. https://doi.org/10.1109/RTEICT.2016.7808140

Wicaksono, G., Andryana, S., & -, B. (2020). Aplikasi Pendeteksi Penyakit Pada Daun Tanaman Apel Dengan Metode Convolutional Neural Network. *JOINTECS (Journal of Information Technology and Computer Science)*, *5*(1), 9. https://doi.org/10.31328/jointecs.v5i1.1221

Zaelani, F., & Miftahuddin, Y. (2022). Perbandingan Metode EfficientNetB3 dan MobileNetV2 Untuk Identifikasi Jenis Buah-buahan Menggunakan Fitur Daun. *Jurnal Ilmiah Teknologi Infomasi Terapan*, *9*(1), 1–11. https://doi.org/10.33197/jitter.vol9.iss1.2022.911

```
import os

import cv2

import math

import numpy as np

import pandas as pd

import tensorflow as tf

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import
train_test_split

from sklearn.metrics import confusion_matrix,
classification_report

from tensorflow.keras import optimizers

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense,
GlobalMaxPool2D

from tensorflow.keras.callbacks import Callback,
EarlyStopping, ReduceLROnPlateau

from tensorflow.keras.preprocessing.image import
ImageDataGenerator

from tensorflow.keras.applications.mobilenet
import MobileNet

from tensorflow.keras.applications import
MobileNetV2

from tensorflow.keras.utils import plot_model

from tensorflow.keras.utils import
to_categorical
```

```
# DATASET
Data = "Ekspresi/"

# Inisialisasi variabel
total_images = 0
for dir in os.listdir(Data):
    count = 0
    for f in os.listdir(Data + dir + "/"):
        count += 1
        total_images += 1
    print(f"{dir} has {count} number of images")
print(f"\ntotal images are {total_images}")

# Daftar ekspresi yang ingin diambil
Ekspresi_Multi = ["happy", "sad", "surprise"]
total_images

# Inisialisasi array untuk gambar dan label
img_arr = np.empty(shape=(total_images,48,48,3))
img_label = np.empty(shape=(total_images))
label_to_text = {}
i = 0
e = 0
for dir in os.listdir(Data):
    if dir in Ekspresi_Multi:
```

```
        label_to_text[e] = dir
        for f in os.listdir(Data + dir + "/"):
            img_arr[i] = cv2.imread(Data + dir +
"/" + f)
            img_label[i] = e
            i += 1
        print(f"loaded all {dir} images to numpy
arrays")
        e += 1
img_arr.shape, img_label
label_to_text


# Menampilkan sebagian plot dataset
fig = plt.figure(1, (9,9))
index = 0
for k in label_to_text:
    sample_indices =
np.random.choice(np.where(img_label==k)[0],
size=4, replace=False)
    sample_images = img_arr[sample_indices]
    for img in sample_images:
        index += 1
        ax = plt.subplot(4,4,index)
        ax.imshow(img[:,:,0], cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
```

```
        ax.set_title(label_to_text[k])
plt.tight_layout()
plt.show()
img_label = to_categorical(img_label)
img_label.shape


# sebelum dinormalisasi
img_arr


# Menampilkan sampel gambar sebelum
dinormalisasi
fig = plt.figure(figsize=(9, 9))


index = 0
for k, expression in enumerate(label_to_text):
    expression_indices = np.where(img_label ==
k)[0][:4]
    sample_images = img_arr[expression_indices]
    for img in sample_images:
        index += 1
        ax = plt.subplot(4, 4, index)
        ax.imshow(img[:,:,0], cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(label_to_text[expression])
```

```
plt.tight_layout()
plt.show()


# Normalisai data
img_arr = img_arr / 255.


# Sesudah di normalisasi
img_arr


# Menampilkan sampel gambar setelah
dinormalisasi
fig = plt.figure(figsize=(9, 9))


index = 0
for k, expression in enumerate(label_to_text):
    expression_indices = np.where(img_label[:,
k] == 1)[0][:4]
    sample_images = img_arr[expression_indices]
    for img in sample_images:
        index += 1
        ax = plt.subplot(4, 4, index)
        ax.imshow(img)
        ax.set_xticks([])
        ax.set_yticks([])
```

```
        ax.set_title(label_to_text[expression])


plt.tight_layout()

plt.show()


# Split data

X_train, X_test, y_train, y_test =
train_test_split(img_arr, img_label,
shuffle=True, stratify=img_label,
test_size=0.2, random_state=42)

X_train.shape, X_test.shape, y_train.shape,
y_test.shape


# Augmentation data

train_datagen = ImageDataGenerator(

    rotation_range=15,

    width_shift_range=0.15,

    height_shift_range=0.15,

    shear_range=0.15,

    zoom_range=0.15 )

train_datagen.fit(X_train)


num_images = 5

sample_images = X_train[:num_images]
```

```
augmented_images = []

for img in sample_images:

    augmented_img =
train_datagen.random_transform(img)

    augmented_images.append(augmented_img)




fig, axes = plt.subplots(nrows=2,
ncols=num_images, figsize=(20, 4))

for i in range(num_images):

    # Menampilkan gambar asli

    axes[0, i].imshow(sample_images[i])

    axes[0, i].set_title('Original')

    axes[0, i].axis('off')


    # Menampilkan hasil augmentasi

    axes[1, i].imshow(augmented_images[i])

    axes[1, i].set_title('Augmented')

    axes[1, i].axis('off')



plt.suptitle(f"Total Data: {len(X_train)}",
y=1.05, fontsize=16)

plt.tight_layout()

plt.show()
```

```
img_width = X_train.shape[1]

img_height = X_train.shape[2]

img_depth = X_train.shape[3]

num_classes = y_train.shape[1]

# Perancangan Model

base_model = MobileNetV2(

    input_shape = (img_width, img_height,
img_depth),

    include_top = False,

    weights = "imagenet",

    classes = num_classes)

model = base_model.layers[-14].output

model =
GlobalMaxPool2D(name="global_pool")(model)

model = Dense(num_classes, activation="softmax",
name="out_layer")(model)

model = Model(inputs=base_model.input,
outputs=model)

model.summary()

for layer in model.layers[:15]:

    layer.trainable = False


early_stopping = EarlyStopping(

    monitor='val_accuracy',

    min_delta=0.00008,

    patience=11,
```

```
    verbose=1,
    restore_best_weights=True,
)
lr_scheduler = ReduceLROnPlateau(
    monitor='val_accuracy',
    min_delta=0.0001,
    factor=0.25,
    patience=4,
    min_lr=1e-7,
    verbose=1,
)
callbacks = [
    early_stopping,
    lr_scheduler,
]

batch_size = 16
epochs = 50
optims = optimizers.Adam(learning_rate=0.001)

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

```
# Fit model
history = model.fit(
    train_datagen.flow(X_train, y_train,
batch_size=batch_size),
    validation_data=(X_test, y_test),
    steps_per_epoch=len(X_train) / batch_size,
    epochs=epochs,
    callbacks=callbacks
)


# Learning Curve
sns.set()
fig = plt.figure(figsize=(12, 4))


#learning curve Accuracy
ax = plt.subplot(1, 2, 1)
sns.lineplot(x=history.epoch,
y=history.history['accuracy'], label='train',
marker='o',)
sns.lineplot(x=history.epoch,
y=history.history['val_accuracy'],
label='valid', marker='o')
plt.title('Accuracy')
plt.tight_layout()


#learning curve Loss
ax = plt.subplot(1, 2, 2)
```

```
sns.lineplot(x=history.epoch,
y=history.history['loss'], label='train',
marker='o')

sns.lineplot(x=history.epoch,
y=history.history['val_loss'], label='valid',
marker='o')

plt.title('Loss')

plt.tight_layout()

plt.show()

#evaluasi model test

predict_test = np.argmax(model.predict(X_test),
axis=1)

ytest_ = np.argmax(y_test, axis=1)



# Menghitung confusion matrix

conf_matrix = confusion_matrix(ytest_,
predict_test)

test_accu = np.sum(ytest_ == predict_test) /
len(ytest_) * 100



#menampilkan hasil akurasi dan classification
report

print(classification_report(ytest_,
predict_test))

print(f"Test Accuracy: {round(test_accu, 4)}
%\n\n")



# Membuat plot confusion matrix

plt.figure(figsize=(8, 6))
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d',
cmap='Reds', cbar=True)

plt.xlabel('Predicted labels')

plt.ylabel('True labels')

plt.title('Confusion Matrix')

plt.show()

# Pengujian Data klasifikasi yang benar

def plot_true_classified(ekspresi):

    true_indices = np.where((ytest_ ==
predict_test) & (ytest_==label[ekspresi]))[0]

    print(f"total {len(true_indices)} true
labels out of
{len(np.where(ytest_==label[ekspresi])[0])} for
ekspresi {ekspresi}")



    # Tampilkan hanya 10 gambar

    true_indices = true_indices[:10]

    cols = 5

    rows = 2

    fig = plt.figure(1, (20, rows * 2))



    for i, idx in enumerate(true_indices):

        sample_img = X_test[idx,:,:,:]

        sample_img =
sample_img.reshape(1,*sample_img.shape)

        true_label = label_to_text[ytest_[idx]]
# Mendapatkan Kelas yang benar dari ytest_
```

```
        pred =
label_to_text[np.argmax(model.predict(sample_img
), axis=1)[0]]


        ax = plt.subplot(rows, cols, i+1)

        ax.imshow(sample_img[0,:,:,0],
cmap='gray')

        ax.set_xticks([])

        ax.set_yticks([])

        ax.set_title(f"True:{true_label},
Predicted:{pred}")

#pemanggilan fungsi

plot_true_classified('happy')

plot_true_classified('sad')

plot_true_classified('surprise')


# Pengujian Data klasifikasi yang salah

def plot_miss_classified(ekspresi):

    miss_indices = np.where((ytest_ !=
predict_test) & (ytest_==label[ekspresi]))[0]

    print(f"total {len(miss_indices)} miss
labels out of
{len(np.where(ytest_==label[ekspresi])[0])} for
emotion {ekspresi}")


    # Tampilkan hanya 10 gambar

    miss_indices = miss_indices[:10]

    cols = 5
```

```
    rows = 2

    fig = plt.figure(1, (20, rows * 2))



    for i,idx in enumerate(miss_indices):

        sample_img = X_test[idx,:,:,:]

        sample_img =
sample_img.reshape(1,*sample_img.shape)

        true_label = label_to_text[ytest_[idx]]
# Mendapatkan label yang benar dari ytest_

        pred =
label_to_text[np.argmax(model.predict(sample_img
), axis=1)[0]]



        ax = plt.subplot(rows,cols,i+1)

        ax.imshow(sample_img[0,:,:,0],
cmap='gray')

        ax.set_xticks([])

        ax.set_yticks([])

        ax.set_title(f"p:{pred}")

        ax.set_title(f"True:{true_label},
Predicted:{pred}")
```

#pemanggilan fungsi

```
plot_miss_classified('happy')

plot_miss_classified('sad')

plot_miss_classified('surprise')
```