BAB 2
TINJAUAN PUSTAKA

2.1 Batik

Batik merupakan warisan budaya Indonesia yang memiliki keragaman motif untuk setiap daerah. Keragaman motif batik banyak dipengaruhi oleh budaya luar dan perkembangan zaman dari mulai kaligrafi Arab, karangan bunga Eropa, burung *phoenix* China hingga bunga sakura Jepang [1]. Dari keragaman motif batik tersebut setidaknya terdapat 181 motif batik yang ada di Indonesia menurut buku Batik: spirit of Indonesia [2]. Beberapa motif batik yang ada di Indonesia yaitu :

2.1.1 Buketan

Buketan adalah salah satu motif batik yang berasal dari Pekalongan. Buketan adalah kata serapan dari bahasa asing yaitu Bouqet. Motif ini identik dengan corak bunga. Motif buketan lahir dari keindahan tumbuhan, kesuburan dan geografis di pulau jawa.

Gambar 1.1 Motif batik buketan

2.1.2 Megamendung

Motif batik ini berasal dari daerah Cirebon memiliki pola awan dengan warna yang tegas. Motif tersebut melambangkan kehidupan manusia dan hubungannya dengan alam semesta [9].

Gambar 1.2 Motif batik megamendung

2.1.3 Sidomukti

Motif batik sidomukti melambangkan harapan untuk hidup yang mulia dan sejahtera [9]. Motif khas Solo mempunyai warna dominan coklat. Pada motif sidomukti terdapat motif bergambar kupu-kupu yang melambangkan simbol harapan dan motif berupa bangunan tahta yang melambangkan derajat yang tinggi [10].

Gambar 1.3 Motif batik sidomukti

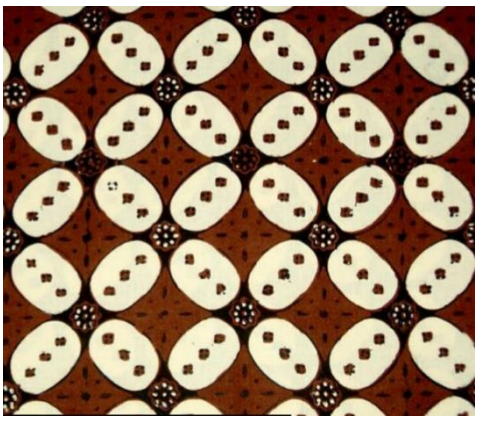
2.1.4 Ceplok

Pola Ceplok merupakan pola-pola batik kuno yang terdapat pada hiasan arca di Candi Hindu dan Budha dengan bentuk kotak-kotak, lingkaran, binatang, bentuk tertutup serta garis-garis miring. Pola dasar yang terdapat pada candi Hindu di arca Ganesha dari Banon Borobudur, arca Hari Hara dari Blitar, Ganesha dari Kediri dan arca arca Parwati dari Jawa merupakan pola dasar dari Pola Kawung. Dasar pola Ceplok terdapat di arca Budha antara lain Budha Mahadewa dari Tumpang dan arca Brkhuti dari candi Jago. Contoh dari motif batik ceplok dapat dilihat pada gambar 2.4 [11].

Gambar 1.4 Motif batik ceplok

2.1.5 Kawung

Pada motif batik kawung, polanya berbentuk irisan buah kawung atau kolang-kaling. Buah yang didapat dari pohon aren ini bermakna bahwa dalam masyarakat Jawa sebaiknya kebaikan hati tidak perlu diketahui orang lain. Selain itu, bunga teratai juga menjadi intrepretasi lain dalam menggambarkan motif batik ini. Empat lembar kelopak bunga teratai ini mengisyaratkan kesucian dan umur yang panjang [12].



Gambar 1.5 Motif batik kawung

2.1.6 Sekarjagad

Motif Batik Sekarjagad adalah salah satu motif yang sangat khas di Indonesia. Batik jenis ini berasal dari Jawa, lebih tepatnya Yogyakarta dan Solo. Kata Sekar Jagad itu sendiri diambil dari perpaduan antara dua bahasa yang dimana "Kaart" dalam Bahasa Belanda berarti "peta" dan "Jagad" dalam Bahasa Jawa berarti "dunia", sehingga motif ini juga melambangkan keragaman baik yang terdapat di Indonesia maupun di seluruh dunia. Sehingga makna yang dihasilkan dari batik yang satu ini sangat luas dan menyeluruh dari semua aspek kehidupan [13].

Gambar 1.6 Motif batik sekarjagad

Motif Sekarjagad memiliki pola geometris berbentuk ceplok yang berulang dan biasanya berisi bermacam bunga. Terkadang bentuk ceplok pada motif Sekar Jagad terlihat seperti peta dunia. Hal ini menggambarkan pesan yang terkandung dalam motif Sekar Jagad berupa keragaman dan harmoni dunia [14].

2.1.7 Lereng

Batik Motif Lereng adalah salah satu jenis batik dengan pola diagonal. Pola diagonal pada batik dapat dilihat pada susunan pola yang berulang dan membentuk corak sama. Pola yang berbentuk miring. Motif ini berarti juga topo broto para raja yang dilakukan di lereng-lereng pegunungan untuk mendapatkan wahyu atau wangsit. Dalam tapa brata itulah mereka dapat melihat pemandangan gunung dan pegunungan yang berderet-deret sehingga menyerupai pereng atau lereng. Untuk itu maka motif ini dijadikan sebagai simbol kesuburan [15].

Gambar 1.7 Motif batik lereng

2.2 Pengolahan Citra

Citra atau gambar merupakan salah satu media yang dapat menyampaikan informasi lebih detil dan jelas bila dibandingkan dengan teks, penggunaannya pun dapat merepresentasikan suatu objek secara lebih nyata, namun tidak semua citra memiliki kualitas yang baik [11].

Dalam pengolahan citra digital terutama citra yang digunakan sebagai data masukan suatu proses, perlu memiliki kualitas yang baik. Hal – hal yang mengganggu kualitas citra contohnya ialah adanya bercak hitam atau putih yang bukan bagian dari objek pada citra.

Pada pengolahan citra terdapat beberapa metode untuk memanipulasi gambar yaitu memotong gambar, memutar gambar dan proses *grayscale* gambar. Metode tersebut akan dilakukan pada tahap *preprocessing* di penelitian ini.

2.2.1 *Scaling* Gambar

Perubahan ukuran gambar sangat diperlukan untuk memanipulasi gambar tersebut sesuai dengan kebutuhan yang diperlukan. Perubahan gambar memiliki rumus pada gambar 2.4 sebagai berikut [12].

```
def resizeGambar( source, newWid, newHt ):
    target = []
    width = getWidth( source )
    height = getHeight( source )
    for x in range(0, newWid):
      for y in range(0, newHt):
        srcX = int( round( float(x) / float(newWid) * float(width) ) )
        srcY = int( round( float(y) / float(newHt) * float(height) ) )
        srcX = min( srcX, width-1)
        srcY = min( srcY, height-1)
        target[x, y] = source[srcX, srcY]

    return target
```

Gambar 1.8 Potongan *code scaling* gambar

Keterangan:

source = matriks gambar sebelum di-*scale*

newWid = panjang baru

newHt = lebar baru

2.2.2 Rotasi Gambar

Rotasi gambar dilakukan agar memiliki nilai *pixel* yang berbeda tetapi tetap memiliki maksud gambar yang sama. Proses ini memiliki rumus sebagai berikut [13].

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

Keterangan:

x'= posisi x pada gambar setelah rotasi

y'= posisi y pada gambar setelah rotasi

x = posisi x pada gambar sebelum rotasi

y = posisi y pada gambar sebelum rotasi

2.3 Kecerdasan Buatan

Kecerdasan Buatan atau *Artificial Intelligence* merupakan suatu ilmu yang mempelajari bagaimana membuat komputer melakukan sesuatu pada suatu kejadian atau peristiwa yang mana orang melakukannya dengan baik. Kecerdasan buatan merupakan proses di mana peralatan mekanik dapat melaksanakan kejadian-kejadian dengan menggunakan pemikiran atau kecerdasan seperti manusia [14].

Aplikasi-aplikasi yang dibuat menggunakan kecerdasan buatan dapat berfikir layaknya manusia seperti menentukan keputusan, menarik kesimpulan, rekomendasi keputusan dan otomasi sebuah sistem.

2.4 Machine Learning

Machine learning merupakan salah satu bidang kajian dari kecerdasan buatan yang belajar memprediksi sesuatu dari sebuah contoh. Daripada sebuah komputer diberikan sejumlah peraturan untuk dipecahkan, *machine learning* memberikan model untuk dievaluasi dan diberikan perintah untuk mengubah model ketika terjadi kesalahan [15].

Machine learning sudah digunakan di perusahaan-perusahaan besar seperti Amazon, Netflix dan Google [16]. Google menerapkan *machine learning* pada *search engine*-nya, lalu Amazon dan Netflix menggunakan *machine learning* untuk merekomendasikan barang atau video yang sejenis ke pelanggannya. *Machine*

learning pun dapat digunakan untuk mengenali sebuah motif batik seperti penelitian yang telah dilakukan oleh Wicaksono [3].

Salah satu kemampuan dari *machine learning* adalah dapat mengenali objek. Salah satu metode yang digunakan untuk mengenali objek yaitu *convolutional neural network* yang akan dibahas pada sub-bab selanjutnya.

2.5 *Convolutional Neural Network*

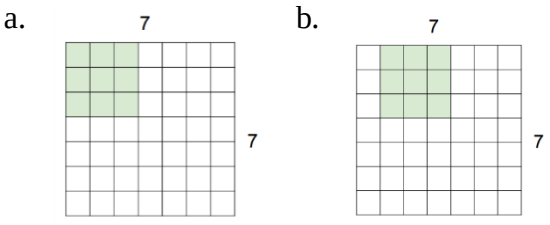
Convolutional Neural Network dikenal juga dengan Convolutional Network. Convolutional Neural Network adalah sebuah neural network sederhana yang setidaknya pada setiap layer yang ada melakukan satu kali perkalian matrix.

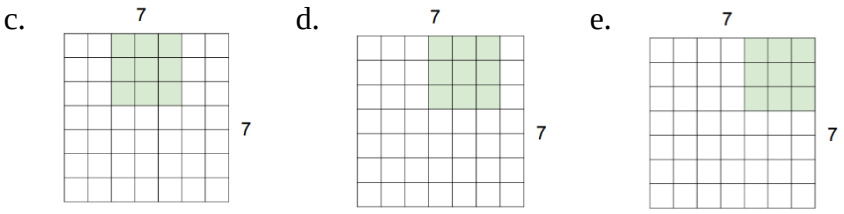
Pada convolutional neural network terdapat tiga layer utama yaitu *convolutional layer*, *pooling layer* dan ReLU layer.

2.5.1 *Convolutional Layer*

Convolutional Layer adalah sebuah *layer* dimana *input* matriks diubah menjadi matriks yang baru menggunakan matriks *filter* atau *kernel* yang ada. Fungsi dari *convolutional layer* adalah membuat *input* matriks atau gambar yang dimasukkan menjadi terlihat tepian-tepiannya.

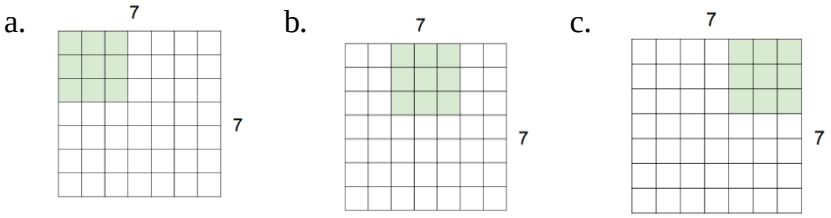
Pada convolutional neural network terdapat beberapa properti yaitu panjang, lebar, kedalaman, *stride* dan *zero padding* [15]. Properti panjang dan lebar seperti kolom dan baris pada sebuah matriks. Properti kedalaman yaitu kedalaman warna sebuah gambar jika *input* matriksnya ada sebuah gambar jika kedalaman warna nilainya tiga berarti kedalaman warna tersebut adalah RGB. Properti *stride* adalah properti untuk mengatur berapa langkah matriks filter bergerak pada sebuah matriks *input*. Cara kerja dari *convolutional layer* yaitu menerapkan matriks filter pada sebuah *input* matriks dengan menghitung *dot product* dari kedua matriks [15].





Gambar 1.9 Contoh matriks pada *convolutional layer* dengan *stride* satu

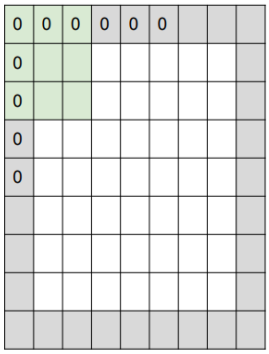
Pada gambar 2.9 dimisalkan ukuran matriks 7x7x1 dan ukuran *filter* yang digunakan berukuran 3x3x1 dengan nilai *stride* satu maka dari convolutional layer tersebut akan menghasilkan matriks 5x5.



Gambar 1.10 Contoh matriks pada *convolutional layer* dengan *stride* dua

Pada gambar 2.10 ukuran matriks input 7x7x1 dan ukuran *filter* yaitu 3x3x1 dengan nilai *stride* dua maka akan menghasilkan matriks berukuran 3x3.

Ketika mencoba menerapkan *filter* 3x3x1 pada matriks input berukuran 7x7x1 dengan menggunakan nilai *stride* tiga, hal tersebut tidak dapat dilakukan karena akan ada satu kolom yang tidak terkena *filter*. Solusinya yaitu dengan menambahkan nilai *zero padding*. Dari dua contoh sebelumnya *zero padding* memiliki nilai 0.

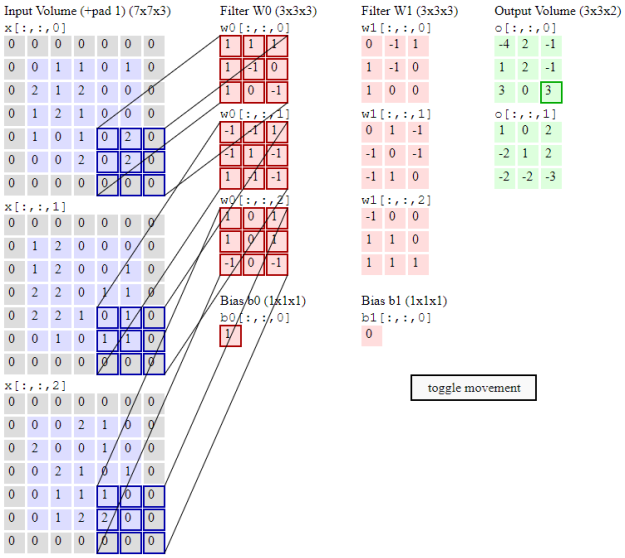


Gambar 1.11 Contoh *zero padding*

Zero padding adalah properti yang akan menambahkan ukuran dari matriks *input*, misal pada contoh sebelumnya *input* matriks berukuran 7x7x1 maka jika ditambahkan *zero padding* dengan nilai satu akan menjadi matriks berukuran

8x8x1. Maka dengan *input* matriks berukuran 8x8x1, ukuran *filter* 3x3x1 dan *stride* bernilai 3 akan menghasilkan matriks baru dengan ukuran 3x3x1.

Gambar 2.12 memperlihatkan perhitungan *convolutional* dengan kasus *input* matriks memiliki kedalaman warna 3. Jumlah *filter* akan sama dengan kedalaman dari *input* matriks. Perhitungan dari gambar 2.7 sama seperti *convolutional* pada kedalaman matriks 1, hanya saja setiap hasil *dot product* dengan *filter* dijumlahkan. Hasil tersebut yang akan menjadi hasil *convolutional*.



Gambar 1.12 Contoh convolutional dengan kedalaman warna lebih dari 1

Pada *convolutional neural network*, *convolutional layer* akan menghasilkan parameter berupa bobot dan bias dengan rumus sebagai berikut.

$$bobot = filter * filter * depth * kernel \quad (2)$$

$$bias = kernel \quad (3)$$

$$total\ parameter\ = bobot + bias \quad (4)$$

Keterangan:

filter = ukuran *filter*

depth = kedalaman warna / jumlah *channel*

kernel = jumlah *kernel* yang digunakan

2.5.2 ReLU (*Rectified Linear Units*) *Layer*

ReLU adalah salah fungsi aktifasi pada sebuah *neural network*. Fungsi aktifasi ini sering digunakan pada *convolutional neural network*. Fungsi dari ReLU yaitu :

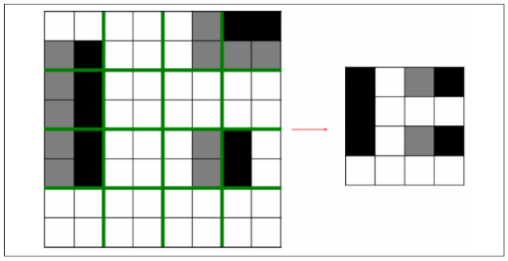
$$R(z) = \max(0, z) \quad (5)$$

Jika z kurang dari 0 maka hasil R(z) adalah 0 dan ketika z bernilai 0 atau lebih dari 0 maka hasil dari R(z) adalah z itu sendiri [17].

2.5.3 *Max Pooling Layer*

Max Pooling layer pada dasarnya sama seperti *convolutinal layer*. Pembeda dari kedua *layer* ini yaitu tidak lazimnya penggunaan *zero padding* pada *max pooling layer*. Fungsi dari *layer* ini mengubah ukuran *input* matriks atau gambar yang dimasukkan menjadi lebih kecil agar mengurangi jumlah parameter pada komputasi.

Pada *convolutional layer*, *filter* digunakan untuk mengubah dengan cara menghitung *dot product input* matriks dan *filter* yang digunakan, sedangkan pada *max pooling layer*, *filter* digunakan untuk mengambil nilai maksimum dari *input* matriks sesuai dengan ukuran *filter* yang digunakan.



Gambar 1.13 Contoh *pooling layer*

2.5.4 *Global Average Pooling Layer*

Global Average Pooling layer digunakan untuk mencegah overfitting dengan mengurangi jumlah parameter pada model [18]. Jika jumlah parameter yang lebih sedikit akan mempercepat proses pelatihan. *Global Average Pooling* mengubah matriks 3D menjadi 1D atau vektor. *Global Average Pooling* memiliki rumus sebagai berikut.

$$GAP^c = \frac{1}{N}\sum_{xy} z_{xy}^c \quad (6)$$

Keterangan:

z_{xy}^c = nilai elemen matriks pada posisi ke c, kolom x dan baris y

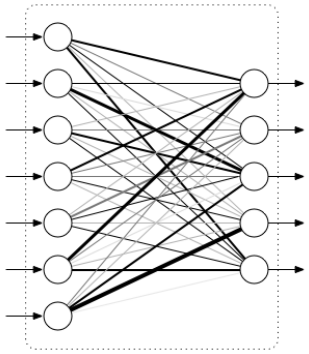
x = jumlah kolom

y = jumlah baris

N = x * y

2.5.5 *Fully Connected Layer*

Fully connected layer adalah *layer* dimana semua *neuron* yang ada terhubung dengan semua masukkannya [19]. *Fully connected layer* dapat digambarkan sama seperti neural network pada umumnya dan dapat dilihat pada gambar 2.9.



Gambar 1.14 Fully connected layer

Untuk menghitung nilai di *layer* selanjutnya menggunakan rumus pada *neural network* pada umumnya seperti berikut.

$$O_k = \left(\sum_{i=0}^{m} x_{k,i}\, w_{k,i} \right) + b_k \;\; k = 0, 1, 2, \ldots t \tag{7}$$

Keterangan:

t = total *neuron* pada *layer* yang dituju

m = total *neuron* pada *layer* sebelumnya

O = nilai *output*

x = nilai *input*

w = bobot

b = bias

2.6 Inisialisasi Bobot *Glorot*

Inisialisasi bobot *Glorot* adalah salah satu metode untuk menginisialisasikan bobot. Inisialisasi bobot *Glorot* memiliki rumus untuk mendapatkan rentang *random* angka [20].

$$w = [-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}] \quad (8)$$

Keterangan:

w = bobot

n_j = jumlah neuron pada layer tersebut

n_{j+1} = jumlah neuron di layer setelahnya

Hasil dari perhitungan di atas akan menjadi rentang angka random pada saat menginisialisasikan bobot.

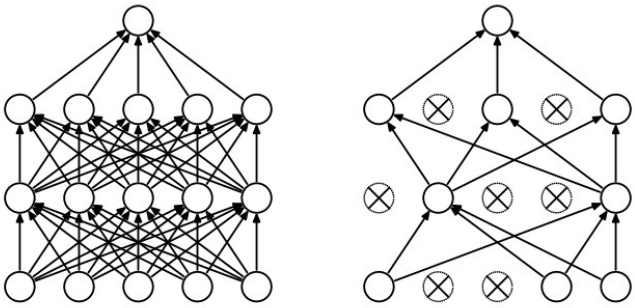
2.7 *One Hot Encoding*

Machine learning tidak dapat mengenali data berupa kategorikal secara langsung. Label berupa kategorikal harus diubah agar dapat dikenali oleh *machine learning*. Maka dari itu *one hot encoding* digunakan untuk merepresentasikan data kategorikal menjadi dikenali oleh *machine learning* [21].

Contoh terdapat data sebagai berikut: buketan, megamendung, buketan, sidomukti dan megamendung. Total data tersebut jika diambil yang uniknya saja maka terdapat tiga data yaitu buketan, megamendung dan sidomukti. Dari ketiga data tersebut ubah menjadi sebuah index maka didapatkan buketan = 0 lalu megamendung = 1 dan sidomukti = 2. Setelah penentuan indeks, buatlah matriks berukuran sama dengan jumlah data tadi yaitu tiga. Isi dari matriks tersebut yaitu 1 pada indeks yang sesuai dengan penentuan indeks sebelumnya dan sisanya diberi nilai 0. Maka hasil dari *one hot encoding* yaitu buketan = [1, 0, 0] lalu megamendung = [0, 1, 0] dan sidomukti = [0, 0, 1].

2.8 Dropout

Dropout adalah salah satu teknik untuk menghindari *overfit* pada sebuah neural network [22]. *Dropout* ini berupa sebuah *layer* yang memiliki *mask* untuk mengaktifkan atau menonaktifkan sebuah *neuron*. Ilustrasinya dapat dilihat pada gambar 2.10.



Gambar 1.15 *Dropout*

Mask tersebut akan dikalikan dengan *input* matriks. *Mask* pada dropout didapatkan dengan *code python* seperti berikut.

```
(numpy.random.rand(*x.shape) < p ) / p
```

Gambar 1.16 Potongan *code python dropout*

Keterangan:

numpy = *library* matriks *ptyhon*

numpy.random.rand() = nilai random 0 hingga 1 dengan ukuran x.*shape*

x = *input*

x.*shape* = ukuran matriks x

p = persentase *dropout* / *rate*

2.9 *Softmax Activation*

Fungsi *softmax* menghitung probabilitas terhadap sejumlah kejadian. Dalam kasus *machine learning* fungsi ini akan menghitung probabilitas setiap label yang ditebaknya. Kelebihan dari fungsi aktifasi ini adalah nilai keluarannya yang berupa rentang probabilitas 0 sampai 1. Jika setiap hasil fungsi *softmax* dijumlahkan maka akan bernilai 1 [23]. Berikut adalah rumus dari fungsi aktifasi softmax.

$$S(O_t) = \frac{e^{(O_i)}}{\sum_{j=0}^{k} e^{(O_j)}} \quad k = 0, 1, 2, \dots t \tag{9}$$

Keterangan:

t = jumlah *neuron*

x = nilai *input*

2.10 *Crossentropy*

Crossentropy adalah salah satu fungi untuk mengetahui seberapa tinggi *error* atau *loss* yang dilakukan oleh sebuah *machine learning*. Fungsi ini digunakan ketika nilai *output* dari sebuah *machine learning* berupa probabilitas. [24]. Berikut adalah rumus dari *crossentropy*.

$$L = -\sum_{i=1}^{M} y_i \log(S(O_i)) \tag{10}$$

Keterangan:

M = jumlah label atau *class*

y = nilai *output*

p = nilai hasil prediksi

2.11 *Backpropagation*

Proses *backpropagation* ini bertujuan untuk memperbaiki parameter yang ada pada sebuah *machine learning* dengan menghitung *gradient* pada setiap parameternya. *Backpropagation* menggunakan konsep *chain rule*, yaitu hubungan antara turunan dari suatu rumus [25]. Berikut ialah tahapannya.

1. Menghitung *gradient* pada bobot di fully connected layer.

$$\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial fcO_i} * \frac{\partial fcO_i}{\partial fcI_i} * \frac{\partial fcI_i}{\partial w_{i,j}} \tag{11}$$

2. Menghitung turunan fungsi *crossentropy*.

$$\frac{\partial L}{\partial fcO_i} = -1 * (y_i * \left(\frac{1}{fcO_i}\right) + (1 - y_i) * \left(\frac{1}{1 - fcO_i}\right)) \tag{12}$$

3. Menghitung turunan nilai keluaran dari *fully connected layer* setelah diaktifkan dengan fungsi aktifasi *softmax*.

$$\frac{\partial fcO_i}{\partial fcI_i} = \frac{e^{fcI_i} * (\sum_{n \neq i}^{t} e^{fcI_n})}{\sum_{n=0}^{t} {e^{fcI_n}}^2} \; i = 0, 1, 2, \dots t \tag{13}$$

4. Menghitung turunan bobot dari *fully connected layer*.

$$\frac{\partial fcI_i}{\partial w_{i,j}} = OutputLayerSebelumnya_j \; i = 0,1,2, \dots t \tag{14}$$

Keterangan:

fcI_i = nilai keluaran *fully connected layer* pada index ke i

fcO_i = nilai masukkan *fully connected layer* pada index ke i

L = fungsi loss yang digunakan (pada penelitian ini adalah fungsi *crossentropy*)

w = bobot

t = jumlah *neuron*

j = 0,1,2, … jumlah *neuron* pada layer sebelumnya

2.12 Adam Optimizer

Seperti namanya *optimizer* digunakan untuk mengoptimasi suatu parameter, mengoptimasi bisa membuat nilai sebuah parameter menjadi maksimum ataupun minimum. Adam Optimizer adalah salah satu adaptif *learning rate* optimasi yang mengkombinasikan RMSProp dan momentum [6]. Sebelum menggunakan adam sebagai optimizer, ada beberapa nilai yang harus diinisialisasikan yaitu [6]:

1. m = 0
2. v = 0
3. $\epsilon = 10^{-8}$
4. $t = 0$
5. $\alpha = 0.001$
6. $\beta_1 = 0.9$
7. $\beta_2 = 0.999$

Tahap-tahap yang dilakukan oleh *Adam Optimizer* yaitu [6]:

1. Tambah t setiap iterasi

$$t = t + 1 \tag{15}$$

2. Menghitung *gradient*

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \tag{16}$$

3. Menghitung bias *first moment*

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g \tag{17}$$

4. Menghitung bias *second moment*

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g \cdot g \tag{18}$$

5. Memperbaiki bias *first moment*

$$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t} \quad (19)$$

6. Memperbaiki bias *second moment*

$$\widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t} \quad (20)$$

7. Memperbaiki parameter

$$\theta_t = \theta_{t-1} - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \quad (21)$$

Keterangan:

g = *gradient*

m = *first moment*

v = *second moment*

β_1, β_2 = *Exponential decay rates*

α = *Step size* atau *learning rate*

θ = parameter yang akan diperbaiki (pada kasus ini adalah bobot)

Ketujuh tahap tersebut diulang sebanyak jumlah *dataset* yang diambil secara *random* hingga semua *epoch* telah selesai. Perbedaan antara *Adam* dengan *RMSProp* yaitu ada pada perubahan *step size* pada awal perubahan parameter dikarenakan adam melakukan *bias correction* pada perhitungannya.

2.13 *RMSprop*

RMSprop salah satu *optimizer* yang mempertahankan rata-rata dari kuadrat *gradient* untuk setiap bobot. Adapun rumus dari *RMSprop* sebagai berikut [31].

$$MeanSquare(w,t) = \rho * MeanSquare(w,t-1) + 0.1(\frac{\partial E}{\partial w}(t))^2 \quad (22)$$

Keterangan :

w = bobot

t = *timestamp*

ρ = 0.9

$\frac{\partial E}{\partial w}$ = *gradient*

Jika hasil diatas dilakukan akar ($\sqrt{MeanSquare(w,t)}$) hasilnya akan lebih bagus [31].

2.14 SGD

SGD adalah optimizer yang sangat sederhana. Proses perbaikan bobot hanya dengan mengkalikan *gradient* dengan *learning rate* [5]. Walaupun *SGD* memiliki proses yang sederhana tetapi memakan waktu yang lama untuk mendekati konvergen. Adapun rumus dari SGD sebagai berikut [5].

$$\theta = \theta - \alpha * \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \quad (23)$$

Keterangan:

θ = bobot atau bias

$\nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ = *gradient* terhadap bobot atau bias, x dan y

x = input

y = label

2.15 UML (*Unified Modeling Language*)

UML adalah salah satu bahasa pemodelan untuk memodelkan suatu perangkat lunak agar perangkat lunak tersebut dapat tergambar secara detail [26]. Keuntungan menggunakan UML yaitu perangkat lunak yang dimodelkan akan terdokumentasi dengan baik maka ketika perangkat lunak sudah berjalan lama dan berganti *developer* maka *developer* yang baru akan mudah mengerti dengan membaca UML tersebut.

Pada UML terdapat beberapa diagram yang digunakan pada penelitian ini yaitu:

1. *Use case* diagram

   Use case diagram adalah diagram untuk mengambarkan fungsi yang ada pada sebuah perangkat lunak. Tujuan dari *use case* diagram untuk menggambarkan relasi antara fungsi pada perangkat lunak dengan aktor atau bisa disebut pengguna [26].
2. *Use case* skenario

Use case skenario adalah deskripsi dari sebuah *use case*. Penjelasan ini diperlukan untuk mendetailkan fungsi agar lebih dimengerti oleh pembaca terutama *developer* [26].

3. *Activity* diagram

   Activity diagram digunakan untuk menggambarkan alur dari proses, bisnis proses, alur pada use case diagram [26].

4. *Class* diagram

   Class diagram digunakan untuk menggambarkan *class* apa saja yang nantinya akan ada pada perangkat lunak tersebut [26]. *Class* yang digambarkan berasal dari objek-objek pada perangkat lunak tersebut.

5. *Sequence* diagram

   Sequence diagram digunakan untuk menggambarkan komunikasi antar *class-class* yang dibuat pada *class* diagram dan juga komunikasi antara *class-class* tersebut dengan aktor.

2.16 *Python*

Python adalah bahasa pemrograman tingkat tinggi untuk pemrograman yang bersifat *general-purpose* [27]. *Python* menggunakan interpreter dan *object oriented programming*. Pada mulanya python diciptakan oleh Guido Van Rossum dan dirilis pada tahun 1991. Kini *python* dikembangkan oleh *Python Software Foundation*.

Pada saat ini bahasa pemrograman *python* sering digunakan untuk melakukan penelitian atau pembuatan aplikasi *machine learning* dikarenakan banyak *library* yang mendukung dan bahasa pemrograman *python* sama seperti matematika. Kelebihan dari bahasa pemrograman *python* yaitu banyaknya library yang sedang dikembangkan, mudah untuk dipelajari dan dapat diintegrasikan dengan bahasa pemrograman lain seperti C, C++ atau java [28]. Walaupun begitu bahasa pemrograman python memiliki beberapa kelemahan yaitu lemah dalam bidang teknologi mobile, lemah dalam pengembangan database layer dan run-time error (error hanya terlihat pada saat program sudah dijalankan) [28].