BAB II
TINJAUAN PUSTAKA

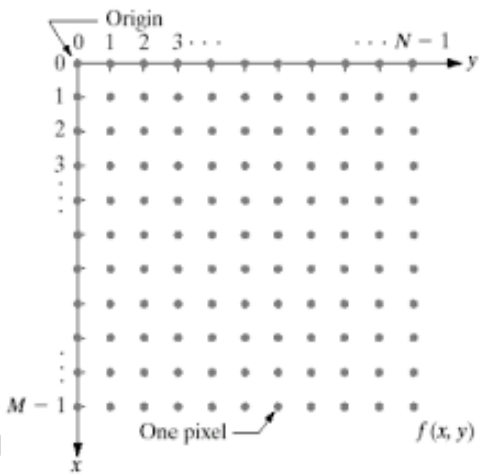
Pada BAB ini berisi tentang beberapa penjelasan teori yang berkaitan dengan metode *Convolutional Neural Network* (CNN). Penjelasan tersebut bertujuan untuk menunjang perancangan penelitian.

2.1 Citra Digital

Secara umum citra adalah kombinasi titik, garis, bidang, dan warna, untuk membuat tiruan dari suatu objek, biasanya sebuah objek fisik atau orang. Bentuk dari sebuah citra adalah gambar dua dimensi seperti foto, lukisan, untuk gambar tiga dimensi adalah patung. Pada dasarnya citra dibagi menjadi dua yaitu citra analog dan citra digital. Citra analog tidak dapat direpresentasikan oleh komputer, sehingga tidak dapat diproses secara langsung oleh komputer. Agar citra analog dapat diproses oleh komputer, terlebih dahulu harus diubah atau diubah menjadi citra digital (Sutojo, 2017).

Gambar dapat disajikan sebagai fungsi dua dimensi *f(x,y)* dimana *x* dan *y* adalah koordinat pada bidang pasial dan *f* pada setiap fungsi *(x,y)* sebanding dengan tingkat keabuan atau tingkat keabuan atau *gray level* dari citra pada titik tersebut. (Putra, 2010). Jika nilai x,y dan nilai *amplitude* f berhingga dan diskrit, maka citra tersebut dapat dikatakan citra digital.

Secara matematis, citra digital merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Agar citra dapat diproses oleh komputer, citra harus direpresentasikan secara numerik dengan nilai diskrit. Posisi koordinat citra digital dapat dilihat pada gambar 2.1.



Gambar 2.1 Koordinat Citra Digital

(sumber gambar *:* *https://nenkk.wordpress.com/2013/06/26/representasi-citra-digital/*)

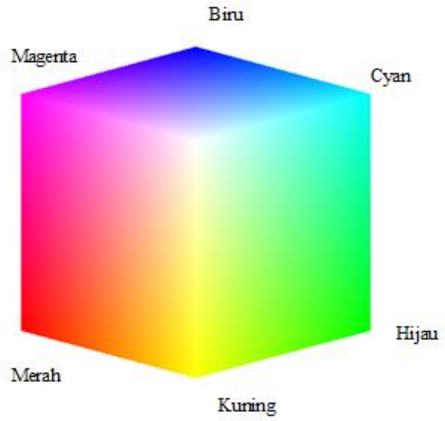
Dalam citra digital 2D, f (x, y) dilambangkan dengan M sebagai baris dan N sebagai kolom. Dalam citra digital, perpotongan antara kolom dan baris disebut *pixel* (*picture element*). Oleh karena itu, gambar dapat diilustrasikan dalam matriks sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad (2.1)$$

Indeks pada kolom (x) dan indeks pada baris (y) mewakili koordinat suatu titik pada citra, sedangkan f (x, y) adalah intensitas (skala abu-abu) pada titik (x,y).

2.2 Citra Warna

Citra berwarna adalah jenis citra yang menggabungkan 3 komponen warna utama yaitu *Red* (R), *Green* (G) dan *Blue* (B). Triwarna ini disebut warna primer dan sering disingkat sebagai warna dasar RGB (Jamaludin, 2021). Setiap warna utama memiliki penyimpanan sebesar 8 bit. Karena kondisi tersebut citra RGB dapat disebut dengan citra 24-bits. Dengan ini, kemungkinan warna dari citra 24-bit adalah $(2^8)^3$ = 16,777,216. Pada masing masing komponen tersebut mempunyai nilai *pixel* dengan rentang nilai 0 sampai 255. Nilai 0 mempresentasikan warna gelap dan nilai 255 mempresentasikan putih (jika citra tersebut *grayscaleI*).



Gambar 2.2 Kubus Warna 24 bit

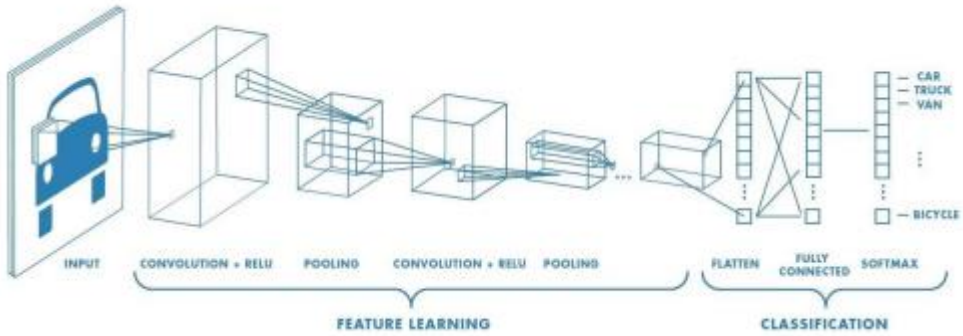
(sumber *gambar :* *http://www.kitainformatika.com/2017/02/konversi-ruang-warna-rgb-ke-cmy-dan-cmy.html*)

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu algoritma *deep learning* yang dikembangkan dari *Multilayer Percepton* (MLP), dirancang untuk memproses pola gambar yang berbeda dari sisi data yang berbeda dalam bentuk *gird*, Contoh gambar dua dimensi adalah gambar atau suara. CNN pertama kali dikembangkan oleh Fukusima (1980) di Tokyo, Jepang, namun pada saat itu masih dikenal sebagai *neurocognition*. Kemudian, CNN dikembangkan kembali oleh LeCun, yang terinspirasi oleh model *neurocognition* yang dibuat sebelumnya. Model CNN LeNet telah berhasil diterapkan oleh LeCun dalam masalah pengenalan angka pada tulisan tangan (Anugerah, 2018). Tahun 2012 (Krizhevsky, Sutskever, & Hinton, 2012) menerapkan metode CNN ke *Imagenet LargeScale Visual Recognition Challenge* (ILSVRC) dan memenangkan kompetisi. Sejak kemenangan Alex Krizhevsky dalam kompetisi tersebut, dapat dibuktikan bahwa metode CNN dapat mengungguli pengenalan objek citra dengan mengalahkan metode *machine learning* lainnya.

CNN memiliki lapisan tunggal dengan susunan tiga dimensi neutron. meliputi (lebar, tinggi dan kedalaman). Lebar dan tinggi adalah ukuran. matriks didasarkan pada lapisan, sedangkan kedalaman mengacu pada kuantitas. Saluran

warna gambar atau bagian lain juga dapat disebut sebagai angka. penyaringan, kemudian pada lapisan ini secara berurutan melalui berbagai proses.



Gambar 2.3 Ilustrasi Arsitektur CNN

(*sumber gambar *: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94***)***

Pada gambar 2.3 ilustrasi arsitektur CNN ini merupakan gambaran yang secara umum memiliki lapisan yang dibagi menjadi dua tipe, yaitu *feature learning* atau disebut lapisan Ekstrasi fitur dan *classification* disebut lapisan klasifikasi. Pada lapisan pertama adalah *feature learning*, lapisan di atas arsitektur yang pada arsitektur awal ini memiliki banyak lapisan dan setiap lapisan itu sendiri terdiri dari neuron yang terhubung ke area lapisan sebelumnya atau dengan area lokal.

Kemudian untuk lapisan pengklasifikasi kedua, yang terdiri dari beberapa lapisan, masing-masing dengan lapisan neuronnya sendiri yang terhubung penuh (*fully connected*) dengan lapisan lainnya. Pengklasifikasi ini menerima input dari lapisan ekstraksi fitur sebagai gambar vektor, dan kemudian gambar vektor diubah menjadi jaringan multi-neural dengan menambahkan beberapa lapisan tersembunyi. Hasil dari pengolahan ini adalah skor kelas untuk klasifikasi.

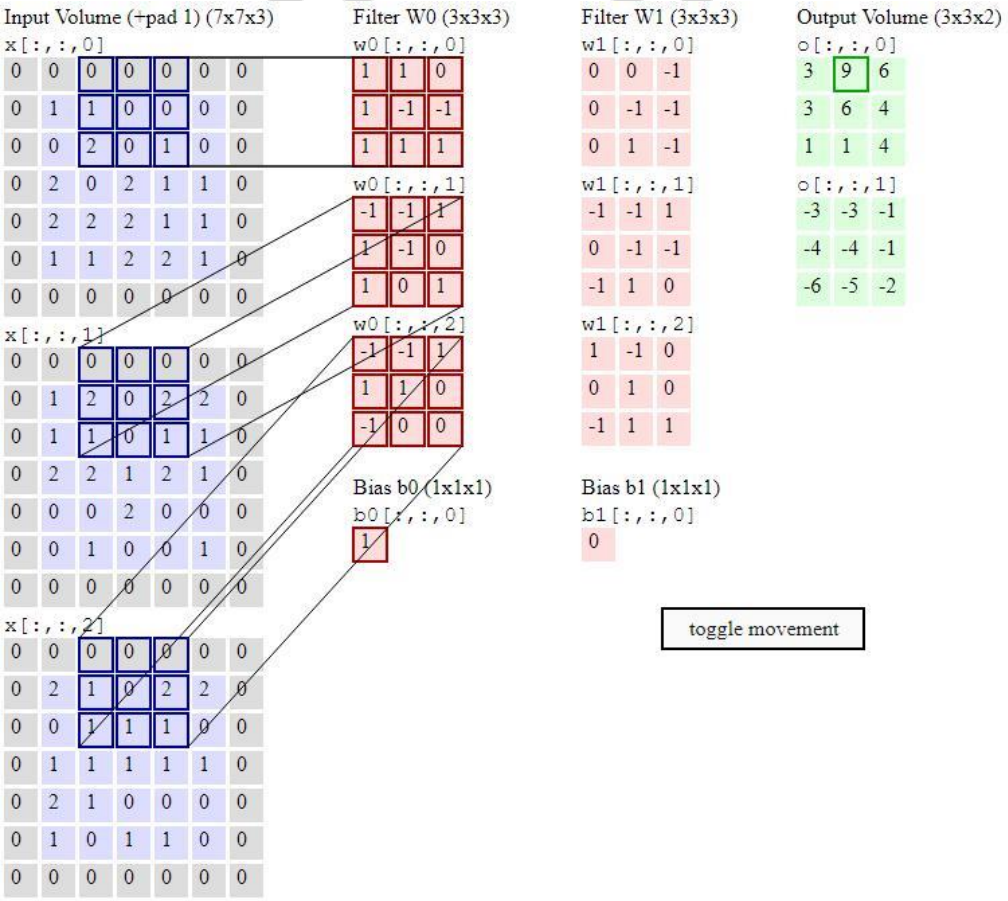
2.3.1 Convolution Layer

Convolution Layer merupakan proses iteratif dari operasi matematika yang disebut konvolusi, dan hasil dari proses konvolusi tersebut menghasilkan informasi yang dikenal sebagai *feature maps.* (Nielsen, 2015) .

Convolution layer memiliki parameter yang terdiri dari *kernel size*, *filters*, *strides*, dan *padding*. Parameter *kernel size* menentukan berapa ukuran konvolusi yang akan di filter. Parameter *filters* menentukan berapa banyak *feature maps* yang

akan di hasilkan dari proses *convolution*. Parameter *stride* menentukan berapa jarak *kernel* bergeser. Dan parameter *padding* menambahkan nilai 0 di tepi terluar *input* (Kazutaka Uchida, 2018).

Pada Gambar 2.4, kotak biru mewakili data input dari citra yang diproses atau konvolusi, kotak merah adalah kernel yang akan bergerak dari sudut kiri atas ke sudut kanan bawah dari input citra. kemudian kotak hijau tersebut dibangkitkan oleh 2 feature maps berukuran 3×3 dari hasil konvolusi .



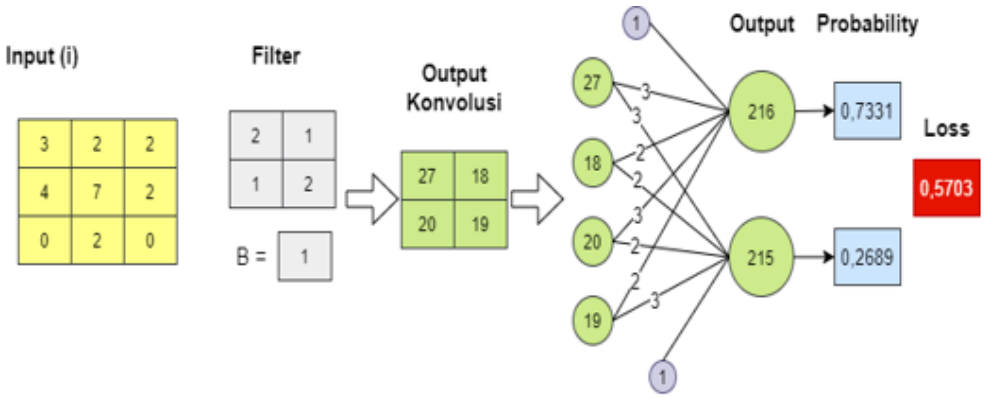
Gambar 2.4 Operasi Convolution

(sumber gambar : https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94**)**

2.3.2 Contoh Perhitungan CNN

Pada Sub-bab ini, penulis dapat menjelaskan tentang bagaimana contoh perhitungan CNN sederhana yang terdiri dari *forward propagation* dan *backward*

propagatioan. Forward propagation merupakan tahap awal dari proses perhitungan CNN, input akan melalui beberapa *layer* untuk nantinya menjadi *output layer*.Sedangkan pada proses *backward propagation*, terjadi *update* nilai pada masing masing bobot dan bias dengan berdasarkan nilai *error* dari *forward propagation* dan *learning rate* (Putra,2016).



Gambar 2.5 Ilustrasi Perhitungan CNN

(sumber gambar : https://cs231n.github.io/convolutional-networks/)

Pada Gambar 2.5 merupakan ilustrasi operasi CNN sederhana sebagai contoh yang menggambarkan langkah-langkah operasi CNN. Dalam Gambar 2.6, data masukannya dibuatlah citra dari mata ikan segar sebagai contoh dengan memiliki ukuran 3×3, *filter* atau kernel menggunakan ukuran 2×2, serta pergeseran *stride* sama dengan 1, *output* dari konvolusi, *flatten*, dan dua kelas *output* sebagai contoh yaitu kelas tidak segar dan segar, lalu untuk keluaran kebenarannya ditentukan yaitu kelas segar (x_2). Berikut ini adalah urutan matematis perhitungan forward propagation diawali dengan *convolution layer* :

$$oc_{1,1} = (i_{1,1} \times w_{1,1}) + (i_{1,2} \times w_{1,2}) + (i_{2,1} \times w_{2,1}) + (i_{2,2} \times w_{2,2}) + b$$

$$x_{1,1} = 6 + 2 + 4 + 14 + 1$$

$$x_{1,1} = 27$$

$$oc_{1,2} = (i_{1,2} \times w_{1,1}) + (i_{1,3} \times w_{1,2}) + (i_{2,2} \times w_{2,1}) + (i_{2,3} \times w_{2,2}) + b \quad (2.2)$$

$$x_{1,2} = 4 + 2 + 7 + 4 + 1$$

$$x_{1,2} = 18$$

$$oc_{2,1} = (i_{2,1} \times w_{1,1}) + (i_{2,2} \times w_{1,2}) + (i_{3,1} \times w_{2,1}) + (i_{3,2} \times w_{2,2}) + b$$

$$x_{2,1} = 8 + 7 + 0 + 4 + 1$$

$$x_{2,1} = 20$$

$$oc_{2,2} = (i_{2,2} \times w_{1,1}) + (i_{2,3} \times w_{1,2}) + (i_{3,2} \times w_{2,1}) + (i_{3,3} \times w_{2,2}) + b$$

$$x_{2,2} = 14 + 2 + 2 + 0 + 1$$

$$x_{2,2} = 19$$

Nilai $oc_{i,j}$ didefinisikan sebagai *output* dari operasi konvolusi. Berdasarkan perhitungan konvolusi di atas, maka diperoleh sebuah *output* berupa fitur atau citra baru hasil dari konvolusi sebagai berikut :

27	18
20	19

Gambar 2.6 Citra Baru Hasil Konvolusi

Setelah melalakukan operasi konvolusi, selanjutnya yaitu menghitung nilai *output* terhadap kelas yang ada.

$$tidak\ segar\ (x_1) = (oc_{1,1} \times w_{1,1}x_1) + (oc_{1,2} \times w_{1,2}x_1) + (oc_{2,1} \times w_{2,1}x_1) + (oc_{2,2} \times w_{2,2}x_1) + bx_1$$

$$x_1 = 27 \times 3 + 18 \times 2 + 20 \times 3 + 19 \times 2 + 1 \quad (2.3)$$

$$x_1 = 216$$

$$segar\ (x_2) = (oc_{1,1} \times w_{1,1}x_2) + (oc_{1,2} \times w_{1,2}x_2) + (oc_{2,1} \times w_{2,1}x_2) + (oc_{2,2} \times w_{2,2}x_2) + bx_2$$

$$x_2 = 27 \times 3 + 18 \times 2 + 20 \times 2 + 19 \times 3 + 1$$

$$x_2 = 215$$

Pada perhitungan *output* dilakukan perhitungan berupa perkalian dari hasil konvolusi ($oc_{i,j}$) dengan bobot-bobot jarigan ($w_{i,j}x_k$), lalu hasil perkalian tersebut dijumlahkan dengan bias (bx_i). Hasil *output* pada setiap kelas menjadi masukan untuk menghitung *probability* pada tiap kelas dengan memakai persamaan seperti di bawah ini :

$$\widehat{y}_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

$$= \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

$$Tidak\ Segar\ (\hat{y}_1) = \frac{2{,}7182^{216}}{2{,}7182^{216} + 2{,}7182^{215}} \tag{2.4}$$

$$= 0{,}7311$$

$$Segar\ (\hat{y}_2) = \frac{2{,}7182^{215}}{2{,}7182^{216} + 2{,}7182^{215}}$$

$$= 0{,}2689$$

Pada setiap masing-masing probabilitas, kelas tidak segar (x_1) mendapatkan nilai tertinggi sehingga model memprediksi kelas tidak segar (x_1) sebagai *output*, namun untuk *output* kebenarannya adalah kelas Segar (x_2). Salah satu cara untuk menjelaskan seberapa baik model yang dibangun adalah dengan mengetahui nilai dari *loss function* (Wulandari, 2019). Setelah mendapatkan nilai probabilitas pada setiap kelas, langkah selanjutnya yaitu menghitung nilai *loss function* dengan menggunakan persamaan dibawah ini:

$$Loss = -\sum_{i} y_i \log(\hat{y}_i)$$

$$= (0 \times \log(0{,}7311)) + (-1 \times \log(0{,}2689)) \tag{2.5}$$

$$= 0{,}5703$$

Pada perhitungan *loss function* di atas yaitu menggunakan *binary cross entropy*. Nilai *loss function* yang diperoleh dari perhitungan *binary cross entropy* yaitu sebesar 0,5703. Nilai tersebut dapat diminimalisir dengan dilakukannya *backward propagation*. *Backward propagation* bertujuan untuk meminimalisir nilai *loss* dengan melakukan pembaruan terhadap parameter-parameter (bobot dan bias). Tahap pertama pada *backward propagation* yaitu menghitung $\frac{\partial L}{\partial x_j}$ yang merupakan turunan dari persamaan di bawah ini:

$$L = -\sum_{i} y_i \log(\hat{y}_i)$$

$$\frac{\partial L}{\partial x_j} = -\sum_{i=1} \frac{\partial (y_i \log(\hat{y}_i))}{\partial x_j}$$

$$
\begin{aligned}
&= -\sum_{i=1} y_i \frac{\partial \log(\hat{y}_i)}{\partial x_j} \\
&= -\sum_{i=1} \left[y_i \frac{\partial \log(\hat{y}_i)}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial x_j} \right] \\
&= -\sum_{i=1} \left[\frac{y_i}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial x_j} \right] \\
&= -\left[\sum_{i \neq j} \left[\frac{y_i}{\hat{y}_i} - \hat{y}_i \hat{y}_j \right] + \left[\frac{y_j}{\hat{y}_j} \times \hat{y}_j (1 - \hat{y}_j) \right] \right] \\
&= -\left[\sum_{i \neq j} \left[-y_i \hat{y}_j \right] + \left[y_j (1 - \hat{y}_j) \right] \right] \\
&= -\left[-\hat{y}_j \sum_{i \neq j} y_i + \left[y_j (1 - \hat{y}_j) \right] \right] \\
&= \left[\hat{y}_j (1 - y_j) \right] - \left[y_j (1 - \hat{y}_j \right] \\
&= \hat{y}_j - \hat{y}_j y_j - y_j + \hat{y}_j y_j \\
&= \hat{y}_j - y_j
\end{aligned}
\tag{2.6}
$$

Dalam memperbarui parameter-parameter (bobot dan bias) dapat menggunakan persamaan di atas. Pada contoh ini, untuk nilai *learning rate* (η) yang ditentukan adalah 0,005. Bobot dan bias yang diperbarui pertama pada *backward propagation* ialah antara *flatten* dan *output*.

Untuk bobot:

$$
\frac{\partial L}{\partial w_{i,j} x_j} = \frac{\partial L}{\partial x_j} \frac{\partial x_j}{\partial w_{i,j} x_j} = (\hat{y}_j - y_i) \frac{\partial x_j}{\partial w_{i,j} x_j} \tag{2.7}
$$

$$
w_{i,j} x_l baru = w_k x_l lama - \eta \frac{\partial L}{\partial w_k x_l}
$$

Untuk bias:

$$
\frac{\partial L}{\partial b x_j} = \frac{\partial L}{\partial x_j} \frac{\partial x_j}{\partial b x_j} = (\hat{y}_j - y_i)
$$

$$
b x_l baru = b x_l lama - \eta \frac{\partial L}{\partial b x_l}
$$

Berikut merupakan contoh perhitungan pembaruan bobot dan bias yang baru untuk $w_{i,j}x_l$ dan bx_l.

Untuk bobot:

$$w_{1,1}x_1 baru = 3 - (0{,}005 \times 19{,}7384) = 2{,}9013$$
$$w_{1,2}x_1 baru = 2 - (0{,}005 \times 13{,}1589) = 1{,}9342$$
$$w_{2,2}x_2 baru = 3 - (0{,}005 \times (-13{,}8900)) = 3{,}0695 \quad (2.8)$$

Untuk bias:

$$bx_1 baru = 1 - (0{,}005 \times 0{,}7310) = 0{,}9963$$
$$bx_2 baru = 1 - (0{,}005 \times (-0{,}7310)) = 1{,}0037$$

Selanjutnya bobot dan bias diantara *input* dan *filter* diperbarui nilainya dengan menggunakan persamaan seperti berikut :

Untuk bobot:

$$\frac{\partial L}{\partial w_{i,j}} = \sum_{i,j}^{n} \frac{\partial L}{\partial oc_{i,j}} \frac{\partial oc_{i,j}}{\partial w_{i,j}}$$
$$w_{i,j} baru = w_{i,j} lama - lr * \frac{\partial L}{\partial w_{i,j}} \quad (2.9)$$

Untuk bias:

$$\frac{\partial L}{\partial b} = \sum_{i,j}^{n} \frac{\partial L}{\partial oc_{i,j}} \frac{\partial oc_{i,j}}{\partial b}$$
$$b\ baru = b\ lama - lr \times \frac{\partial L}{\partial b}$$

Dapat dilihat contoh perhitungan pembaruan parameter bobot dan bias untuk $w_{i,j}$ dan b .

Untuk bobot:

$$w_{1,1} baru = 2 - (0{,}005 \times (-24{,}1247)) = 2{,}1206$$
$$w_{1,2} baru = 1 - (0{,}005 \times (-15{,}3521)) = 1{,}0768 \quad (2.10)$$
$$w_{2,1} baru = 1 - (0{,}005 \times (-16{,}8142)) = 1{,}0841$$
$$w_{2,2} baru = 2 - (0{,}005 \times (-14{,}6211)) = 2{,}0731$$

Untuk bias:

$$b\ baru = 1 - (0{,}005 \times (-0{,}7311)) = 1{,}0037$$

Setelah mendapatkan parameter-parameter (bobot dan bias) baru, maka dilakukan perhitungan kembali proses *forward propagation*. Pada iterasi (*epoch*) kedua model melakukan proses konvolusi kembali menggunakan data yang sama namun dengan *filter* (kernel) yang berbeda. Pada iterasi (*epoch*) kedua ini didapati hasil probabilitas pada masing-masing kelas. Kelas tidak segar (x_1) mendapatkan nilai probabilitas 0,000002, sedangkan pada kelas segar (x_2) mendapatkan nilai 0,999998.

Dalam iterasi (*epoch*) kedua, model mampu memprediksi kelas yang benar dan mendapatkan nilai *loss function* yaitu sebesar 0,0000009. Setelah model berhasil memprediksi kelas yang benar dan nilai *loss function* dapat dikatakan sudah mencapai minimum, maka proses iterasi (*epoch*) berhenti.

Seluruh perhitungan di atas merupakan proses satu kali iterasi (*epoch*) *forward propagtion* dan *backward propagation* pada satu data. Proses ini akan terus mengulang hingga model mampu memprediksi kelas yang benar sehingga mencapai nilai *loss* yang minimum (Wulandari, 2019).

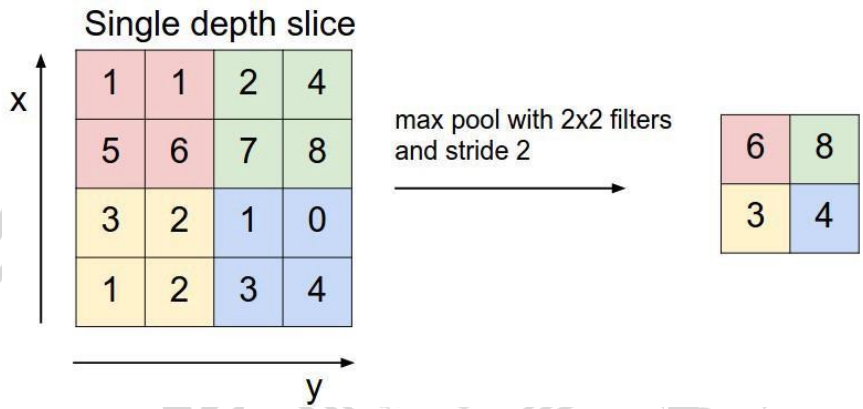
Tabel 2.1 Hasil Pelatihan CNN Sederhana

Epoch	Probability		Loss	Predictions
	$\hat{y}_1$	$\hat{y}_2$		
1	0,731053	0,268947	0,570333	$\hat{y}_1$
2	0,000002	0,999998	0,0000009	$\hat{y}_2$

2.3.3 Pooling Layer

Pooling layer merupakan bagian dari CNN dimana metode ini digunakan setelah *convolution layer*. Pada *pooling layer* yang menerima output dari *convolutional layer*, pada layer ini ukuran citra akan diperkecil secara spasial dengan mengurangi jumlah parameter. Menurut (Nurfita dan Ariyanto, 2018) yaitu *pooling layer* disederhanakan menjadi *feature maps* yang dihasilkan oleh layer konvolusi. Penyederhanaan ini memungkinkan untuk memulihkan informasi penting dari hasil konvolusi layer atau yang bisa disebut *feature maps* dan memilih detail yang tidak lagi digunakan sehingga komputasi layer berikutnya lebih ringan.

Adapun dua jenis *pooling layer*, yaitu *average pooling* dan *max pooling*. Salah satu yang sering digunakan atau umum digunakan untuk *pooling* pada CNN adalah *max pooling*, dimana metode ini mengambil nilai yang paling maksimal dari setiap masing-masing *grid*, nilai maksimum dari setiap *grid* untuk membuat feature maps yang disederhanakan. Pada Gambar 2.7, grid dengan warna hijau, kuning, merah, dan biru adalah kelompok grid yang dipilih dari nilai maksimumnya. Hasil pengolahan ini dapat dilihat pada Gambar 2.7, himpunan grid di sebelah kanan.

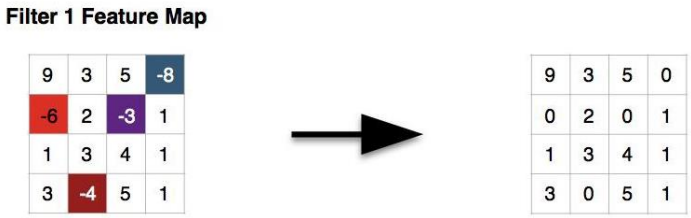


Gambar 2.7 Ilustrasi *Max Pooling*

(sumber gambar : https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94**)**

2.3.4 Aktivasi ReLU

Aktivasi Rectified Linear Unit (ReLU) adalah lapisan aktivasi yang umum digunakan dalam model CNN. Kelas aktivasi ini merupakan lapisan aktivasi pada model CNN yang menerapkan fungsi (x) = max(0, x) yang artinya nilai keluaran dari neuron dinyatakan 0 jika nilai masukannya negatif, sedangkan jika nilai masukannya adalah positif, maka nilai keluarannya dalah nilai dari masukkannya sendiri (Ramachandran, zoph, dan Le, 2017). Dapat dilihat pada ilustrasi berikut ini.



Gambar 2.8 Ilustrasi ReLu

(sumber gambar : https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94**)**

2.3.5 Aktifasi *Softmax*

Aktivasi *Softmax* alias *Softmax Classifier* merupakan salah satu fungsi aktivasi yang digunakan di CNN. Fungsi aktivasi softmax ini digunakan pada lapisan keluaran atau *output layer*. Fungsi *softmax* pada dasarnya adalah probabilitas eksponensial yang dinormalisasi dari pengamatan kelas yang dinyatakan sebagai aktivasi neuron. Fungsi eksponensial adalah untuk meningkatkan probabilitas nilai terbesar atau terbesar dari kelas sebelumnya. Persamaan dari fungsi *softmax* sebagai berikut :

$$f_i(X) = \frac{e^{xi}}{\sum_k e^{xk}} \qquad (2.11)$$

Fi menunjukkan hasil fungsi dari setiap elemen I dalam vektor yang diturunkan dari kelas X, sebuah hipotesis yang menyediakan model pelatihan yang diklasifikasikan oleh fungsi *softmax*. *Softmax* memungkinkan untuk menghitung probabilitas untuk semua label, dari label ini akan diambil sebuah vektor dengan nilai real dan akan diubah menjadi vektor dengan nilai 0 dan 1 dan ketika semuanya ditambahkan akan menjadi satu.

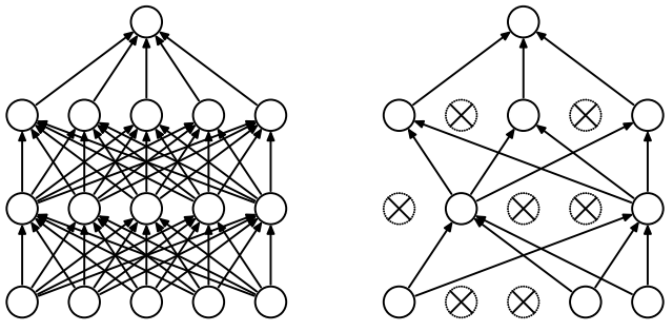
2.3.6 Fully-Connected Layer

Feature extraction layer menghasilkan *feature map* yang selalu berupa array multidimensi. Oleh karena itu, yang perlu dilakukan adalah penyelarasan *"flatten"* atau yang biasa disebut dengan *reshape feature map* pada saat modifikasi vektor yang dapat diolah atau digunakan sebagai input data dari *fully-connected layer* sebelum tidak dapat diproses.

Fully connected layer merupakan lapisan terakhir dari CNN dan sering digunakan dalam implementasi *Multi Layer Perceptron* (MLP) untuk memproses data dengan cara yang dapat diklasifikasikan secara linier. Seperti halnya jaringan syaraf tiruan, setiap neuron yang ada pada lapisan sebelumnya dihubungkan ke setiap neuron pada lapisan berikutnya oleh lapisan *fully-connected*. (Anugerah, 2018).

2.3.7 Dropout Regularization

Dropout adalah metode pelatihan jaringan saraf dengan cara memilih neuron secara acak dan kemudian neuron yang dipilih tidak akan digunakan selama pelatihan. Artinya, neuron yang dipilih atau tidak dipilih akan dijatuhkan secara acak. Ini berarti bahwa selama pembelajaran neuron yang dibuang akan dihentikan sementara dan selama backpropagation neuron ini tidak akan diterapkan pada bobot baru (Baldi dan sadowski, 2013).



Gambar 2.9 Contoh Implementasi *Dropout*

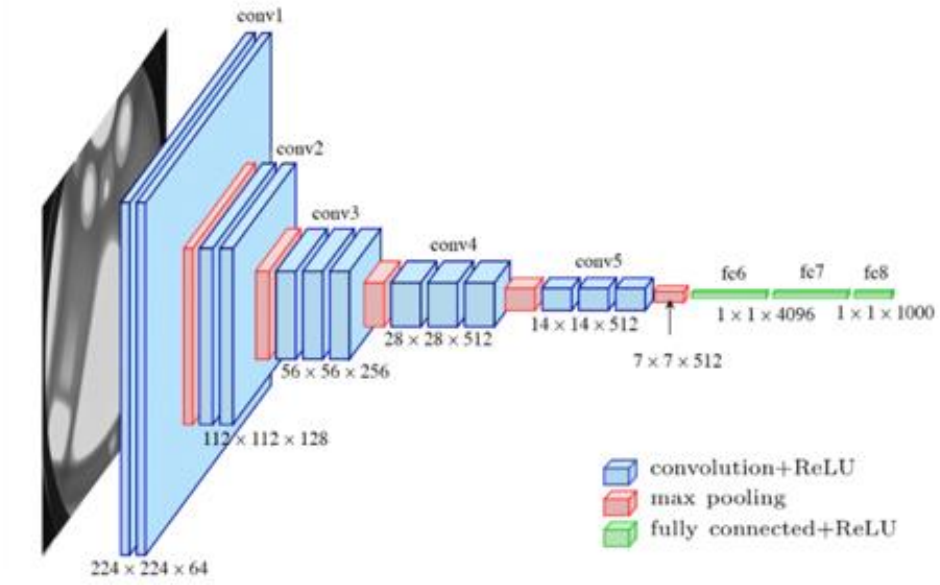
(sumber gambar *:* https://idanovinda.medium.com/mengapa-diperlukan-regularisasi-pada-model-neural-network-d622ed98f9a8*)*

Pada Gambar 2.10, jaringan saraf (a) adalah jaringan saraf biasa dengan dua *hidden layer*. Sedangkan jaringan saraf (b) merupakan jaringan saraf yang menerapkan teknik *regularization dropout* dan terdapat neuron yang sudah tidak digunakan lagi.

2.4 Arsitektur VGG-16

CNN biasanya digunakan untuk memperkenalkan objek, seperti gambar, objek, atau pemandangan, dan CNN juga dapat mendeteksi dan mengelompokkan objek. CNN dapat dilatih secara langsung melalui data citra tanpa ekstraksi manual. Penemuan pertama dari Hubel dan Wiesel adalah mempelajari persepsi visual dari *visual cortex* kucing. Studi tentang *visual cortex* hewan sangat kuat dalam hal sistem pemrosesan visual yang pernah ada. Banyak penelitian didasarkan pada prinsip-prinsip kerja dan menciptakan model-model baru.

VGGNet merupakan salah satu arsitektur CNN yang dikembangkan oleh K.Simonyan dan A. Zisserman. VGG (*Visual Geometry Grup*) merupakan sebuah kelompok peneliti dari Universitas Oxford yang kemudian membuat sebuah arsitektur CNN yang dinamai dengan *VGGNet* (Wu, 2017). Penemuannya pada 2014 dalam ImageNet Challenge 2014 mendapatkan juara pertama dan kedua. Jaringan *VGGNet* terbaik memiliki 16 *convoluiton layer* beserta *fully connected layer* dengan hanya melakukan konvolusi 3×3 dan pooling 2×2 dari lapisan awal hingga lapisan akhir (Simonyan dan Zisserman, 2014). Alasan menggunakan konvolusi 3×3 yaitu dapat membuat fungsi keputusan lebih disktriminatif dan dapat mengurangi jumlah parameter.



Gambar 2.10 Arsitektur VGG-16

(sumber gambar : https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94***)***

Dalam gambar arsitektur VGG16 memiliki desain minimal 16 lapisan, 13 lapisan convolutional dan 3 lapisan ully connected layer. Arsitektur VGG16 menangani 5 konvolusi di mana setiap proses memiliki jumlah filter yang berbeda seperti yang ditunjukkan pada Gambar 2.6. Arsitektur ini memulai dengan filter yang sangat rendah yaitu 64 filter pada lapisan block pertama dan kemudian meningkat dua kali lipat pada setiap kali proses konvolusi, hingga mencapai 512 filter (Simonyan dan Zisserman, 2014). Meskipun lapisan pada satu block mempunyai filter yang sama, penumpukan lapisan konvolusi tersebut memungkinkan dapat menguraikan masukan secara berurutan.

Arsitektur VGG mempunyai beberapa fitur yaitu (Siomonyan dan Zisserman, 2014):

1. *Input* layer : Menerima masukan gambar dengan ukuran 224×224.
2. *Convolution Layer* : Masukan gambar dikonvolusi menggunakan kernel yang berukuran 3×3, kemudian *stride* 1, dan *zero padding*, sehingga ukuran *input* dan *output* tetap sama atau dengan kata lain resolusi setelah konvolusi tetap seperti masukan.

3. *Max pooling* : *Max pooling* menggunakan *pool size* 2×2 dan *stride* 2.
4. Tidak semua operasi konvolusi dilakukan *max pooling*.
5. *Fully Connected Layer* pertama dan kedua mempunyai 4096 filter menggunakan *dropout regularization* dengan *rate* 0.5 dan *fully connected layer* yang ketiga mempunyai 1000 filter.
6. Semua operasi konvolusi dan *fully connected layer* menggunakan fungsi aktivasi Relu.