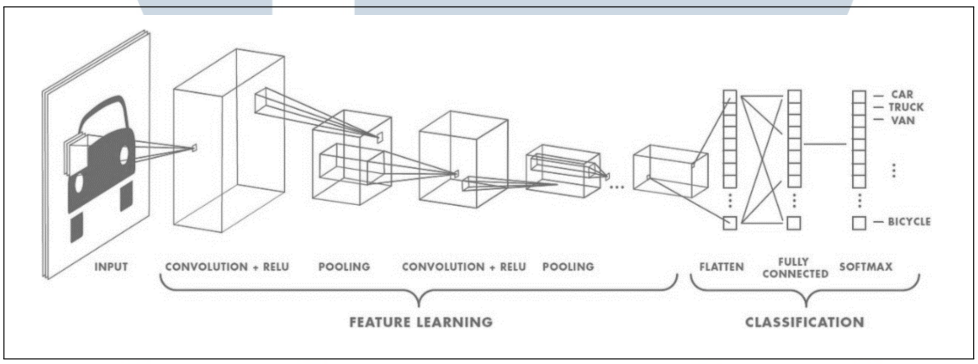
BAB 2
LANDASAN TEORI

2.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah jenis *deep learning* yang dirancang khusus untuk pengolahan data grid, seperti citra [7]. CNN terinspirasi oleh arsitektur visual mamalia, sebagaimana dijelaskan dalam penelitian Hubel dan Wiesel, yang menemukan bahwa neuron tertentu di otak merespons stimulus visual dalam area terbatas (*receptive field*) [8]. Bidang reseptif neuron saling bertumpang tindih secara parsial (*partially overlap*) sebagian sehingga bersama-sama mencakup seluruh area reseptif [9].



Gambar 2.1. Arsitektur CNN

Sumber: [10]

Arsitektur CNN seperti yang ada pada Gambar 2.1, di mana terdapat *input layer*, *output layer*, dan *hidden layers* [11]. *Hidden layers* inicterdiri dari lapisan-lapisan utama, yaitu lapisan konvolusi, lapisan *pooling*, dan lapisan *fully connected*, yang bersama-sama mengekstrak fitur dari input citra untuk menghasilkan prediksi yang akurat.

2.1.1 Convolutional Layer

Lapisan neuron konvolusi merupakan komponen utama dalam CNN. Dalam tugas klasifikasi citra, satu atau lebih matriks 2D (atau saluran) digunakan sebagai input untuk lapisan konvolusi, dan beberapa matriks 2D dihasilkan sebagai output. Jumlah matriks input dan output dapat berbeda. Selain jumlah matriks, ukuran *feature map output* juga dapat berbeda karena dipengaruhi oleh ukuran

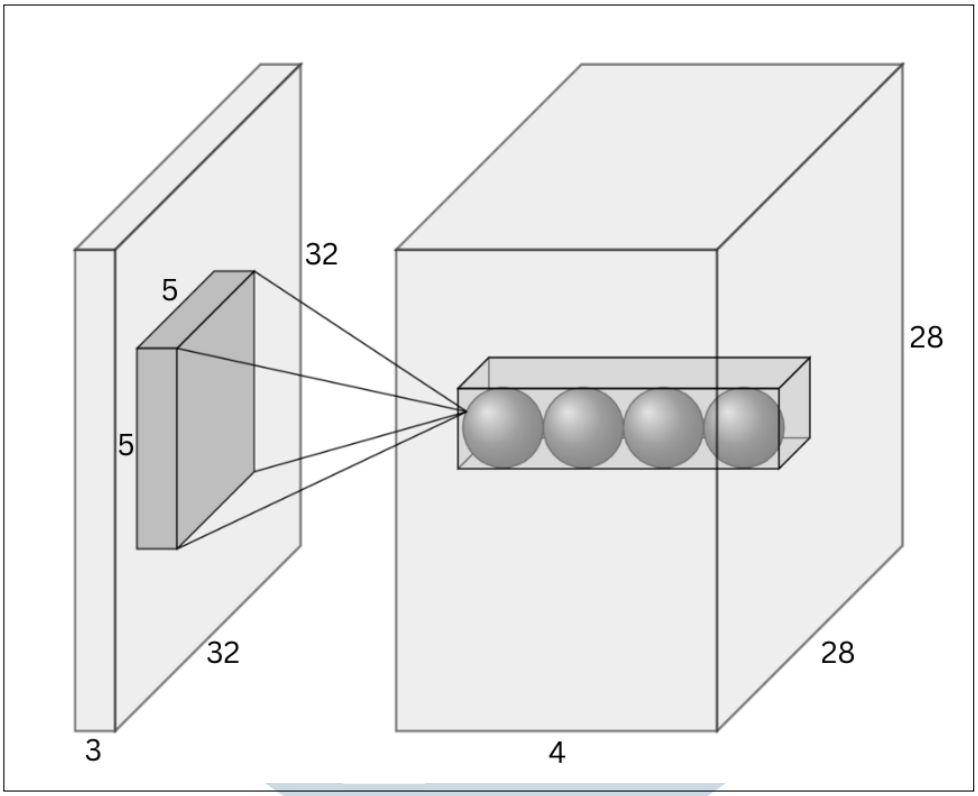
filter, *padding*, dan *stride*. *Padding* digunakan untuk mempertahankan dimensi, sedangkan *stride* menentukan seberapa jauh filter bergeser di setiap langkah [11]. Proses untuk menghitung satu matriks output didefinisikan sebagai berikut:

$$A_j = f\left(\sum_{i=1}^{N} I_i * K_{i,j} + B_j\right) \tag{2.1}$$

Persamaan 2.1 adalah operasi konvolusi, dengan penjelasan sebagai berikut:

- I_i adalah matriks input ke-i,
- $K_{i,j}$ adalah kernel yang diaplikasikan pada input ke-i untuk menghasilkan output ke-j,
- B_j adalah nilai bias,
- f adalah fungsi aktivasi non-linear.

Kernel bekerja dengan cara melintasi citra secara lokal, menghitung dot product antara kernel dan sub-matriks pada citra input. Setiap kernel menangkap fitur tertentu, seperti tepi, sudut, atau tekstur. Dengan menambahkan lebih banyak lapisan konvolusi, jaringan dapat menangkap fitur yang lebih kompleks, seperti pola geometris atau objek spesifik dalam gambar [7] [12].



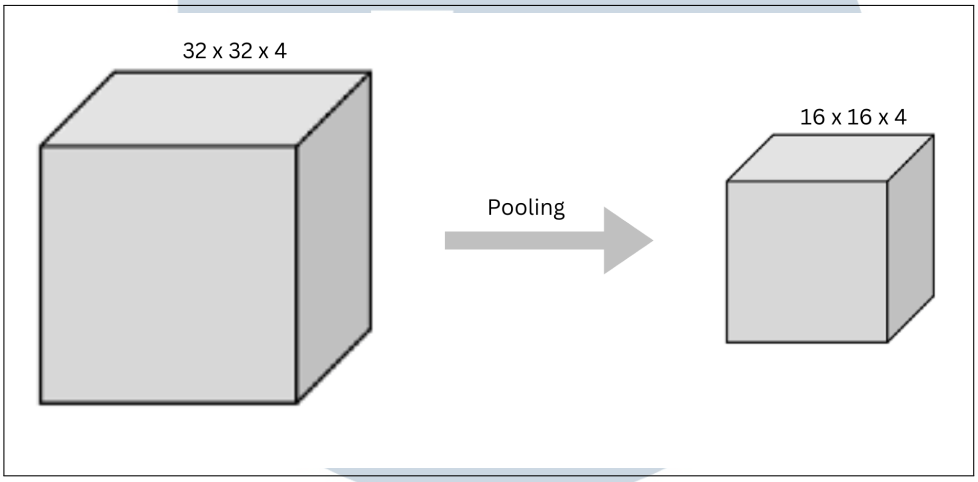
Gambar 2.2. *Convolutional layer* dengan empat *filters* berukuran 5 x 5
Sumber: [11]

Gambar 2.2 menggambarkan proses konvolusi dalam CNN, di mana sebuah filter berukuran 5 × 5 diterapkan pada input berukuran 32 × 32 × 3, menghasilkan *feature map* berukuran 28 × 28 × 4. Input memiliki 3 saluran (misalnya, RGB), dan setelah konvolusi dengan 4 filter, hasilnya memiliki 4 saluran. Setiap filter bekerja dengan menggeser jendela kecil (5 × 5) di seluruh citra input, menghitung dot product dengan nilai piksel dalam jendela tersebut, lalu menerapkan fungsi aktivasi. Karena tidak ada padding, ukuran output lebih kecil dibandingkan input, dihitung dengan rumus (32 - 5 + 1)×(32 - 5 + 1) = 28 × 28. Lapisan konvolusi ini memungkinkan CNN menangkap fitur penting seperti tepi, pola tekstur, dan bentuk dalam gambar.

Tidak seperti *fully connected layer* dalam jaringan neural tradisional, lapisan konvolusi hanya terhubung dengan bagian kecil dari lapisan sebelumnya, yang disebut *local receptive field*. Hal ini membuat CNN lebih efisien dalam menangkap pola lokal seperti tepi, sudut, dan tekstur dalam gambar.

2.1.2 Pooling Layer

Lapisan *pooling* digunakan untuk lebih mengurangi ukuran *feature map* yang dihasilkan oleh lapisan konvolusi dengan tetap mempertahankan fitur yang paling signifikan. Proses ini menyaring nilai-nilai kecil dalam peta fitur, sehingga output menjadi lebih tahan terhadap gangguan atau *noise* [13]. Gambar 2.3 menunjukkan proses pooling untuk mereduksi dimensi data.



Gambar 2.3. Proses *pooling* untuk mereduksi dimensi data

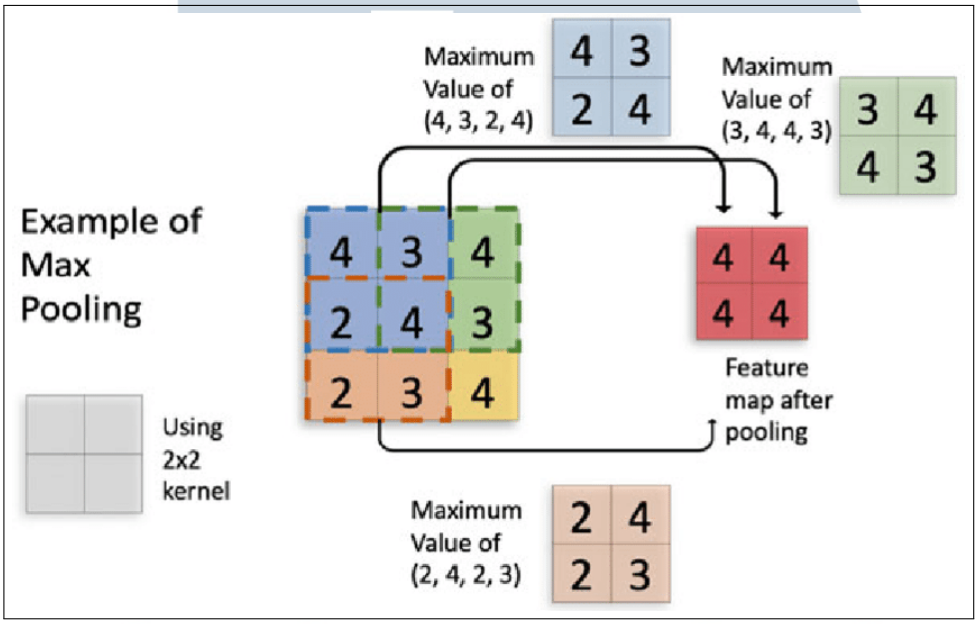
Sumber: [11]

Gambar 2.3 menunjukkan proses *pooling* dalam CNN, di mana sebuah *feature map* berukuran 32 × 32 × 4 direduksi menjadi 16 × 16 × 4 setelah operasi *pooling*. *Pooling* berfungsi untuk mengurangi dimensi data sekaligus mempertahankan informasi penting dalam *feature map*, sehingga meningkatkan efisiensi komputasi dan membuat model lebih tahan terhadap pergeseran atau distorsi kecil dalam input. Dalam ilustrasi ini, hanya ukuran spasial (32 × 32 menjadi 16 × 16) yang mengalami reduksi, sementara jumlah saluran (*depth*) tetap 4, menunjukkan bahwa pooling diterapkan secara independen pada setiap saluran fitur.

Pooling layer memiliki beberapa jenis, yaitu, *max pooling*, *average pooling*, dan *soft pooling*. Umumnya, digunakan *max pooling* sebagai metode penggabungan yang digunakan dalam CNN.

A Max Pooling

Max pooling adalah metode *pooling* yang paling umum digunakan dalam CNN. Teknik ini memilih nilai maksimum dari setiap wilayah *pooling*, misalnya pada matriks ukuran $k \times k$. Proses ini membantu jaringan untuk menangkap fitur paling signifikan dalam area tertentu, seperti tepi atau sudut tajam seperti yang diilustrasikan pada Gambar 2.4.



Gambar 2.4. Ilustrasi Operasi *Max Pooling*

Sumber: [14]

Persamaan matematis untuk *max pooling* dapat dituliskan sebagai [14]:

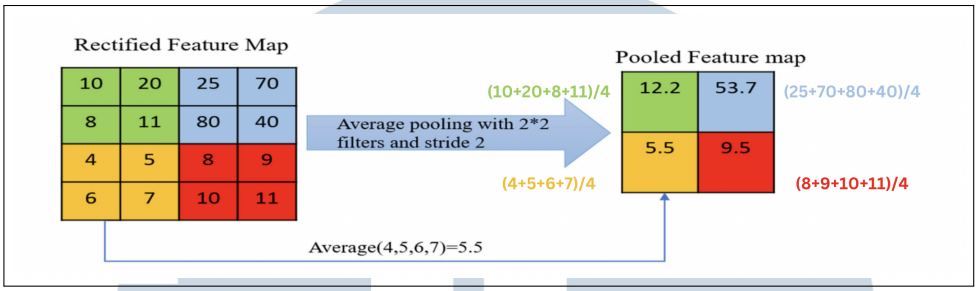
$$f_{max}(X) = max_i x_i \tag{2.2}$$

Kelemahan dari *max pooling* adalah mengabaikan informasi lain dalam area *pooling*, yang dapat menyebabkan kehilangan detail penting jika elemen lain memiliki nilai signifikan.

B Average Pooling

Average pooling menghitung nilai rata-rata dari semua elemen dalam wilayah *pooling*. Teknik ini berguna untuk mereduksi ukuran fitur sambil

mempertahankan distribusi informasi secara keseluruhan seperti yang diilustrasikan pada Gambar 2.5.



Gambar 2.5. Ilustrasi Operasi *Average Pooling*
Sumber: [14]

Persamaan untuk *average pooling* adalah [14]:

$$f_{avg}(X) = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.3}$$

di mana N adalah jumlah elemen dalam wilayah pooling. Namun, *average pooling* dapat melemahkan kontras antara elemen dengan nilai tinggi dan rendah, sehingga sensitivitas terhadap fitur tertentu menjadi berkurang.

C Soft Pooling

Soft pooling menggabungkan kelebihan dari *max pooling* dan *average pooling* dengan memberikan bobot pada setiap elemen berdasarkan nilai aktivasi. Aktivasi yang lebih tinggi mendapatkan bobot lebih besar dibandingkan dengan aktivasi yang lebih rendah [14].

Bobot dihitung berdasarkan fungsi eksponensial normalisasi sebagai berikut:

$$w_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.4}$$

dan hasil poolingnya adalah:

$$f_{soft}(X) = \sum_i w_i x_i \tag{2.5}$$

Teknik ini lebih adaptif karena mempertimbangkan semua elemen dalam wilayah *pooling*, namun lebih mahal secara komputasi dibandingkan *max pooling* atau *average pooling*.

2.1.3 Activation Layer

Fungsi aktivasi merupakan elemen kunci dalam *Convolutional Neural Networks* (CNN) yang memperkenalkan non-linearitas ke dalam jaringan, memungkinkan model untuk mempelajari hubungan kompleks dalam data. Salah satu fungsi aktivasi yang paling umum digunakan adalah ReLU (*Rectified Linear Unit*), yang didefinisikan sebagai [14]:

$$f(x) = \max(0, x) \tag{2.6}$$

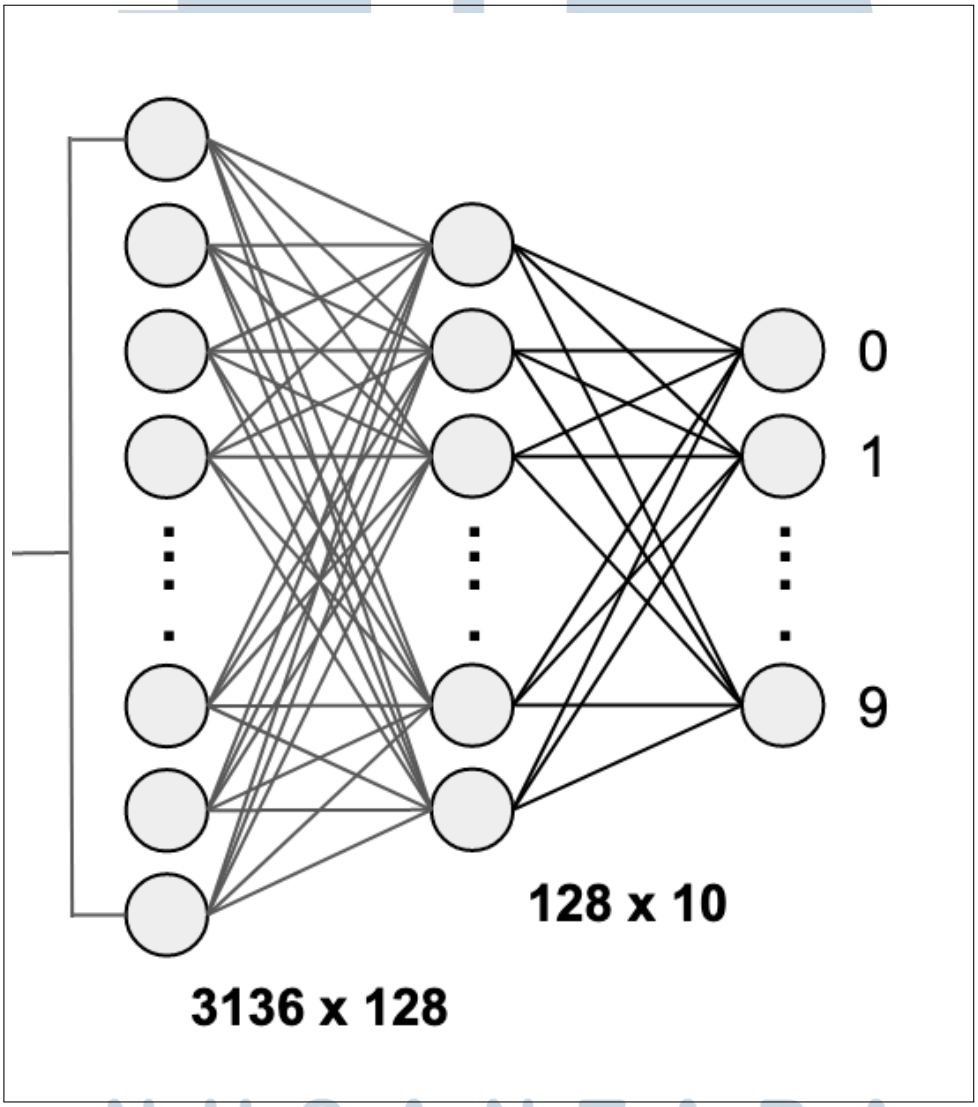
Meskipun ReLU (*Rectified Linear Unit*) banyak digunakan karena efisiensinya dan kemampuannya dalam mengurangi masalah *vanishing gradient*, fungsi ini memiliki kelemahan seperti *dead neurons*, di mana nilai negatif sepenuhnya ditekan sehingga beberapa neuron tidak pernah diaktifkan. Untuk mengatasi keterbatasan ini, beberapa fungsi aktivasi alternatif telah dikembangkan.

Salah satu alternatifnya adalah Leaky ReLU, yang memperkenalkan kemiringan kecil untuk input negatif agar mencegah neuron mati. Parametric ReLU (PReLU) lebih lanjut mengembangkan konsep ini dengan memungkinkan model mempelajari parameter kemiringan secara otomatis selama pelatihan. Sementara itu, fungsi aktivasi seperti Sigmoid dan Tanh, yang digunakan dalam jaringan saraf awal, mengalami masalah vanishing gradient akibat efek saturasi, di mana gradien menjadi sangat kecil pada rentang nilai yang tinggi atau rendah sehingga memperlambat proses pembelajaran [15]. Masalah ini menjadi semakin signifikan pada arsitektur CNN yang lebih dalam, menyebabkan konvergensi yang lambat dan performa yang kurang optimal dalam tugas klasifikasi yang kompleks.

Selain itu, fungsi aktivasi seperti *Softmax* lebih sering digunakan dalam klasifikasi multi-kelas karena mampu mengonversi nilai mentah (*logits*) menjadi distribusi probabilitas yang lebih stabil.

2.1.4 Fully Connected Layer

Lapisan *fully connected* adalah bagian terakhir dari *Convolutional Neural Network* (CNN) yang bertanggung jawab untuk mengintegrasikan semua fitur yang diekstraksi oleh lapisan konvolusi dan *pooling*, guna menghasilkan output akhir. Dalam lapisan ini, setiap neuron dihubungkan secara penuh dengan setiap neuron di lapisan sebelumnya, sehingga memungkinkan jaringan untuk menggabungkan informasi dari seluruh peta fitur yang dihasilkan.



Gambar 2.6. Ilustrasi Operasi *Fully Connected Layer*

Gambar 2.6 menunjukkan operasi Fully Connected Layer dalam jaringan saraf tiruan, di mana setiap neuron di satu lapisan terhubung ke semua neuron di

lapisan berikutnya. Dalam arsitektur ini, lapisan pertama memiliki 3136 neuron yang dikoneksikan ke 128 neuron di lapisan tersembunyi, dan selanjutnya ke 10 neuron di lapisan output, yang mewakili jumlah kelas dalam klasifikasi. Setiap koneksi menyimpan bobot yang dipelajari selama pelatihan untuk memetakan fitur yang diekstraksi ke prediksi akhir.

Fully connected layer sering digunakan dalam tugas klasifikasi, di mana output biasanya berupa probabilitas untuk setiap kelas melalui fungsi aktivasi seperti *softmax* [14] [16]. Proses di *fully connected layer* dapat diformulasikan sebagai berikut:

$$y = f(Wx + b) \tag{2.7}$$

Di mana:

- y adalah output dari lapisan,
- W adalah matriks bobot yang menghubungkan neuron input dengan neuron output,
- x adalah vektor input dari lapisan sebelumnya,
- b adalah vektor bias,
- f adalah fungsi aktivasi seperti ReLU atau sigmoid.

Pada tugas klasifikasi, fungsi *softmax* sering digunakan di lapisan akhir untuk mengubah skor logit menjadi distribusi probabilitas:

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{2.8}$$

Di mana:

- $P(y_i)$ adalah probabilitas prediksi untuk kelas i,
- z_i adalah skor logit untuk kelas i,
- N adalah jumlah kelas.

Untuk mencegah *overfitting*, regularisasi seperti *dropout* sering diterapkan pada lapisan *fully connected*. *Dropout* bekerja dengan menonaktifkan sejumlah

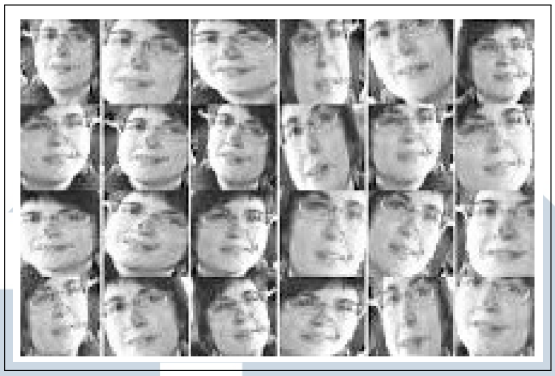
neuron secara acak selama pelatihan, yang memaksa jaringan untuk belajar dari berbagai kombinasi fitur dan meningkatkan generalisasi pada data baru [14].

Fully connected layer sangat efektif dalam menggabungkan fitur, namun memiliki jumlah parameter yang besar dibandingkan lapisan konvolusi. Hal ini meningkatkan risiko *overfitting*, terutama pada dataset kecil. Oleh karena itu, penggunaan *fully connected layer* yang optimal sangat penting dalam desain CNN, terutama untuk menghasilkan prediksi yang akurat di berbagai aplikasi seperti klasifikasi citra dan deteksi objek.

2.2 Data Augmentation

Data augmentasi adalah teknik yang digunakan dalam pembelajaran mesin, terutama dalam pelatihan model berbasis *Convolutional Neural Network* (CNN), untuk meningkatkan variasi dan ukuran dataset. Metode ini menciptakan data baru dari dataset yang ada dengan melakukan transformasi seperti rotasi, pemotongan, pencerminan, pergeseran, zoom, dan perubahan intensitas warna. Dengan demikian, data augmentasi membantu model menggeneralisasi lebih baik terhadap data yang belum terlihat sebelumnya dan mengurangi risiko *overfitting* [17] [18].

Dalam penelitian menggunakan citra X-ray, data augmentasi digunakan untuk meningkatkan keragaman data dengan *rescale*, *flipping horizontal*, dan *zooming*. Teknik ini membuat model lebih tahan terhadap perubahan kecil dalam citra, seperti posisi atau orientasi objek [18]. Di sisi lain, penelitian pengenalan ekspresi wajah menunjukkan bahwa data augmentasi meningkatkan akurasi validasi hingga 96,24% dengan menghasilkan citra baru dari dataset asli seperti yang ada pada Gambar 2.7, memungkinkan model untuk mempelajari pola yang lebih luas, bahkan dengan dataset yang terbatas [17].



Gambar 2.7. Contoh Hasil Data Augmentasi pada dataset ekspresi wajah
Sumber: [17]

Selain itu, teknik augmentasi sering digunakan bersama dengan fungsi seperti ImageDataGenerator dalam *library* Keras, yang mempermudah implementasi berbagai transformasi augmentasi secara efisien selama pelatihan model [17]. Penggunaan data augmentasi ini sangat penting untuk memperluas kemampuan model dalam memahami pola yang kompleks di berbagai aplikasi.

2.3 Ruang Warna YCbCr

Ruang warna YCbCr adalah salah satu model warna yang sering digunakan dalam sistem pengolahan gambar digital dan video karena efisiensinya dibandingkan dengan representasi langsung seperti RGB. Dalam model ini, setiap warna direpresentasikan dengan tiga komponen:

- **Y**: Menggambarkan intensitas atau kecerahan (*luminance*),
- **Cb**: Mengindikasikan perbedaan intensitas antara biru dan hijau (*chrominance*),
- **Cr**: Mengindikasikan perbedaan intensitas antara merah dan hijau (*chrominance*).

Keunggulan utama ruang warna YCbCr adalah kemampuan memanfaatkan sifat persepsi manusia yang lebih sensitif terhadap perubahan intensitas cahaya daripada perubahan warna. Dengan demikian, informasi intensitas (Y) dapat disimpan dengan presisi tinggi, sedangkan informasi warna (Cb dan Cr) dapat dikompresi tanpa mengorbankan kualitas gambar secara signifikan [7] [19]. Konversi dari RGB ke YCbCr dilakukan menggunakan persamaan berikut:

$$Y = 0.114B + 0.587G + 0.299R \tag{2.9}$$

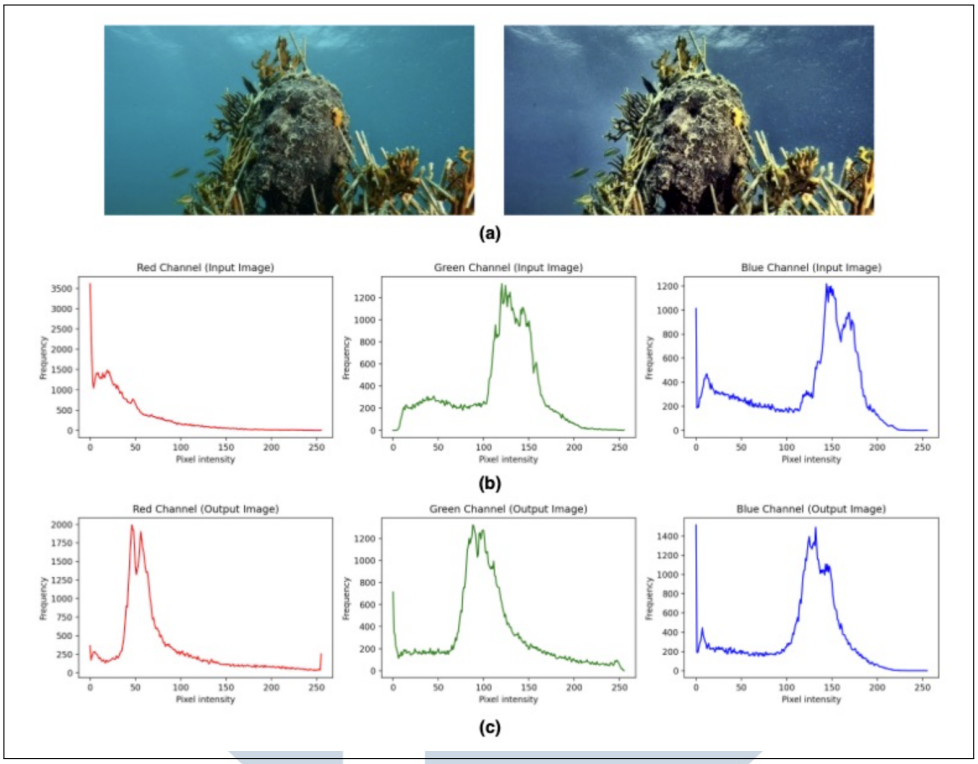
$$Cb = 0.564(B - Y) \tag{2.10}$$

$$Cr = 0.713(R - Y) \tag{2.11}$$

Dalam aplikasi seperti kompresi JPEG dan pengolahan video, ruang warna YCbCr digunakan untuk mengurangi data yang tidak terlalu penting secara visual. Misalnya, *subsampling chrominance* (*Cb* dan *Cr*) sering dilakukan dengan pola seperti 4:2:0 atau 4:2:2, yang mengurangi resolusi warna dibandingkan dengan luminance [7].

2.4 Color Histogram Features

Histogram warna adalah representasi distribusi intensitas dalam sebuah citra, di mana sumbu horizontal merepresentasikan nilai intensitas (atau level abu-abu pada gambar grayscale dan kanal warna pada gambar berwarna) seperti yang ada pada Gambar 2.8, sementara sumbu vertikal menunjukkan jumlah piksel pada setiap intensitas. Untuk membangun histogram, nilai intensitas piksel dikelompokkan ke dalam interval-interval yang disebut bins. Setiap bin mewakili rentang tertentu dari nilai intensitas, dan jumlah piksel yang memiliki nilai dalam rentang tersebut dihitung untuk membentuk tinggi bin.



Gambar 2.8. Contoh konversi dari YCbCr ke RGB. (a) Gambar referensi (gambar input dan output), (b) Distribusi histogram gambar output YCbCr, (c) Distribusi histogram gambar output RGB

Sumber: [20]

Penggunaan bins memberikan fleksibilitas dalam representasi histogram. Dengan jumlah bins yang lebih banyak, histogram menjadi lebih detail, menangkap variasi kecil dalam distribusi intensitas. Namun, hal ini juga dapat meningkatkan sensitivitas terhadap noise. Sebaliknya, jumlah bins yang lebih sedikit menghasilkan histogram yang lebih halus, tetapi dapat kehilangan detail penting pada distribusi intensitas. Jumlah bins sering kali dipilih berdasarkan aplikasi tertentu, dengan mempertimbangkan trade-off antara detail dan kehalusan.

Dalam aplikasi warna, histogram dapat dihitung untuk masing-masing kanal warna (misalnya, RGB atau YCbCr), menghasilkan histogram terpisah untuk setiap komponen warna. Pendekatan ini memungkinkan analisis distribusi intensitas dan warna secara mendalam, yang sangat berguna dalam pengenalan pola, klasifikasi gambar, dan pengambilan gambar berbasis konten. Pengaturan bins yang tepat dapat meningkatkan akurasi dalam analisis ini dengan menangkap pola-pola visual yang relevan dari data gambar.

Histogram juga digunakan untuk merepresentasikan model distribusi

probabilitas intensitas gambar, di mana probabilitas suatu intensitas dihitung sebagai rasio antara jumlah piksel pada intensitas tersebut dengan jumlah total piksel dalam gambar [5]. Probabilitas histogram didefinisikan sebagai:

$$P(g) = \frac{N(g)}{M} \tag{2.12}$$

Di mana:

- $P(g)$: Probabilitas untuk intensitas g,
- $N(g)$: Jumlah piksel dengan intensitas g,
- M: Jumlah total piksel dalam gambar.

Kode 2.1 pseudocode yang menggambarkan langkah-langkah utama dalam ekstraksi fitur berdasarkan distribusi warna dari suatu citra. Proses ini mencakup perhitungan histogram dengan properti statistik, normalisasi, dan penyimpanan fitur yang dihasilkan.

```
1. Inisialisasi vektor fitur F = []
2. Dapatkan dimensi gambar: H, W, C
3. Hitung total jumlah piksel: M = H * W
4. Konversi gambar ke ruang warna YCbCr
5. Untuk setiap kanal warna C dalam {Y, Cb, Cr}:
    a. Hitung histogram N(g) dengan B bin
    b. Normalisasi histogram: P(g) = N(g) / M
    c. Tambahkan hasil normalisasi P(g) ke vektor fitur F
6. Kembalikan vektor fitur F
```

Kode 2.1: Algoritma Ekstraksi Fitur Histogram Warna

Properti histogram memberikan informasi statistik yang penting tentang karakteristik global sebuah gambar dan banyak digunakan dalam klasifikasi berbasis histogram. Beberapa properti statistik histogram yang sering digunakan untuk ekstraksi fitur mencakup mean, standard deviation, skew, energy, dan entropy.

2.4.1 Mean (Rata-rata)

Mean mencerminkan kecerahan rata-rata sebuah gambar, dengan nilai yang lebih tinggi menunjukkan gambar yang lebih terang. Mean mencerminkan kecerahan rata-rata gambar dengan persamaan:

$$\bar{g} = \sum_{g=0}^{L-1} g \cdot P(g) \tag{2.13}$$

Mean yang tinggi menunjukkan gambar yang lebih terang, sementara mean yang rendah menunjukkan gambar yang lebih gelap.

2.4.2 Standard Deviation

Standard deviation, di sisi lain, mengukur kontras atau variasi intensitas dalam gambar, di mana nilai yang lebih tinggi menunjukkan kontras yang lebih besar. Standar deviasi menggambarkan kontras gambar dengan persamaan:

$$\sigma_g = \sqrt{\sum_{g=0}^{L-1} (g - \bar{g})^2 \cdot P(g)} \tag{2.14}$$

2.4.3 Skew (Kemiringan)

Skew mengukur asimetri distribusi intensitas terhadap mean. Skew positif menunjukkan distribusi condong ke nilai intensitas tinggi, sementara skew negatif menunjukkan distribusi condong ke nilai intensitas rendah. Skew mengukur asimetri distribusi intensitas dengan persamaan:

$$\text{Skew} = \frac{1}{\sigma_g^3} \sum_{g=0}^{L-1} (g - \bar{g})^3 \cdot P(g) \tag{2.15}$$

2.4.4 Energy

Energy memberikan informasi tentang konsentrasi distribusi intensitas, dengan nilai tinggi menunjukkan distribusi yang terkonsentrasi pada beberapa level intensitas. Energy menunjukkan tingkat konsentrasi distribusi intensitas dengan persamaan:

$$\text{Energy} = \sum_{g=0}^{L-1} [P(g)]^2 \tag{2.16}$$

Nilai energy tinggi berarti distribusi intensitas terkonsentrasi pada beberapa level, mempermudah kompresi data gambar.

2.4.5 Entropy

Entropy, sebaliknya, mengukur kompleksitas gambar, di mana nilai yang lebih tinggi menunjukkan distribusi intensitas yang lebih merata dan kompleksitas visual yang lebih tinggi. Entropy mengukur kompleksitas gambar, atau jumlah informasi yang dikandung dengan persamaan:

$$\text{Entropy} = -\sum_{g=0}^{L-1} P(g) \cdot \log_2[P(g)] \tag{2.17}$$

Properti histogram ini sangat berguna dalam klasifikasi gambar berbasis warna atau intensitas karena mudah dihitung, efisien, dan memberikan representasi global yang bermanfaat untuk memahami pola dalam sebuah gambar. Dalam penelitian ini, fitur histogram digunakan untuk klasifikasi gambar dalam sistem pengambilan gambar berbasis konten (CBIR) dan menghasilkan efisiensi komputasi yang tinggi, menjadikannya metode yang ideal untuk aplikasi dengan kebutuhan komputasi cepat [5].

2.5 Evaluasi Model

Evaluasi model adalah proses untuk mengukur kinerja dan menentukan tingkat keberhasilan dari model yang telah dibangun. Evaluasi dilakukan dengan menggunakan data uji (*testing data*) yang tidak terlibat dalam pelatihan model. Proses ini membantu menilai seberapa baik model dapat membuat prediksi yang akurat dan generalis untuk data yang baru. Beberapa metrik evaluasi utama yang digunakan dalam klasifikasi adalah **akurasi**, **presisi**, **recall**, dan **F1-Score**, yang semuanya dihitung berdasarkan **confusion matrix**.

2.5.1 Confusion Matrix

Confusion matrix adalah representasi matriks dari hasil prediksi model klasifikasi. Matriks ini berisi empat komponen yang menggambarkan hasil prediksi model dibandingkan dengan label aktual. Empat komponen utama dalam confusion matrix adalah:

- **True Positive (TP)**: Prediksi positif yang benar.
- **True Negative (TN)**: Prediksi negatif yang benar.
- **False Positive (FP)**: Prediksi positif yang salah.
- **False Negative (FN)**: Prediksi negatif yang salah.

Dari confusion matrix, berbagai metrik evaluasi dapat dihitung untuk memahami performa model.

2.5.2 Akurasi

Akurasi mengukur proporsi prediksi yang benar dari total keseluruhan prediksi [11]. Akurasi sering digunakan sebagai metrik awal untuk menilai model, terutama jika kelas-kelas dalam data seimbang. Rumus akurasi adalah:

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.18}$$

Namun, akurasi dapat menjadi kurang representatif ketika data tidak seimbang, yaitu ketika satu kelas memiliki lebih banyak sampel dibandingkan kelas lain.

2.5.3 Presisi

Presisi mengukur proporsi prediksi positif yang benar-benar positif [11]. Ini menunjukkan seberapa baik model menghindari kesalahan prediksi positif yang salah (*False Positive*). Rumus presisi adalah:

$$\text{Presisi} = \frac{TP}{TP + FP} \tag{2.19}$$

Presisi penting dalam situasi di mana kesalahan prediksi positif harus dihindari, seperti dalam diagnosis medis, di mana kesalahan mengidentifikasi seseorang yang sehat sebagai sakit dapat menimbulkan konsekuensi serius.

2.5.4 Recall

Recall (atau sensitivitas) mengukur kemampuan model untuk menemukan semua sampel positif yang benar[11]. Recall penting ketika kesalahan prediksi negatif yang salah (*False Negative*) harus dihindari. Rumus recall adalah:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.20}$$

Recall lebih relevan dalam situasi di mana kesalahan dalam tidak mengidentifikasi kasus positif lebih merugikan, seperti dalam deteksi penyakit.

2.5.5 F1-Score

F1-Score adalah rata-rata harmonis dari presisi dan recall, yang memberikan gambaran seimbang antara kedua metrik. F1-Score berguna ketika penting untuk menyeimbangkan antara *False Positive* dan *False Negative*. Rumus F1-Score adalah:

$$F1 = 2 \times \frac{\text{Presisi} \times \text{Recall}}{\text{Presisi} + \text{Recall}} \tag{2.21}$$

F1-Score sangat cocok untuk digunakan pada dataset yang tidak seimbang, di mana salah satu kelas jauh lebih banyak dibandingkan kelas lain.