BAB II
LANDASAN TEORI

2.1. Pergerakan Mata

Analisis pergerakan mata pergerakan mata merupakan salah satu indikator yang kuat dalam klasifikasi pergerakan mata. Beberapa konsep penting dalam analisis pergerakan mata:

1. Anatomi dan Fisiologi

   Mata Pemahaman tentang struktur anatomi mata seperti kornea, iris, pupil, dan otot-otot ekstraokular sangat penting untuk memahami mekanisme pergerakan mata [15] .Selain itu, proses fisiologis dan neurologis yang mengontrol pergerakan mata juga perlu dipahami dengan baik.

2. Mekanisme Pergerakan Mata

   Pergerakan mata melibatkan dua jenis gerakan utama, yaitu sakade (gerakan cepat) dan fiksasi (fokus visual). Sakade memungkinkan mata untuk bergerak dari satu titik fokus ke titik fokus lainnya, sementara fiksasi memungkinkan mata untuk fokus pada objek tertentu untuk memperoleh informasi visual [16] .

3. Hubungan dengan Kondisi Psikologis

   Pergerakan mata dapat dipengaruhi oleh kondisi psikologis seseorang, seperti stres, kecemasan, dan usaha kognitif yang lebih besar. Saat berbohong, seseorang cenderung mengalami peningkatan kecemasan dan beban kognitif, yang dapat memengaruhi pola pergerakan matanya [17].

4. Isyarat Kebohongan melalui Pergerakan Mata

   Penelitian menunjukkan bahwa terdapat beberapa pola pergerakan mata yang terkait dengan kebohongan, seperti menghindari kontak mata, peningkatan jumlah kedipan, dan pergerakan horizontal yang berlebihan [18]. Pola-pola ini dapat digunakan sebagai indikator untuk mendeteksi kebohongan melalui analisis pergerakan mata.

2.2. Pengolahan Citra Digital

Pengolahan citra *digital* (*Digital Image Processing*) adalah cabang ilmu yang mempelajari beragam teknik dalam memanipulasi citra visual. Citra yang dimaksudkan mencakup gambar diam (foto) maupun gambar bergerak (yang diambil dari *webcam*). Penggunaan istilah "*digital*" di sini menunjukkan bahwa manipulasi citra dilakukan melalui perangkat computer [19] .

Secara matematis, citra dapat dianggap sebagai fungsi kontinu yang menggambarkan intensitas cahaya pada bidang dua dimensi. Untuk dapat diolah menggunakan komputer digital, citra harus diubah menjadi representasi numerik dengan nilai-nilai diskrit. Proses konversi dari fungsi kontinu ke nilai-nilai diskrit ini dikenal sebagai digitalisasi citra.

Representasi sebuah citra digital dapat dilakukan melalui matriks dua dimensi $f(x,y)$ yang memiliki *M* kolom dan *N* baris. Setiap perpotongan kolom dan baris dalam matriks tersebut disebut piksel (*pixel*), yang merupakan elemen terkecil dari citra tersebut.

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(M-1.1) & \cdots & f(N-1,M-1) \end{bmatrix} \qquad (2.1)$$

Suatu citra f(x,y) dalam fungsi matematis dapat dituliskan sebagai berikut:

$$0 \leq \mathrm{x} \leq \mathrm{M}-1$$
$$0 \leq \mathrm{y} \leq \mathrm{N}-1$$
$$0 \leq \mathrm{f(x,y)} \leq \mathrm{G}-1$$

dimana : M = jumlah piksel baris (*row*) pada *array* citra

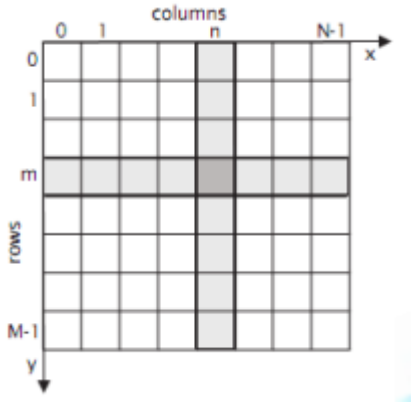
N = jumlah piksel kolom (*column*) pada *array* citra

G = nilai skala keabuan (*graylevel*)

Besarnya nilai ***M***, ***N*** dan ***G*** pada umumnya merupakan perpangkatan dari dua.

$$M = 2^m \; ; \; N = 2^n \; ; G = \; 2^m$$

Nilai **m**, **n**, dan **k** merupakan bilangan bulat positif. Rentang (0,G) disebut sebagai skala keabuan (*grayscale*). Besarnya nilai G bergantung pada proses digitalisasi yang dilakukan. Umumnya, pada skala keabuan, nilai 0 (nol) merepresentasikan intensitas warna hitam, sedangkan nilai 1 (satu) merepresentasikan intensitas warna putih. Pada citra dengan kedalaman bit sebanyak 8 bit, nilai G setara dengan 28, yaitu 256 warna atau derajat keabuan yang berbeda.

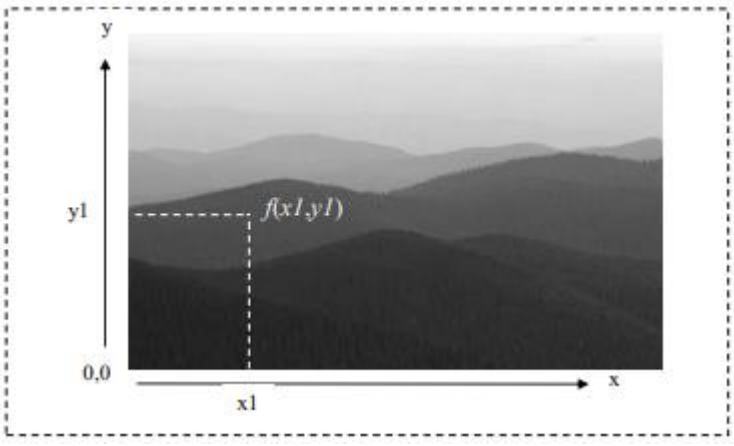


Gambar 2. 1 Representasi Citra Digital Dalam 2 Dimensi [19]

Dengan menggunakan pengolahan citra digital, objek tertentu dapat terdeteksi. Salah satu pendekatan yang digunakan adalah segmentasi warna, dimana metode normalisasi RGB adalah salah satunya. Metode ini memiliki keunggulan dalam kemudahan, kecepatan proses, dan efektivitasnya dalam mendeteksi objek seperti rambu lalu lintas [20], serta aplikasi deteksi wajah [21]. Beberapa teknik yang relevan dalam penelitian ini :

2.2.1.Dasar-Dasar Pengolahan Citra

Citra digital direpresentasikan oleh fungsi $f(x,y)$, di mana x dan y adalah koordinat baris dan kolom, dan f adalah nilai keabuan dari citra di titik (x,y). Kecerahan setiap piksel dalam citra disimpan sebagai nomor. Semakin tinggi nomor piksel, semakin gelap piksel tersebut, dan sebaliknya, semakin rendah nilai piksel, semakin terang. Umumnya, sistem menggunakan 256 tingkat kecerahan untuk setiap piksel, di mana nilai maksimum adalah 255 (gelap) dan nilai minimum adalah 0 (terang) [20].



Gambar 2. 2 Cita Digital [22]

a. Gambar Grayscale

Citra yang terdiri dari satu *layer* warna dengan derajat keabuan tertentu dapat dijelaskan melalui suatu fungsi :

$$\boldsymbol{f(x,y)\Sigma|0 \dots 255|} \quad (2.2)$$

b. Gambar Biner

Citra yang hanya memiliki dua nilai, yaitu 1 dan 0, dapat dinyatakan dalam suatu fungsi :

$$\boldsymbol{f(x,y)\Sigma\{0,1\}} \quad (2.3)$$

c. Gambar Berwarna

Citra digital terdiri dari tiga lapisan warna, yang disebut sebagai *RGB* (*Red-Green-Blue*). Lapisan R (*Red*) merupakan matriks yang menggambarkan intensitas kecerahan warna merah, lapisan G (*Green*) adalah matriks yang menunjukkan tingkat kecerahan warna hijau, dan lapisan B (*Blue*) adalah matriks yang mengekspresikan tingkat kecerahan warna biru. Representasi dari citra digital ini dijelaskan dengan persamaan:

$$\boldsymbol{f_R(x,y)\Sigma|0 \dots 255|} \quad (2.4)$$
$$\boldsymbol{f_G(x,y)\Sigma|0 \dots 255|}$$
$$\boldsymbol{f_B(x,y)\Sigma|0 \dots 255|}$$

Proses pengolahan citra digital menggunakan komputer dimulai dengan mentransformasikan citra ke dalam bentuk nilai-nilai diskrit dari tingkat keabuan pada titik-titik elemen citra. Citra dalam bentuk ini disebut sebagai citra digital. Elemen-elemen citra digital, ketika ditampilkan pada layar monitor, akan muncul sebagai piksel (*picture element/pixel*) yang mengisi ruang tampilan.

2.2.2. Model Citra Digital

Terdapat dua jenis citra: citra kontinu dan citra diskrit (citra digital). Citra kontinu dihasilkan oleh sistem optik yang menerima sinyal analog, seperti mata manusia dan kamera analog. Sementara itu, citra diskrit dihasilkan melalui proses digitalisasi, yang menghasilkan citra digital, seperti kamera digital dan scanner [19]. Citra adalah fungsi kontinu dari intensitas cahaya pada bidang dua dimensi, yang diungkapkan dalam bentuk fungsi:

$$\boldsymbol{f(x,y) = \mathbf{i}(\mathbf{x},\mathbf{y}).\mathbf{r}(\mathbf{x},\mathbf{y})} \quad (2.5)$$

Dimana :

$\boldsymbol{i}(\mathbf{x},\mathbf{y})$: $illumination$, besarnya $0 < \mathrm{i}(\mathrm{x},\mathrm{y}) < \infty$

$\boldsymbol{r}(\mathbf{x},\mathbf{y})$: $reflectance$, besarnya $0 < \mathrm{r}(\mathrm{x},\mathrm{y}) < 1$

Nilai i(x, y) ditentukan oleh sumber cahaya, misalnya:

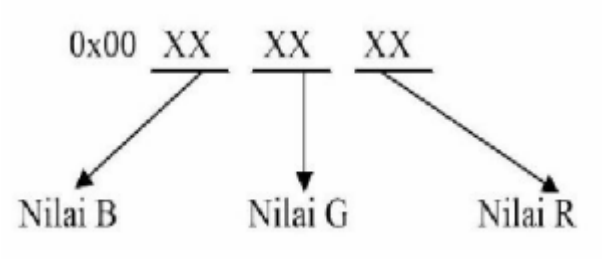
1. Pada hari cerah, matahari menghasilkan i(x, y) ≈ 9000 *foot candles*
2. Pada hari mendung, matahari menghasilkan i(x, y) ≈ 1000 *foot candles*
3. Pada malam bulan purnama, sinar bulan menghasilkan i(x, y) ≈ 0.01 *foot candles*

Nilai r(x, y) ditentukan oleh karakteristik objek dalam gambar, misalnya:

1. Benda hitam memiliki r(x, y) = 0.01
2. Dinding putih memiliki r(x, y) = 0.8
3. Benda logam stainless steel memiliki r(x, y) = 0.65

2.2.3. Model Warna RGB

Pengolahan citra didasarkan pada manipulasi warna *RGB* pada lokasi tertentu. Dalam pengolahan citra, warna direpresentasikan dalam nilai heksadesimal dari 0x00000000 hingga 0x00ffffff. Warna hitam diwakili oleh 0x00000000, sementara warna putih diwakili oleh 0x00ffffff. Definisi nilai delapan warna tersebut dapat dilihat pada Gambar 2.3, di mana variabel 0x00 menunjukkan bahwa angka di belakangnya adalah dalam format heksadesimal.



Gambar 2. 3 2 Nilai warna RGB dalam heksadesimal [22]

Pengkodean warna dalam model RGB dapat diwakili dalam angka heksadesimal (basis 16) untuk setiap komponen R, G, atau B. Misalnya:

1. Hitam murni direpresentasikan oleh kode #000000 (R=00, G=00, B=00)
2. Putih sempurna direpresentasikan oleh kode #FFFFFF (R=FF, G=FF, B=FF)
3. Biru murni direpresentasikan oleh kode #0000FF (R=00, G=00, B=FF)

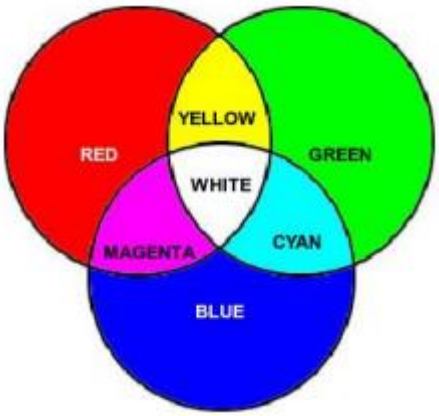
Model warna *RGB* merupakan model warna aditif di mana warna merah (*red*), hijau (*green*), dan biru (*blue*) ditambahkan bersama dengan berbagai cara untuk menciptakan beragam warna. Model warna aditif ini digunakan dalam pencahayaan, video, dan monitor. Sebagai contoh, monitor menciptakan warna dengan memancarkan cahaya melalui fosfor merah, hijau, dan biru.

Tujuan utama dari model warna *RGB* adalah untuk mereproduksi dan menampilkan gambar dalam sistem elektronik, seperti televisi dan komputer. Model warna *RGB* juga digunakan dalam fotografi konvensional.

RGB adalah ruang warna yang bergantung pada perangkat. Setiap perangkat dapat mendeteksi atau mereproduksi nilai *RGB* dengan cara yang berbeda. Untuk menciptakan warna dengan *RGB*, tiga sumber cahaya warna (merah, hijau, dan biru) harus ditumpangkan (misalnya, dengan emisi dari layar hitam atau dengan refleksi dari layar putih). Setiap sumber cahaya tersebut disebut sebagai komponen warna, dan masing-masing dapat memiliki intensitas yang berbeda.

RGB sering digunakan dalam perangkat input seperti TV berwarna, kamera *video*, scanner, dan kamera digital. Sedangkan perangkat output seperti berbagai jenis TV (CRT, LCD, plasma, dll), komputer dan layar HP, proyektor video, layar LED multiwarna, dan layar lebar seperti JumboTron menggunakan model warna *RGB*.

RGB juga umum digunakan dalam desain web. Awalnya, keterbatasan dalam kedalaman warna perangkat keras video membatasi palet warna menjadi 216 warna *RGB*, yang ditetapkan oleh *Netscape Color Cube*. Namun, dengan keunggulan menampilkan 24-bit, penggunaan penuh dari 16,7 juta warna dalam kode warna *RGB HTML* tidak lagi menjadi masalah bagi sebagian besar pengunjung situs web.

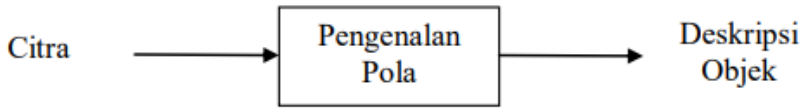


Gambar 2. 4 Model warna RGB [22]

2.2.4. Pengenalan Pola (Pattern Recognition)

Pengenalan pola (*Pattern Recognition*) adalah ilmu yang bertujuan untuk mengklasifikasikan atau menggambarkan sesuatu berdasarkan pengukuran kuantitatif fitur atau sifat utama dari suatu objek. Proses pengenalan pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin, seperti komputer. Tujuannya adalah untuk mengidentifikasi suatu objek dalam citra.

Manusia dapat mengenali objek yang dilihatnya karena otak manusia telah belajar mengklasifikasi objek-objek di alam sehingga mampu membedakan satu objek dengan yang lainnya. Kemampuan sistem visual manusia ini dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, kemudian memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek di dalam citra.



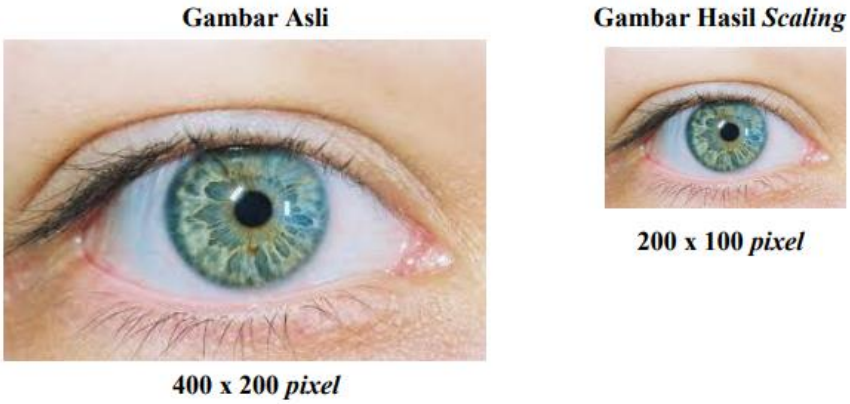
Gambar 2. 5 Pengenalan Pola [22]

Sebagai contoh, dalam pengenalan pola, citra mata seperti yang ditunjukkan dalam Gambar 2.6 dapat digunakan sebagai data masukan untuk mengenali mata. Dengan menggunakan algoritma pengenalan pola yang sesuai, diharapkan komputer dapat mengenali bahwa citra tersebut menggambarkan mata.

Gambar 2. 6 Citra Mata Yang Digunakan Untuk Pengenalan Mata [22]

2.2.5. Penskalaan Citra (Scaling)

Penskalaan citra (*scaling*) merupakan suatu operasi geometri yang menghasilkan efek memperbesar atau memperkecil ukuran citra input sesuai dengan variabel penskalaan citranya.



Gambar 2. 7 Penskalaan Citra [22]

Penskalaan citra digunakan untuk memperbesar (*zoom-in*) atau memperkecil (*zoom-out*) citra. Adapun salah satu metode untuk teknik penskalaan citra yaitu interpolasi bilinear:

2.2.5.1. Interpolasi Billinear

Interpolasi bilinear menentukan nilai piksel baru dengan menggabungkan rata-rata berbobot dari empat piksel tetangga terdekat dalam ukuran 2x2 pada gambar asli. Dalam interpolasi bilinear, fungsi dasarnya adalah garis linear, di mana setiap keluaran piksel dihitung dari kombinasi linear keempat piksel input. Metode ini menetapkan nilai piksel baru berdasarkan rata-rata berbobot dari empat piksel dalam matriks 2x2 di citra asli. Piksel yang paling dekat dengan piksel baru memiliki bobot lebih besar dibandingkan dengan piksel yang lebih jauh [22] . Proses *scaling* dengan metode interpolasi bilinear melibatkan

perhitungan nilai piksel baru pada gambar hasil scaling berdasarkan nilai piksel dari empat titik terdekat pada gambar asal. Rumus untuk interpolasi bilinear adalah:

$$P = (1-x)(1-y)P1 + x(1-y)P2 + (1-x)yP3 + xyP4 \quad (2.6)$$

Di mana:

- P adalah nilai piksel pada posisi baru
- x dan y adalah jarak relatif terhadap piksel terdekat pada gambar asal
- P1, P2, P3, P4 adalah nilai piksel dari 4 titik terdekat pada gambar asal

2.2.6. Konversi Warna *Grayscale*

Konversi warna merupakan proses mengubah representasi warna citra dari satu ruang warna ke ruang warna lainnya. Misalnya, mengonversi citra dari ruang warna *RGB* (*Red, Green, Blue*) ke *grayscale*. Citra digital *black and white* (*grayscale*) setiap pikselnya mempunyai warna gradasi mulai dari putih sampai hitam. Rentang tersebut berarti bahwa setiap piksel dapat diwakili oleh 8 bit, atau 1 byte. Rentang warna pada *black and white* sangat cocok digunakan untuk pengolahan file gambar. Salah satu bentuk fungsinya digunakan dalam kedokteran (*X-ray*). Pengubahan dari citra berwarna ke citra grayscale mengikuti aturan sebagai berikut:

$$\boldsymbol{Grayscale = 0,2989\,R + 0,5870\,G + 0,1140\,B} \quad (2.7)$$

Contoh gambar hasil *grayscale* :

Gambar 2. 8 Citra Grayscale [22]

2.3. Normalisasi Min-Max

Normalisasi Min-Max adalah sebuah metode normalisasi menggunakan strategi linier yang mengubah data dari satu rentang nilai ke rentang nilai yang baru. Hal ini bertujuan untuk menghasilkan keseimbangan nilai perbandingan antar data sebelum dan sesudah proses normalisasi. Data tersebut diubah menjadi nilai-nilai yang seimbang antara 0 hingga 1. Metode *Min-Max Normalization* merupakan salah satu teknik untuk mengubah data kompleks tanpa menghilangkan informasi yang terkandung di dalamnya. Berikut persamaan *Min-max Normalization* :

$$x' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{2.8}$$

Keterangan:

xi = nilai tertentu yang akan dinormalisasi

x’ = nilai hasil normalisasi

min(x) = nilai minimal dari sebuah atribut

max(x) = nilai maksimal dari sebuah atribut

Metode normalisasi *Min-max Normalization* melibatkan transformasi linier pada data asli untuk mencapai keseimbangan nilai perbandingan antar data sebelum dan sesudah proses normalisasi [23].

2.4. Computer Vision

Terminologi lain yang erat kaitannya dengan pengolahan citra digital adalah *computer vision* atau *machine vision*. Pada dasarnya, *computer vision* berusaha meniru cara kerja visual manusia. Manusia melihat objek melalui indera penglihatan (mata), kemudian citra objek tersebut diteruskan ke otak untuk diinterpretasi, sehingga manusia dapat memahami objek yang terlihat. Interpretasi ini dapat digunakan untuk pengambilan Keputusan [24].

Sebagaimana mata dan otak manusia, *computer vision* adalah sistem yang memiliki kemampuan untuk menganalisis objek secara visual setelah data objek tersebut dimasukkan dalam bentuk citra.

Proses-proses dalam *computer vision* dapat dibagi menjadi 3 aktivitas utama:

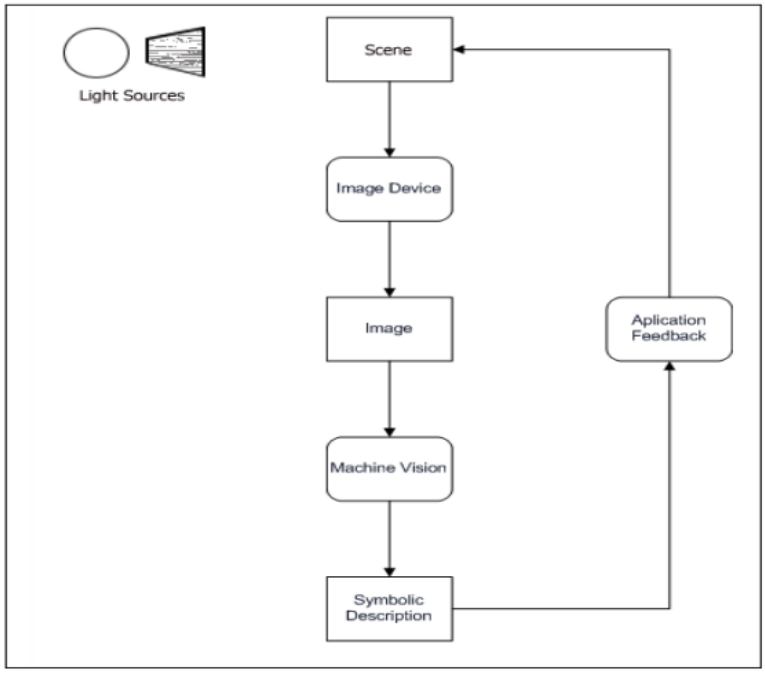
1. Memperoleh atau mengakuisisi citra digital.

2. Melakukan teknik komputasi untuk memproses atau memodifikasi data citra.
3. Menganalisis dan menginterpretasi citra menggunakan hasil pemrosesan untuk tujuan tertentu, seperti memandu robot, mengontrol peralatan, memantau manufaktur, dan lain-lain.

Pengolahan citra merupakan tahap awal (*preprocessing*) dalam *computer vision*, sedangkan pengenalan pola merupakan proses untuk menginterpretasi citra. Teknik-teknik dalam pengenalan pola memiliki peran penting dalam *computer vision* untuk mengenali objek.

2.4.1. Elemen-Elemen Computer Vision

Gambar 2.10 di bawah ini adalah struktur yang mendasari elemen-elemen suatu mesin vision.



Gambar 2. 9 Struktur Computer Vision secara umum [27]

Keterangan untuk gambar di atas adalah sebagai berikut:

a. *Light sources*: Merupakan sumber cahaya yang digunakan untuk aplikasi seperti layar laser, sistem robotika, dan sebagainya.
b. *Scene*: Merupakan kumpulan objek yang ada dalam lingkungan atau gambar.
c. *Image Device*: Merupakan alat yang digunakan untuk mengubah gambar menjadi sesuatu yang dapat dimengerti oleh mesin, seperti kamera atau sensor citra.

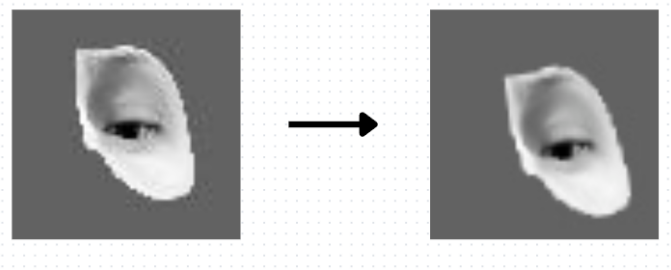
d. *Image*: Merupakan gambar dari suatu objek yang merupakan representasi dari keadaan sesungguhnya.
e. *Machine vision*: Merupakan mesin yang menginterpretasikan gambar terkait dengan ciri-ciri pola atau objek yang dapat ditelusuri oleh sistem.
f. *Symbolic description*: Merupakan sistem yang dapat digunakan untuk mengonversi struktur kerja sistem ke simbol-simbol tertentu yang dapat dimengerti oleh sistem.
g. *Application feedback*: Merupakan kondisi yang dapat memberikan respon untuk menerima gambar dari suatu sistem penglihatan, seperti umpan balik untuk memperbaiki atau memperbaiki proses pengolahan gambar

2.5. Augmentasi

Augmentasi data adalah teknik dalam pemrosesan gambar yang digunakan untuk memperluas dataset pelatihan dengan membuat variasi pada gambar-gambar yang ada. Data augmentasi mencakup berbagai teknik yang meningkatkan ukuran dan kualitas dataset pelatihan sehingga memungkinkan pembuatan model *deep learning* yang lebih baik [25]. Dalam domain pengolahan citra dan visi komputer, augmentasi data sangat penting karena data yang cukup dan beragam diperlukan untuk melatih model pembelajaran mesin yang efektif. *Overfitting* dapat dicegah melalui augmentasi data dengan memperkenalkan variasi pada data pelatihan, sehingga model dapat lebih generalisasi terhadap data yang belum pernah dilihat sebelumnya

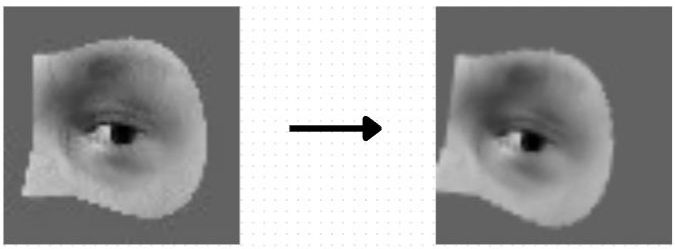
Cara kerja augmentasi data dilakukan dengan menerapkan berbagai transformasi pada data yang sudah ada untuk menghasilkan sampel baru. Beberapa metode augmentasi yang akan digunakan pada penelitian ini antara lain:

- Rotasi: Gambar dirotasi beberapa derajat untuk menciptakan variasi baru.



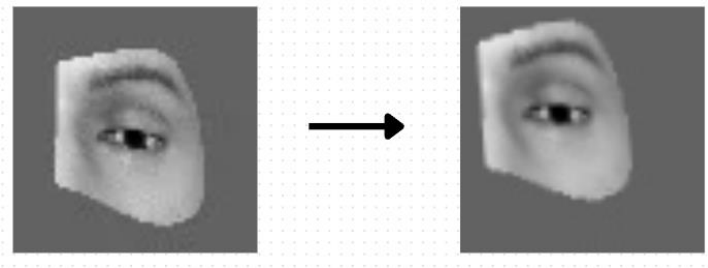
Gambar 2. 10 Augmentasi Rotasi

- Width Shift: Perpindahan gambar secara horizontal.



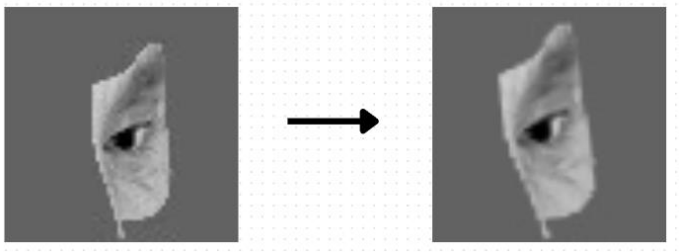
Gambar 2. 11 Augmentasi Weight Shift

- Height Shift: Perpindahan gambar secara vertikal.



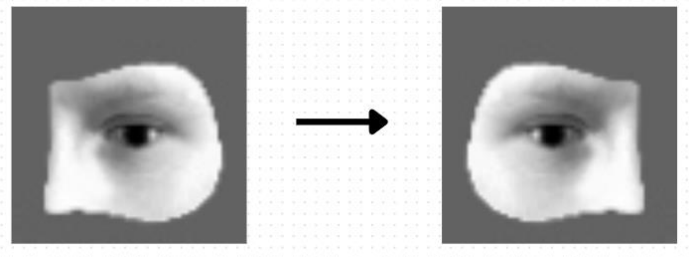
Gambar 2. 12 Augmentasi Height Shift

- Zoom Range: Penskalaan gambar untuk memperbesar atau memperkecil.



Gambar 2. 13 Augmentasi Zoom Range

- Horizontal Flip: Pencerminan gambar secara horizontal.



Gambar 2. 14 Augmentasi Horizontal Flip

Misalnya, dalam pengolahan citra, sebuah gambar dapat dirotasi beberapa derajat, dicerminkan secara horizontal atau vertikal, atau diterapkan perubahan kecerahan untuk menciptakan variasi baru dari gambar asli. Teknik-teknik ini tidak hanya meningkatkan

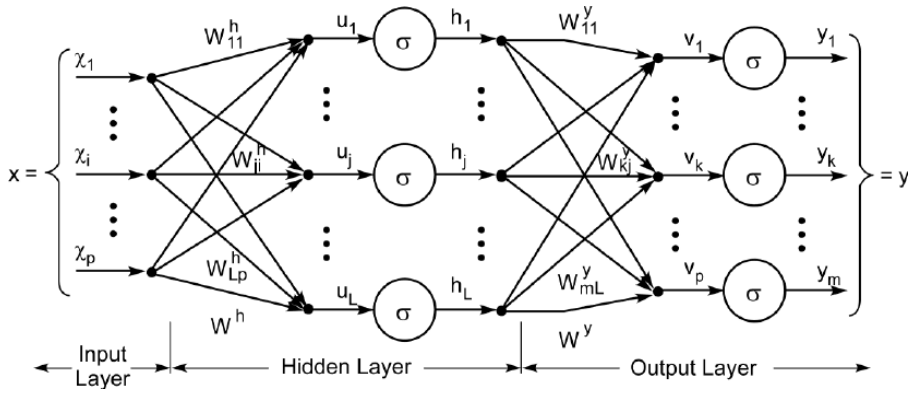
jumlah data pelatihan tetapi juga membantu model untuk menjadi lebih robust terhadap perubahan dan variasi dalam data input.

2.6. Jaringan Syaraf Tiruan (*Neural Network*)

Neural Network, atau yang dikenal sebagai Jaringan Syaraf Tiruan (JST), merupakan salah satu metode pembelajaran yang sering digunakan dalam memecahkan berbagai jenis masalah, baik yang bersifat diskrit, real, maupun vektor. JST juga merupakan representasi model dari sistem syaraf manusia dalam menyelesaikan berbagai tugas tertentu [8]. Dalam proses pembelajaran JST, terdapat serangkaian kejadian yang terjadi sebagaimana berikut:

a. Jaringan Syaraf Tiruan (JST) merespons terhadap lingkungan sekitarnya.
b. Sebagai hasil dari respons terhadap rangsangan tersebut, JST mengalami perubahan dalam dirinya.
c. JST memberikan respons baru kepada lingkungan dengan cara yang berbeda karena adanya perubahan dalam struktur internalnya sendiri.

Proses pembelajaran dalam jaringan syaraf tiruan meliputi pembelajaran yang diawasi (*Supervised Learning*) dan pembelajaran tanpa pengawasan (*Unsupervised Learning*). Pembelajaran yang diawasi merupakan proses pembelajaran yang memerlukan pengetahuan tentang lingkungan yang direpresentasikan oleh kumpulan contoh input-output. Salah satu contoh model JST yang termasuk dalam pembelajaran yang diawasi adalah JST dengan arsitektur *Multi Layer Perceptron* (MLP), yang sering digunakan dalam konteks pendidikan.



Gambar 2. 15 Model Multilayer Perceptron [8]

MLP adalah model JST yang paling umum digunakan dalam bidang pendidikan dan aplikasi lainnya. Keunggulan arsitektur dan proses pembelajaran yang sederhana membuatnya mudah dipelajari. Ilustrasi arsitektur MLP dapat dilihat pada gambar berikut ini.

Keteranagn :

Neuron-neuron :	Bobot:
Xi = input (i = 1, 2, .., n)	V0j = bias unit hidden
Zj = neural hidden (j = 1, 2, ..., p)	Vij = bobot unit hidden
Yk = output (k= 1, 2, ..., m)	W0k = bias unit output
	Wjk = bobot unit output

Secara umum proses pelatihan yang dilakukan adalah sebagai berikut :

1. Pengambilan input
2. Penelusuran error
3. Penyesuaian bobot

Neural network memiliki beberapa jenis, berikut adalah konsep neural network yang digunakan pada penelitian ini :

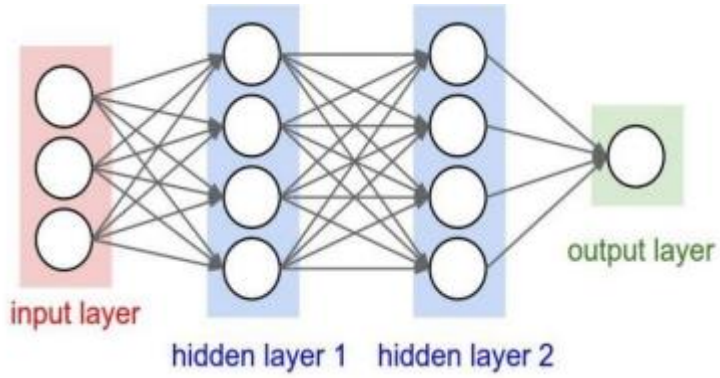
1. *Recurrent Neural Network* (*RNN*)

2.7. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik [26].

2.7.1. Konsep CNN

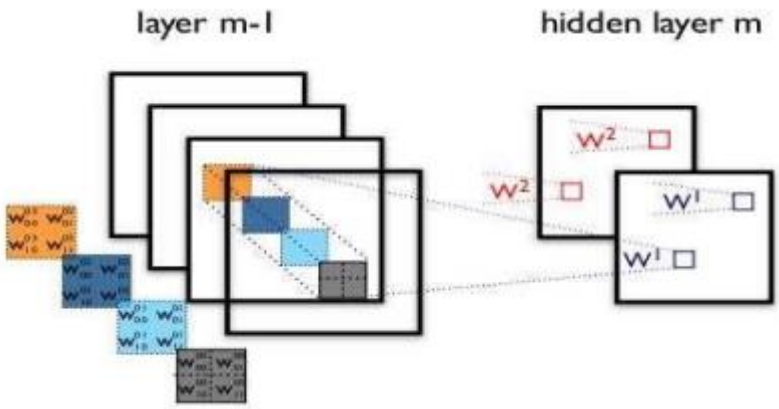
Cara kerja CNN mirip dengan MLP, namun perbedaannya terletak pada ukuran *neuron*. *Neuron* pada CNN memiliki dua dimensi, sedangkan *neuron* pada MLP hanya memiliki satu dimensi.



Gambar 2. 16 Arsitektur MLP Sederhana [26]

Pada Gambar 2.14, ditunjukkan bahwa sebuah MLP memiliki beberapa layer (dilambangkan dengan kotak merah dan biru) dengan setiap layer terdiri dari sejumlah neuron (ditandai dengan lingkaran putih). MLP menerima dan mempropagasi data satu dimensi melalui jaringan hingga menghasilkan output tertentu. Hubungan antar neuron pada dua *layer* yang berdekatan diatur oleh parameter bobot yang menentukan kualitas model. Setiap input data dioperasikan secara linier menggunakan parameter bobot tersebut, dan hasil perhitungannya diubah dengan menggunakan fungsi aktivasi nonlinier.

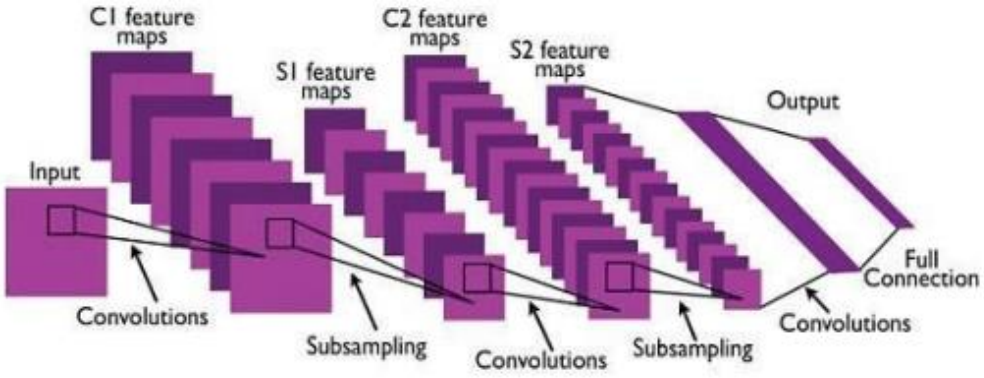
Sebaliknya, pada CNN, data yang dipropagasi dalam jaringan berbentuk dua dimensi, menyebabkan operasi linier dan parameter bobot pada CNN berbeda. Di CNN, operasi linier melibatkan operasi konvolusi, dan parameter bobot berbentuk empat dimensi yang terdiri dari kumpulan kernel konvolusi, seperti yang ditunjukkan pada Gambar 2. Rumus dimensi bobot pada CNN adalah: *neuron input* x *neuron output* x tinggi x lebar. CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara karena sifat proses konvolusi tersebut.



Gambar 2. 17 Proses Konvolusi pada CNN [26]

2.7.2. Arsitektur CNN

Jaringan Saraf Tiruan (JST) terdiri dari beberapa layer dan neuron di setiap layernya, tanpa aturan baku yang menentukan jumlahnya. Dalam MLP, jaringan tanpa *hidden layer* dapat memetakan semua persamaan linear. Jaringan dengan satu atau dua *hidden layer* mampu memetakan sebagian besar persamaan pada data sederhana. Namun, untuk data yang lebih kompleks, penggunaan lebih dari dua layer dapat menyebabkan *overfitting* dan mengurangi efektivitas *backpropagation* secara signifikan. Oleh karena itu, penggunaan lebih dari dua *layer* tidak direkomendasikan.



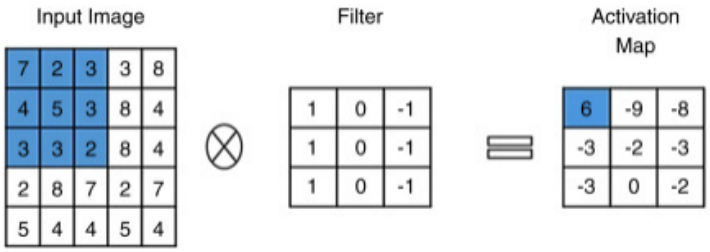
Gambar 2. 18 Arsitektur Convolutional Neural Network [29]

Seiring dengan perkembangan *deep learning*, ditemukan cara untuk mengatasi keterbatasan MLP dalam mengolah data kompleks, yaitu dengan menggunakan fungsi yang mengubah data input menjadi bentuk yang lebih sederhana untuk MLP. Hal ini menginspirasi ide untuk menambahkan beberapa layer pada model, yang bertugas mengubah data sebelum diproses oleh metode klasifikasi, sehingga berkembanglah model neural network dengan lebih dari tiga layer. Karena layer awal berfungsi sebagai metode

ekstraksi fitur, jumlah layer dalam *Deep Neural Networks* (DNN) tidak memiliki aturan baku dan bervariasi tergantung pada *dataset* yang digunakan. Oleh karena itu, setiap layer dalam jaringan dan jumlah neuron di setiap layer dikategorikan sebagai *hyperparameter* yang perlu dioptimasi melalui proses pencarian [26].

2.7.3. Convolution Layer

Lapisan konvolusional adalah blok bangunan utama dari CNN. Lapisan ini terdiri dari sekumpulan filter (atau kernel) yang parameternya harus dipelajari selama pelatihan. Ukuran filter biasanya lebih kecil dari ukuran gambar sebenarnya. Setiap filter akan menyapu gambar dan menghasilkan peta aktivasi. Filter konvolusi bergerak melintasi tinggi dan lebar gambar, dan pada setiap posisi spasial, dilakukan perkalian titik antara setiap elemen filter dan masukan. Gambar 2.21 menunjukkan contoh proses konvolusi. Entri pertama dari peta aktivasi (ditandai dengan warna biru pada Gambar 2.21) dihitung dengan menggabungkan filter dengan bagian yang ditandai biru pada gambar *input*. Peta aktivasi dihasilkan dengan mengulangi proses ini untuk setiap elemen gambar input. Volume *output* dari lapisan konvolusi dibuat dengan menumpuk peta aktivasi setiap filter sepanjang dimensi kedalaman. Setiap komponen dari peta aktivasi dapat dianggap sebagai output dari sebuah neuron, yang terhubung ke wilayah lokal kecil pada gambar input dengan ukuran yang sama dengan ukuran filter. Semua *neuron* dalam peta aktivasi berbagi parameter satu sama lain. Karena konektivitas lokal dari lapisan konvolusional, jaringan dipaksa untuk mempelajari filter yang memberikan respons maksimum terhadap wilayah input lokal. Lapisan konvolusional awal menangkap fitur tingkat rendah seperti garis pada gambar, sedangkan lapisan-lapisan berikutnya mengekstraksi fitur tingkat tinggi seperti bentuk dan objek tertentu [27].



Gambar 2. 19 Operasi Konvolusi [27]

Salah satu komponen kunci dalam CNN adalah fungsi aktivasi, yang menentukan bagaimana neuron diaktifkan setelah operasi konvolusi dan *pooling*. Fungsi aktivasi yang umum digunakan adalah *Rectified Linear Unit* (ReLU).

2.7.3.1. Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) adalah fungsi aktivasi yang umum digunakan dalam jaringan saraf, terutama dalam *Convolutional Neural Networks* (CNN). ReLU membantu dalam mempertahankan fitur penting dan meningkatkan sparsity, sementara *pooling* mengurangi dimensi data dan membantu mengontrol [28]. Fungsi dan Kegunaan ReLU antara lain :

1. Memperkenalkan Non-Linearitas:

   Dalam jaringan saraf, lapisan-lapisan konvolusi pada dasarnya melakukan operasi linier pada data input. Fungsi aktivasi seperti *ReLU* diperkenalkan untuk memperkenalkan non-linearitas ke dalam model, yang memungkinkan jaringan untuk belajar dan merepresentasikan hubungan yang kompleks dalam data. Tanpa fungsi aktivasi, jaringan saraf akan berperilaku seperti model linier, yang memiliki keterbatasan dalam menyelesaikan masalah yang kompleks.

2. Mengatasi Masalah *Vanishing Gradient*:

   Fungsi aktivasi sebelumnya seperti sigmoid dan tanh sering mengalami masalah vanishing gradient, di mana gradien cenderung mengecil selama *backpropagation*, menyebabkan pelatihan model menjadi sangat lambat atau bahkan terhenti. *ReLU* membantu mengatasi masalah ini karena gradiennya konstan untuk nilai input yang positif, sehingga mempercepat proses pelatihan jaringan saraf.

3. *Sparsity* (Kebanyakan Neuron Tidak Aktif):

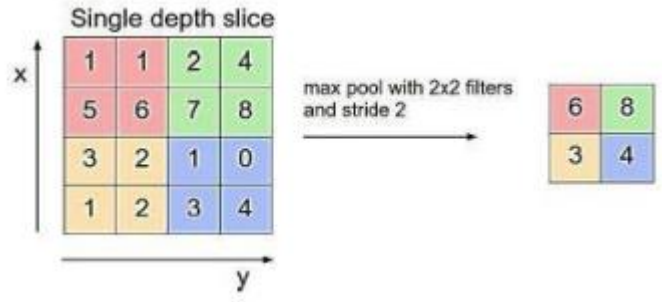
   ReLU menghasilkan sparsity secara alami, karena banyak neuron yang akan memiliki output nol. Sparsity ini dapat meningkatkan efisiensi komputasi dan mengurangi risiko *overfitting*, karena hanya sebagian kecil dari neuron yang aktif pada waktu tertentu.

Dalam penelitian ini, penggunaan fungsi aktivasi *ReLU* pada lapisan *pooling* di *CNN* diharapkan dapat meningkatkan efisiensi dan kinerja model dalam tugas klasifikasi arah bola mata menggunakan *IndRNN*. Kombinasi ini memungkinkan jaringan untuk belajar representasi fitur yang lebih kuat dan lebih *robust* dari data input.

2.7.4. Pooling Layer

Tujuan utama dari *Pooling Layer* adalah untuk mengurangi jumlah parameter dari tensor input, sehingga membantu mengurangi *overfitting*, mengekstrak fitur representatif

dari tensor input, dan mengurangi perhitungan, sehingga meningkatkan efisiensi. Ada dua jenis *Pooling*, yaitu *Max Pooling* dan *Average Pooling*. Seperti yang ditunjukkan pada Gambar 2.18, dalam *Max Pooling*, sebuah kernel berukuran n * n (misalnya 2x2) dipindahkan melintasi matriks, dan untuk setiap posisi, nilai maksimum diambil dan dimasukkan ke dalam posisi yang sesuai pada matriks keluaran. Sedangkan dalam Average Pooling, kernel dengan ukuran n * n dipindahkan melintasi matriks, dan untuk setiap posisi, rata-rata dari semua nilai diambil dan dimasukkan ke dalam posisi yang sesuai pada matriks keluaran [29].



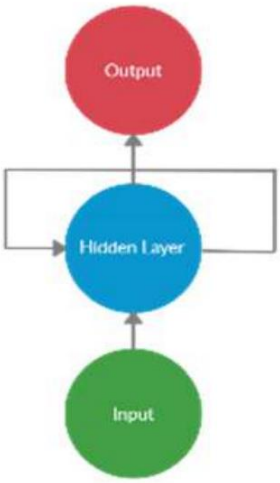
Gambar 2. 20 Operasi Max Pooling [29]

2.7.5. Flatten Layer

Lapisan *flatten* dalam jaringan saraf konvolusional (*CNN*) meruntuhkan dimensi spasial dari fitur yang dikumpulkan, namun mempertahankan dimensi saluran. Proses ini mengubah representasi matriks data menjadi vektor satu dimensi. Vektor ini kemudian diteruskan ke lapisan *fully connected layer*, yang juga dikenal sebagai *Dense Layer*. Dalam lapisan ini, setiap neuron terhubung ke setiap neuron di lapisan sebelumnya, menciptakan jaringan saraf yang sepenuhnya terhubung.

2.8. RNN

RNN termasuk dalam jenis jaringan pemodelan *Neural Network*. *RNN* yang juga disebut jaringan umpan balik adalah jenis jaringan pada *neural network* dimana terdapat *loop* sebagai koneksi umpan balik dalam jaringan. [30] Jaringan *RNN* adalah jaringan yang mengakomodasi *output* jaringan untuk menjadi input pada jaringan tersebut yang kemudian digunakan untuk menghasilkan output yang baru.

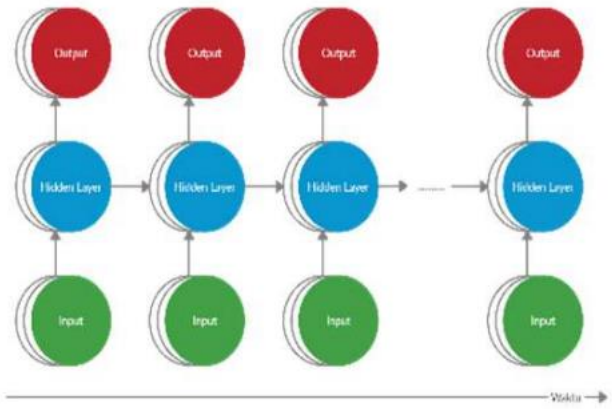


Gambar 2. 21 Arsitektur RNN [28]

RNN dirancang khusus untuk data yang memiliki urutan atau pola bertahap. Berbeda dengan penggunaan *neural network* lainnya di mana input dan *output* tidak saling terkait, hal ini dapat menyebabkan penumpukan tugas yang berlebihan dan kompleksitas yang tinggi. Namun, *RNN* cocok digunakan dalam aplikasi yang memproses data ilmiah dengan pola waktu (*time series*) karena *RNN* mampu memproses setiap elemen data dalam urutan tertentu, kemudian menggunakan hasil komputasi sebelumnya untuk menghasilkan *output* baru. Dengan demikian, *RNN* dapat memanfaatkan informasi yang telah direkam sebelumnya dalam berbagai panjang urutan.

Arsitektur *RNN* terdiri dari node yang disusun dalam lapisan-lapisan berurutan. Setiap node dalam suatu lapisan terhubung dengan setiap node dalam lapisan berikutnya, dengan setiap node memiliki aktivasi yang bervariasi tergantung pada waktu.

Pada diagram arsitektur *RNN*, lingkaran hijau menunjukkan lapisan *input*, di mana data dimasukkan untuk diolah oleh *RNN*. Kemudian, data masuk ke lapisan tersembunyi (lingkaran biru) yang memiliki koneksi siklik. Pola siklik ini memungkinkan RNN untuk menyimpan informasi sementara yang akan digunakan dalam pemrosesan data berikutnya. Terakhir, *output layer* menghasilkan hasil dari proses komputasi *RNN*.

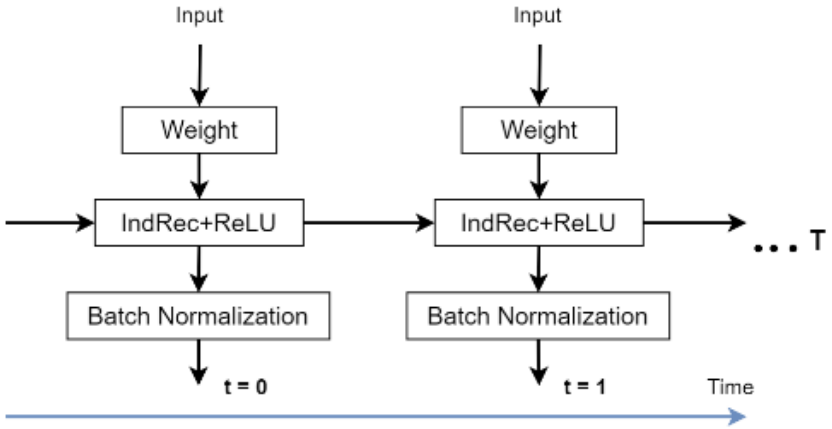


Gambar 2. 22 Arsitektur RNN Banyak Layer [28]

Arsitektur *RNN* menunjukkan bagaimana *RNN* bekerja dengan beberapa lapisan dan perulangan status tersembunyi sebanyak data waktu yang tersedia. Variabel x_t merupakan input, s_t *hidden layer*, dan O_t merupakan output. Pada *hidden layer*, memori dari kalkulasi disimpan menggunakan rumus yang diberikan. Fungsi f biasanya berupa fungsi non-linear seperti *tanh* atau *ReLu*. Nilai awal dari *hidden layer*, S_{t-1}, sering kali diinisialisasi sebagai 0, sedangkan u adalah vektor data input dan w adalah vektor data output pada *RNN*.

2.9. Independently Recurrent Neural Network (IndRNN)

IndRNN adalah varian dari *Recurrent Neural Network* (*RNN*) untuk mengatasi masalah *vanishing/exploding* gradient yang sering terjadi pada *RNN* konvensional. *IndRNN* menggunakan gerbang independen pada setiap waktu untuk menghindari masalah tersebut [10].



Gambar 2. 23 Arsitektur IndRNN [9]

Pada *IndRNN*, setiap neuron memiliki gerbangnya sendiri yang dikendalikan oleh aktivasinya sendiri pada waktu sebelumnya dan input pada waktu sekarang. Ini berbeda dengan *RNN* tradisional yang menggunakan aktivasi dari semua neuron pada waktu sebelumnya. Persamaan untuk *IndRNN* dapat dinyatakan sebagai berikut:

$$h\_t = x\_t * (W\_x * x\_t) + (1 - x\_t) * (W\_h * h\_(t-1)) \quad (2.9)$$

Dimana:

- h_t adalah state tersembunyi pada waktu t
- x_t adalah input pada waktu t
- W_x dan W_h adalah bobot untuk input dan state tersembunyi sebelumnya adalah operasi perkalian elemen-wise

Kelebihan utama *IndRNN* adalah kemampuannya untuk menghindari masalah *vanishing/exploding gradient*, sehingga dapat mempelajari dependensi jangka panjang dengan lebih baik dibandingkan *RNN* tradisional. Selain itu, *IndRNN* juga lebih cepat dan lebih efisien dalam pelatihan dan inferensi [10].

IndRNN telah diterapkan dengan sukses pada berbagai tugas seperti pemrosesan bahasa alami [31] , analisis sentimen [32] , dan penerjemahan mesin [33] . Selain itu, *IndRNN* juga telah digunakan untuk pendeteksian kantuk berdasarkan fitur kedipan mata [34] .

2.9.1. Data Input 3D

IndRNN, seperti *RNN* pada umumnya, memerlukan data input dalam bentuk tiga dimensi: dimensi sampel, dimensi waktu, dan dimensi fitur. Dimensi sampel mewakili jumlah contoh data yang diberikan, dimensi waktu mewakili panjang urutan input, dan dimensi fitur mewakili jumlah fitur atau variabel yang digunakan untuk merepresentasikan setiap langkah waktu dalam urutan [35].

Representasi data dalam bentuk tiga dimensi ini penting untuk pemrosesan data urutan seperti teks, suara, atau sinyal lainnya yang memiliki ketergantungan temporal. Setiap sampel data diwakili sebagai matriks 2D dengan dimensi waktu dan dimensi fitur, dan kumpulan sampel data ini membentuk tensor 3D yang menjadi input untuk IndRNN.

2.9.2. Normalisasi *Batch*

Normalisasi *batch* (*batch normalization*) adalah teknik yang digunakan dalam pelatihan jaringan saraf untuk mempercepat pelatihan dan meningkatkan kinerja model [36]. Pada dasarnya, normalisasi *batch* melakukan normalisasi input pada setiap lapisan dengan cara mengurangi input dengan rata-rata *batch* dan membaginya dengan standar deviasi *batch*.

Normalisasi *batch* membantu mempercepat pelatihan dengan menstabilkan distribusi input pada setiap lapisan, sehingga mengurangi internal *covariate shift* (pergeseran distribusi input pada lapisan dalam jaringan). Ini membuat pelatihan lebih stabil dan memungkinkan penggunaan learning rate yang lebih besar, yang dapat mempercepat konvergensi [36].

2.9.3. Dropout

Dropout adalah teknik regularisasi yang digunakan dalam pelatihan jaringan saraf untuk mencegah *overfitting* [37]. Pada dasarnya, dropout menonaktifkan secara acak sejumlah neuron dalam jaringan pada setiap iterasi pelatihan, memaksa jaringan untuk mempelajari representrepresentasi yang lebih *robust* dan terdistribusi dengan menonaktifkan sebagian neuron secara acak selama pelatihan. Pada saat pelatihan selesai, bobot jaringan akan diskalakan dengan faktor *dropout* untuk mempertahankan nilai harapan yang sama seperti saat pelatihan.

Secara matematis, pada tahap pelatihan, setiap unit output dari lapisan akan dimatikan (set ke nol) dengan probabilitas tertentu, misalnya 0,5. Sedangkan pada saat inferensi atau evaluasi menggunakan model yang telah dilatih, tidak ada *neuron* yang dimatikan. Namun, bobot yang masuk ke lapisan berikutnya akan diskalakan dengan faktor probabilitas *dropout* yang digunakan selama pelatihan.

2.9.4. Lapisan Output

Lapisan output dalam *IndRNN* ditentukan oleh jenis tugas atau masalah yang ingin diselesaikan. Untuk tugas klasifikasi multi-kelas, lapisan output yang umum digunakan adalah lapisan *Softmax* yang akan menghasilkan probabilitas kelas-kelas yang mungkin.

2.9.4.1. Fungsi Aktivasi Softmax

Softmax adalah fungsi aktivasi yang umum digunakan dalam lapisan output jaringan saraf untuk tugas klasifikasi multi-kelas. *Softmax* mengonversi skor mentah (*raw scores*) dari lapisan sebelumnya menjadi distribusi probabilitas yang jumlahnya satu, sehingga dapat diinterpretasikan sebagai probabilitas kelas-kelas yang mungkin.

Fungsi *softmax* memastikan bahwa jumlah probabilitas dari semua kelas adalah 1, dan setiap probabilitas kelas berada dalam rentang [0, 1]. Kelas dengan probabilitas tertinggi kemudian diprediksi sebagai output klasifikasi.

Softmax sering digunakan bersama dengan fungsi *loss cross-entropy* dalam tugas klasifikasi multi-kelas, karena kombinasi ini memungkinkan optimasi langsung dari probabilitas kelas yang diharapkan [38].

Secara spesifik, pada lapisan *Softmax*, skor untuk setiap kelas k akan dihitung terlebih dahulu dari lapisan sebelumnya. Kemudian, skor tersebut akan dinormalisasi menggunakan fungsi *Softmax* untuk menghasilkan probabilitas kelas, dimana jumlah probabilitas dari semua kelas adalah 1. Kelas dengan probabilitas tertinggi akan diprediksi sebagai *output*.

2.9.4.2. Optimasi Adam

Optimasi Adam merupakan varian dari algoritma optimasi *stochastic gradient descent* (*SGD*) yang dirancang khusus untuk mengatasi permasalahan yang sering terjadi saat menggunakan *SGD*, seperti masalah dari laju pembelajaran yang tidak seragam dan bising dari gradien stokastik.

Algoritma Adam bekerja dengan mempertimbangkan estimasi dari gradien saat ini dan gradien dari iterasi sebelumnya untuk mendapatkan perbaruan parameter yang lebih baik dalam konteks optimasi masalah *machine learning*.

Dalam algoritma Adam, perbaruan parameter (misalnya bobot-bobot dalam jaringan saraf) dilakukan dengan mengombinasikan antara estimasi gradien saat ini dan rata-rata eksponensial tertimbang dari gradien sebelumnya. Algoritma ini secara adaptif menyesuaikan tingkat perbaruan bobot berdasarkan informasi gradien yang dikumpulkan dari iterasi sebelumnya.

Pendekatan adaptif ini telah diakui dapat membantu akselerasi konvergensi dalam proses pelatihan model dengan menangani masalah seperti pengaktifan satuan jaringan saraf atau mengatasi suatu efek pelembatan pada proses pembelajaran (*vanishing/exploding gradients*) [39].

2.9.4.3. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (*SGD*) merupakan salah satu algoritma optimasi yang paling mendasar dan banyak digunakan dalam pelatihan model machine learning, khususnya jaringan saraf tiruan. *SGD* merupakan varian dari metode *Gradient Descent* (*GD*), yang dirancang untuk mempercepat proses pelatihan dengan menggunakan *subset* acak dari data pelatihan (*mini-batch*) pada setiap iterasi [40].

Gradient Descent bekerja dengan cara memperbarui parameter model (misalnya bobot-bobot dalam jaringan saraf) secara iteratif untuk meminimalkan fungsi *loss*. Pada setiap iterasi, gradien dari fungsi *loss* terhadap parameter dihitung, dan parameter diperbarui dengan cara bergerak ke arah yang berlawanan dengan gradien tersebut.

Pemilihan antara *SGD* dan Adam sangat tergantung pada jenis masalah yang dihadapi dan kebutuhan spesifik dari tugas pelatihan. Untuk masalah yang memerlukan konvergensi cepat, terutama di awal pelatihan, Adam mungkin lebih cocok. Namun, untuk tugas-tugas di mana kemampuan generalisasi sangat penting, SGD sering kali menjadi pilihan yang lebih baik [41].

2.10. K-Fold Cross Validation

K-Fold Cross Validation adalah metode validasi model yang sering digunakan dalam machine learning untuk menilai performa model prediksi. Teknik ini membagi dataset menjadi K bagian atau *'folds'* yang berukuran hampir sama. Dalam prosesnya, data training dibagi menjadi K subset, kemudian model dilatih K kali, di mana setiap kali menggunakan K-1 *subset* sebagai data training dan satu subset sebagai data testing [42]. Hasil dari setiap fold ini kemudian dirata-ratakan untuk memberikan estimasi kinerja model secara keseluruhan.

Metode *K-Fold Cross Validation* memiliki beberapa keuntungan utama. Pertama, teknik ini memastikan setiap observasi dalam dataset digunakan baik sebagai data training maupun data testing, sehingga dapat mengurangi bias dan memberikan estimasi kinerja model yang lebih stabil dan akurat. Kedua, teknik ini sangat berguna dalam situasi dengan data yang terbatas karena memungkinkan pemanfaatan data secara maksimal untuk validasi dan pelatihan.

2.11. Evaluasi dan Pengukuran Kinerja

Evaluasi dan pengukuran kinerja model merupakan aspek penting dalam siklus pengembangan machine learning. Langkah ini bertujuan untuk menilai seberapa baik model memprediksi output yang diharapkan dan memahami kelemahan serta kekuatan model tersebut. Dalam poin ini, akan dibahas beberapa metrik evaluasi utama dan alat yang digunakan untuk mengukur kinerja model, termasuk akurasi, confusion matrix, loss, dan classification report.

2.11.1. Akurasi

Akurasi merupakan metrik yang mengukur seberapa sering model membuat prediksi yang benar. Metrik ini sering digunakan karena kesederhanaannya dan kemampuannya untuk memberikan gambaran umum tentang kinerja model klasifikasi.

Dalam penelitian [43], dijelaskan bahwa meskipun akurasi memberikan gambaran umum tentang kinerja model, metrik ini dapat menyesatkan terutama pada dataset yang tidak seimbang . Oleh karena itu, metrik tambahan seperti *precision, recall, dan F1-score* sering digunakan untuk memberikan gambaran yang lebih komprehensif.

Kelebihan :

- Kesederhanaan: Mudah dipahami dan dihitung.
- Umum: Cocok digunakan sebagai metrik awal untuk mendapatkan gambaran umum tentang kinerja model.

Kekurangan:

Sensitivitas terhadap Kelas Tidak Seimbang: Pada *dataset* dengan kelas yang tidak seimbang, akurasi dapat memberikan gambaran yang menyesatkan. Misalnya, jika 95% data termasuk dalam satu kelas, model yang selalu memprediksi kelas mayoritas akan memiliki akurasi 95% meskipun kinerjanya buruk pada kelas minoritas.

2.11.2. Confussion Matrix

Confusion matrix merupakan alat yang sangat berguna untuk memahami kinerja model klasifikasi. Matriks ini menunjukkan jumlah prediksi benar dan salah yang dibuat oleh model, dibagi menurut kelas yang sebenarnya dan yang diprediksi [44].

Table 2 1 Struktur Confussion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Kelebihan:

- Detail dan Komprehensif: Menyediakan informasi detail tentang prediksi benar dan salah untuk setiap kelas.
- Banyak Metrik Turunan: Memungkinkan perhitungan berbagai metrik lain seperti *precision, recall, dan F1-score*.

Kekurangan:

- Kompleksitas: Mungkin membingungkan untuk dipahami tanpa penjelasan yang memadai.
- Ukuran Matriks: Untuk *dataset* dengan banyak kelas, confusion matrix dapat menjadi sangat besar dan sulit untuk dianalisis secara langsung.

2.11.3. Loss

Loss merupakan metrik yang digunakan untuk mengukur seberapa baik model beradaptasi dengan data pelatihan selama proses *training*. Fungsi *loss* membantu dalam mengarahkan proses training model dengan memberikan sinyal seberapa jauh prediksi model dari nilai yang sebenarnya. Jenis *Loss Function* yang digunakan adalah *Cross-Entropy Loss* :

2.11.3.1. Cross-Entropy Loss

Cross-Entropy Loss, juga dikenal sebagai *log loss*, adalah fungsi *loss* yang digunakan secara luas dalam masalah klasifikasi, baik untuk klasifikasi biner maupun multi-kelas. Fungsi ini mengukur performa model klasifikasi dengan menghitung perbedaan antara distribusi yang diprediksi oleh model dan distribusi target yang sebenarnya [35].

Kelebihan:

- Sensitif terhadap Kesalahan: *Cross-Entropy Loss* memberikan penalti yang lebih besar untuk prediksi yang salah jauh dari kebenaran, sehingga model didorong untuk membuat prediksi yang lebih akurat.
- Probabilistik: Fungsi ini bekerja baik dengan model yang memberikan *output* dalam bentuk probabilitas, seperti regresi logistik dan jaringan saraf.

Kekurangan:

- *Overfitting*: Pada dataset yang kecil atau dengan jumlah kelas yang besar, model bisa mudah mengalami *overfitting*, terutama jika tidak ada regularisasi yang memadai.
- Komputasi: Perhitungan logaritma bisa menjadi komputasi intensif untuk *dataset* yang sangat besar, meskipun ini umumnya tidak menjadi masalah dengan kemajuan teknologi komputasi saat ini.

2.11.4. Classification Report

Classification report merupakan ringkasan dari kinerja model klasifikasi, mencakup metrik seperti *precision, recall*, dan F1-*score* untuk setiap kelas. Laporan ini memberikan pandangan yang lebih komprehensif mengenai seberapa baik model berperformasi untuk setiap kelas yang berbeda.

Table 2 2 Contoh Tabel Classification Report

	Precision	Recall	F1-Score	Support
0	0.90	0.85	0.87	100
1	0.90	0.88	0.84	100
Avg/Total	0.85	0.86	0.86	200

1. Akurasi: Persentase prediksi yang benar dari total data uji.
2. Presisi: Persentase prediksi positif yang benar dari total prediksi positif.
3. *Recall*: Persentase prediksi positif yang benar dari total data positif sebenarnya.
4. *F1-Score*: Harmonic mean dari presisi dan *recall*.

Kelebihan:

- Detail: Menyediakan metrik kinerja untuk setiap kelas secara terpisah, memberikan pandangan yang lebih detail tentang kinerja model pada setiap kelas.
- Komprehensif: Menyediakan metrik tambahan seperti *support,* yang menunjukkan jumlah sampel dari setiap kelas.

Kekurangan:

- Kompleksitas: Metrik yang banyak bisa membingungkan untuk pengguna yang tidak berpengalaman.
- Tidak Langsung: Mungkin tidak memberikan pandangan langsung tentang kinerja keseluruhan model tanpa analisis tambahan.

2.12. Penelitian Terkait

Table 2 3 Penelitian Terkait

Judul	Penulis	Metode	Hasil
Detektor Kebohogan Dengan Analisis Pembesaran Diameter Pupil Dan Pergerakan Mata [4]	Reza Adriansyah Rusmanto	*Support Vector Machine, circular hough transform*	Akurasi sebesar 73% dengan analisis pergerakan bola mata menggunakan metode *circular hough transform.* SVM memiliki performa baik pada berbagai masalah klasifikasi. Kesederhanaan model membuatnya mudah diimplementasikan dan diinterpretasikan. Metode ini sangat tergantung pada kualitas gambar yang tinggi dan kesulitan dalam menangani variasi pergerakan iris yang kompleks.
Deteksi Pergerakan Mata untuk Mendeteksi Kebohongan Menggunakan *Circular Hough Transform* [3]	Asep Nana Hermana, Irma Amelia Dewi, Rhyaldi Yunus	*Circular Hough Transform*	Akurasi sebesar 52% dari pengujian pada 50 subjek anak-anak. Kelebihan dari metode ini adalah kesederhanaannya dan kemudahan dalam implementasi. Namun, kekurangannya termasuk akurasi yang rendah dan ketergantungan pada kualitas gambar serta variasi pergerakan mata yang kompleks.

Klasifikasi Arah Mata Berdasarkan *Facial Landmark* Menggunakan CNN [5]	M. A. Nurdin, R. C. Wihandika, F. Utaminingrum	*Convolution Neural Network (CNN)*	Rata-rata akurasi sebesar 95% menggunakan 2 layer dengan 32 dan 64 filter. CNN sangat efektif dalam mengekstraksi fitur visual dari gambar dan mencapai akurasi tinggi sebesar 95%. Tetapi dataset yang digunakan relatif kecil (sekitar 200 sampel pergerakan mata), yang dapat mempengaruhi generalisasi model. CNN juga memerlukan sumber daya komputasi yang besar dan kurang optimal untuk menangani data temporal.
Implementasi *Deep Learning* Untuk Pendeteksian Kantuk Berdasarkan Fitur Kedipan [34]	Hanif Annur Rahman	*Independent Recurrent Neural Network (IndRNN), Support Vector Machine (SVM)*	Akurasi 91.88% untuk *IndRNN* dibandingkan dengan 86.48% untuk *HM-LSTM*. IndRNN lebih cepat dalam proses pelatihan dan lebih efektif dalam menangani data sekuensial atau temporal. Selain itu IndRNN lebih banyak diimplementasikan untuk pemrosesan teks dan belum banyak diaplikasikan pada klasifikasi arah mata. Implementasi dan pemahaman model IndRNN lebih kompleks dibandingkan model lainnya.

Independently Recurrent Neural Network (IndRNN): *Building A Longer and Deeper RNN* [10]	S. Li, W. Li, C. Cook, C. Zhu, Y. Gao	*IndRNN*	*IndRNN* dapat mempelajari pola jangka panjang dan menghasilkan kinerja yang lebih baik daripada *RNN* dan *LSTM* pada tugas pemodelan bahasa dan pengenalan angka tulisan tangan.
Klasifikasi Pola Pergerakan Bola Mata *Menggunakan Metode Multilayer Backpropagation* [45]	Karina Amadea, Fitra A. Bachtiar, Gusti Pangestu	*Multilayer Backpropagation Neural Network*	Metode ini berhasil mengklasifikasikan pola pergerakan bola mata dengan akurasi hingga 90%.
Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images [27]	Sakib Mostafa dan Fang-Xiang Wu	*Convolutional Autoencoder dan CNN*	Metode ini berhasil mendiagnosis autism spectrum disorder menggunakan citra MRI struktural dengan akurasi yang baik. Ekstraksi ciri fitur dilakukan menggunakan *CNN*.
Klasifikasi Citra Menggunakan *Convolutional Neural Network dan K Fold Cross Validation* [29]	Peryanto Ari, Anton Yudhana, Rusydi Umar	*CNN dan K-Fold Cross Validation*	Metode ini berhasil mengklasifikasikan citra dengan akurasi yang tinggi. CNN digunakan untuk ekstraksi ciri fitur dari citra.