BAB II
TINJAUAN PUSTAKA

2.1.Penelitian Terdahulu

Dengan penelitian yang dilakukan oleh penulis, sangat diperlukan adanya referensi dasar dari penelitian terdahulu. Dari adanya referensi tersebut dapat membantu penulis untuk melalukan penelitian yang berkaitan dan menghindari terjadinya duplikasi pada penelitian. Sehingga penelitian yang dilakukan oleh penulis dapat berkontribusi untuk pengembangan dalam bidang ilmu pengetahuan.

Pada penelitian yang telah dilakukan berjudul *Tire Defect Detection Using Fully Convolutinal Neural Network.* Penelitian ini menggunakan metode berdasarkan *Fully Convolutional Neural Network* (FCN) untuk mendeteksi kecacatan pada gambar sinar – X ban. Karena kemampuan prediksi yang baik dari piksel FCN, lokasi, dan segmentasi kecacatan diselesaikan secara bersamaan. Arsitektur yang digunakan dalam metode ini terutama terdiri dari tiga fase. Fase pertama yaitu *Deep Network,* yang digunakan untuk mengestrak fitur ban. Dengan mengganti lapisan yang terhubung sepenuhnya menjadi lapisan konvolusi. Pada tahap kedua, dan fitur *maps* mempertahankan informasi spasial yang memadai, dengan menambahkan lapisan *up-sampling,* pada tahap kedua, output dengan ukuran yang sama dengan gambar asli dapat dihasilkan. Setelah dua fase pertama, hasil akan dikembangkan dari segmentasi kasar dan menyempurnakannya melalui penggabungan fitur *maps* multi skala. Pada fase pertama arsitektur menggunakan VGG16 klasik. Ini terdiri dari aplikasi berulang dari tumpukan 3x3 lapisan kovolusi dan 2x2 pooling layer, masing- masing *convolution layer* diikuti *rectified linear unit (ReLU)* untuk *non-linearity rectification.* Kemudian lapisan yang terhubung penuh di VGG16 dililitkan sehingga fitur *maps* yang dihasilkan dapat menyimpan informasi spasial yang lengkap. Dataset yang digunakan terdiri dari 914 gambar ban, diantaranya 700 gambar dipilih secara acak sebagai *training* dan 214 sisanya digunakan untuk pengujian. Gambar – gambar tersebut melibatkan berbagai cacat seperti kotoran logam, gelembung, dan tumpang tindih.

Hasil model klasfikasi yang dihasilkan mencapai akurasi 78,91% menggunakan arsitektur VGG16 dengan 3 *fusion layer*(Wang et al. 2019).

Penelitian berjudul *Tire Defects Classifcation with Multi-Contrast Convolutional Neural Networks,* yang pertama dengan normalisasi kontras gambar dan augmentasi data digunakan untuk menghindari masalah jaringan yang berlebihan dengan sejumlah besar parameter. Selanjutnya, CNN multi-kolom diusulkan dengan menggabungkan beberapa CNN yang dilatih pada data praproses yang berbeda menjadi CNN multi-kolom (MC-CNN), dan kemudian prediksi mereka dirata-ratakan sebagai output dari jaringan yang diusulkan. Dari dataset yang berjumlah 1582 gambar kecacatan ban dan gambar *rontgen* yang mendapatkan akurasi rata – rata 98,47%(Cui et al. 2018).

Dari referensi penelitian sebelumnya penulis menggunakan metode CNN dengan beberapa lapisan konvolusi dan hidden layer yang berbeda yang sudah dijelaskan di bagian latar belakang. Tujuan dari penelitian ini adalah mengetahui bagaimana algoritma *Deep Learning* dapat mengenali dan mengklasifikasikan citra ban berdasarkan kelas yang ditentukan

2.2.Kecacatan Ban

Kendaraan bermotor merupakan suatu yang penting untuk seluruh masyarkat baik dari golongan menengah keatas maupun golongan menengah kebawah, dikarenakan kendaraan sangat berpengaruh dalam melakukan perpindahan antara satu tempat ke tempat lain. Dari perpindahan tempat tersebut kendaraan harus dibekali dengan kecepatan dan keselamatan saat berkendara. Tetapi dalam hal kecepatan itu biasanya terdapat banyak kecelakaan kendaraan yang dikarenakan kurang siapnya alat kendaraan yang dipakai tersebut mulai dari mesin maupun ban yang dipakai.

Dalam suatu produksi ban, kecacatan ban sangat berpengaruh untuk kualitas ban itu secara langsung menentukan masa pakai ban tersebut. Ban dalam segi produksi otomatis secara cepat juga mempengaruhi dari kualitas ban tersebut. Dan itu juga sangat mempengaruhi dari segi biaya dan waktu yang dibutuhkan untuk memproduksi

ban. Jika kecacatan ban terdeteksi secara cepat, itu akan sangat membantu meningkatkan kualitas produk ban(Zhu and Ai 2018). Untuk ban yang normal, dapat ditunjukkan pada Gambar 2.1

Gambar 2. 1 Ban Normal

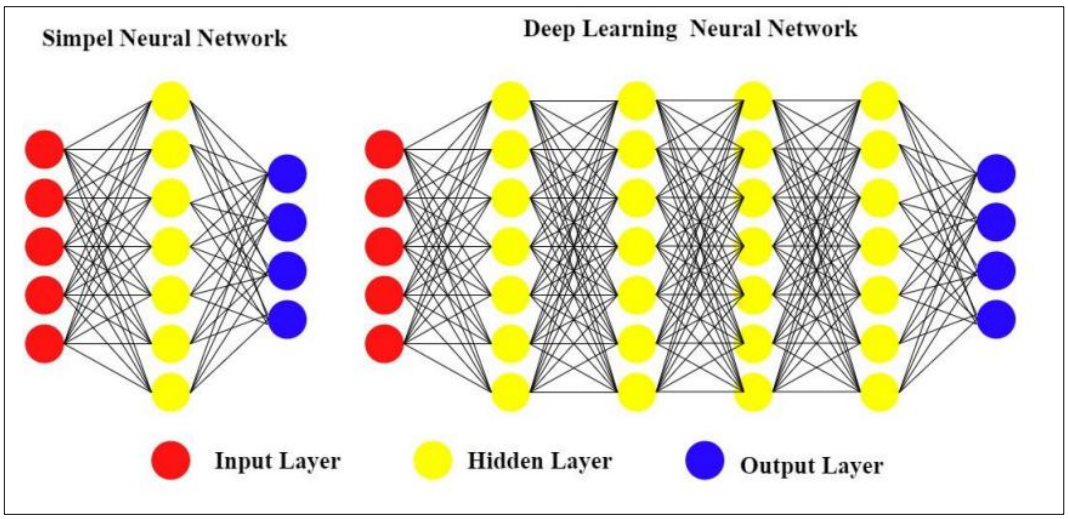
Pada **Gambar 2.1** merupakan ban normal, ciri – ciri dari ban normal memiliki permukaan yang masih halus dan tidak ada keretakan pada ban

Gambar 2. 2 Ban Retak

Pada **Gambar 2.2** merupakan gambar ban retak (*cracked*), ciri – ciri dari ban retak memiliki permukaan tidak halus lagi dan keliatan keretakan pada ban.

2.3.Deep Learning

Deep Learning merupakan sub bidang dari pengembangan mechine learning yang mengajarkan komputer dapat membuat keputusan akurat tanpa bantuan manusia (Grossfeld, 2020). Algoritma Deep Learning yang tertanam pada sebuah komputer dapat belajar mengklasifikasi secara langsung dari sebuah data berupa gambar, suara, teks, atau video (Lecun 2017).



Gambar 2. 3 Deep Learning

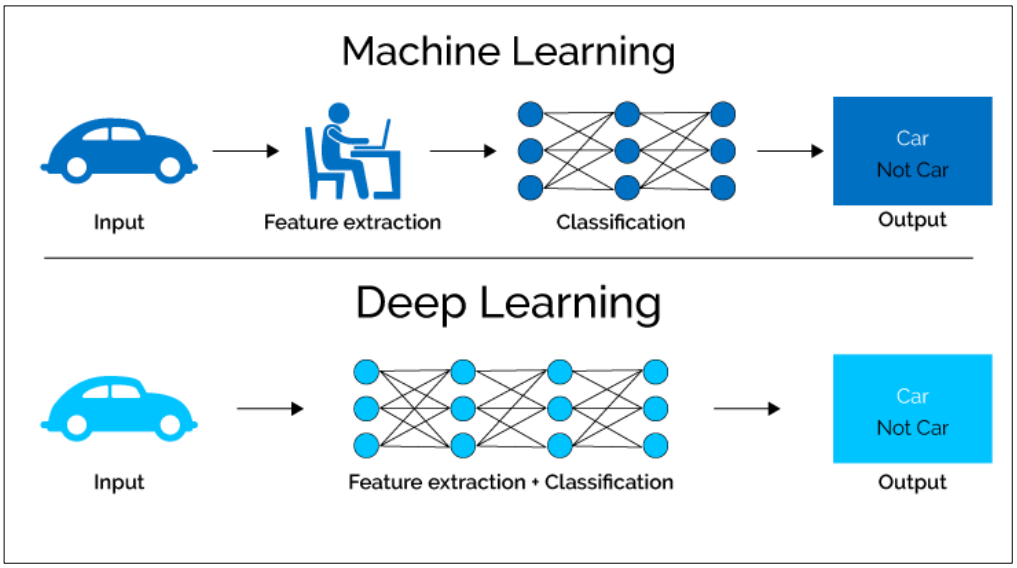
Penjelasan pada **Gambar 2.**3 yaitu perbedaan dan persamaan arsitektur dari *Simpel Neural Network* dan *Deep Learning Neural Network.* Jika dari persamaannya selalu dimulai dari *Input Layer* selanjutnya diikuti dengan *Hidden Layer* yang bagian terakhir yaitu *Output Layer*. Dan dari perbedaan nya sangat terlihat jelas untuk *Simpel Neural Network* hanya mempunyai beberapa *Hidden Layer* sedangkan *Deep Learning Neural Netwok* mempunyai banyak *Hidden Layer*.

1. *Input Layer* : Menerima nilai input
2. Hidden Layer : Satu set neuron antara lapisan input dan output. Bisa ada satu atau beberapa lapisan
3. Output Layer : Biasanya memiliki satu neuron, dan keluarannya berkisar antara 0 dan 1, yaitu lebih besar dari 0 dan lebih kecil dari 1. Tetapi beberapa output juga dapat hadir.

Kemampuan pemrosesan disimpan dalam kekuatan koneksi antar unit, yang disebut bobot. Kekuatan masukan tergantung pada nilai berat. Nilai bobot bisa positif, negatif atau nol. Bobot negatif berarti sinyal berkurang atau terhambat. Bobot nol berarti tidak ada hubungan antara kedua neuron. Bobotnya disesuaikan dengan memperoleh keluaran yang dibutuhkan

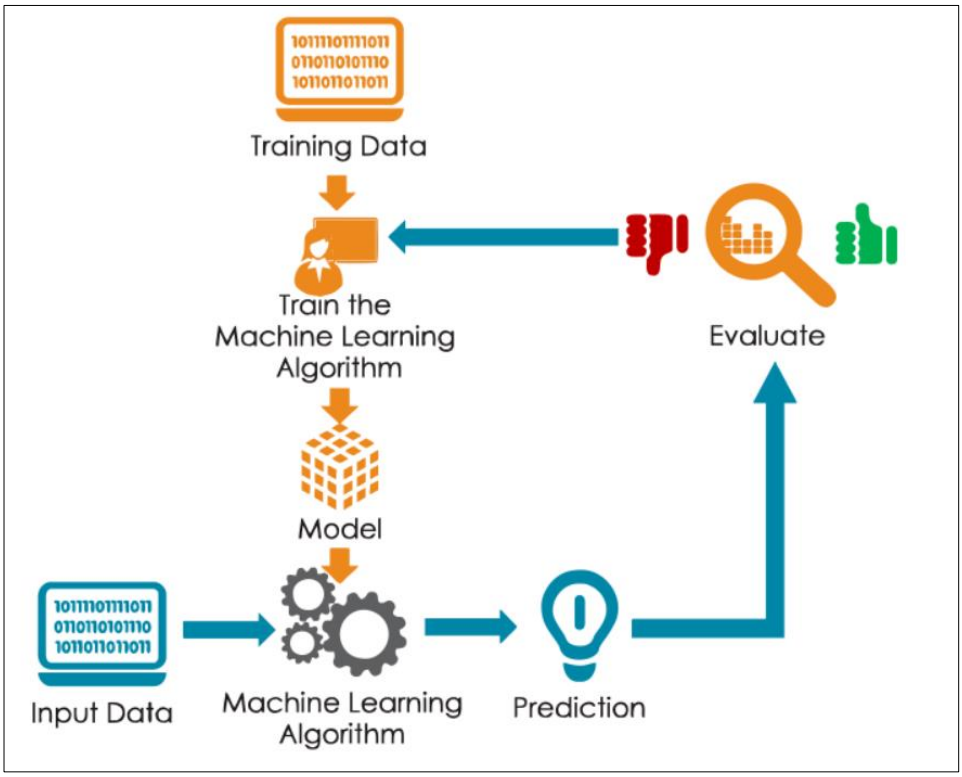
2.4.Machine Learning

Pembelajaran mesin adalah aplikasi *Artificial Intelligence* (AI) yang memungkinkan sistem untuk belajar dan meningkatkan dari pengalaman tanpa diprogram secara eksplisit. Pembelajaran mesin berfokus pada pengembangan program komputer yang dapat mengakses data dan menggunakannya untuk belajar sendiri.



Gambar 2. 4 Perbedaan Machine Learning dan Deep Learning

Penjelasan pada Gambar 2.4 diatas merupakan perbedaan dari *Machine Learning* dan *Deep Learning* itu sendiri. Dari gambar diatas dapat dilihat bawah perbedaan terdapat pada performanya ketika jumlah data terus meningkat dan bagaimana menyelesaikan suatu masalah. *Deep Learning* digunakan untuk membuat jaringan syaraf buatan yang tidak mampu mengolah data dalam jumlah kecil secara maksimal(Rezkia 2021).



Gambar 2. 5 Cara Kerja Machine Learning

Pada **Gambar 2.5** merupakan cara kerja *Machine Learning.* Komputer akan melakukan proses belajar (*training*) untuk menghasilkan suatu model. Proses belajar ini menggunakan algoritma *machine learning* sebagai penerapan teknik statistika. Model inilah yang menghasilkan informasi, kemudian dapat dijadikan pengetahuan untuk memecahkan suatu permasalahan sebagai proses *input-output*. Model yang dihasilkan dapat melakukan klasifikasi ataupun prediksi ke depan.(4)

Untuk memastikan efisiensi model yang terbentuk, data akan dibagi menjadi data pembelajaran (*train* dataset) dan data pengujian (*test* dataset). Pembagian data yang digunakan bervariasi bergantung algoritma yang digunakan.(4)

2.5.Pengolahan Citra

Pengolahan citra digital adalah teknik mengolah citra yang bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin

komputer yang dapat berupa foto maupun gambar bergerak. Pengolahan citra digital memiliki beberapa kelebihan, yaitu murah, cepat, dan tidak merusak sampai yang diukur dan mampu mengidentifikasi fisik produk secara obyektif(Effendi, Fitriyah, and Effendi 2017).

Secara harfiah, citra (*image*) adalah gambar pada bidang dwimatra (dua-dimensi). Ditinjau dari sudut pandangan matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian berkas cahaya tersebut, pantulan cahaya ini ditangkap oleh alat optik sehingga bayangan objek yang disebut citra tersebut terekam

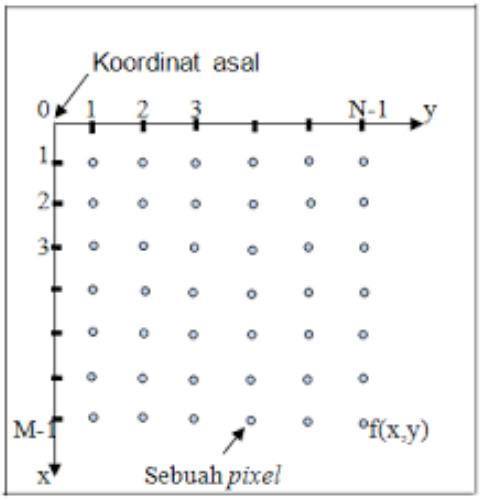
Sebelum masuk ke pengolahan citra, terlebih dahulu melakukan *input* citra. Selanjutnya *input* citra diubah menjadi bentuk matriks, kemudian pengolahan citra dilakukan, setelah dilakukan pengolahan citra maka akan diperoleh *output* citra. Pengolahan citra sendiri memiliki beberapa teknik seperti *filtering*, transformasi, segmentasi, *masking*, dan lain-lain

2.6.Pencitraan Digital

Pencitraan digital merupakan disiplin ilmu yang mengkaji perihal bagaimana teknik komputasi dari pengolahan sebuah citra. Citra disini merupakan sebuah gambar diam (foto) dan gambar bergerak (video). Sedangkan digital disini adalah pengolahan citra yang dilakukan menggunakan komputer (Sutoyo dkk., 2009). Citra digital dipresentasikan dengan matriks , sehingga pengolahan pada citra digital pada dasarnya memanipulasi elemen-elemen matriks yang berupa piksel yang terdapat pada sebuah larik (Studi et al. 2016).

Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun *complex* yang di representasikan dengan deretan bit tertentu. Pengolah citra digital menunjuk pada proses gambar 2 dimensi menggunakan komputer. Citra digital dapat di definisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom dengan x dan y adalah koordinat spasial dan amplitudo f di titik koordinat(x,y) dinamakan intensitas

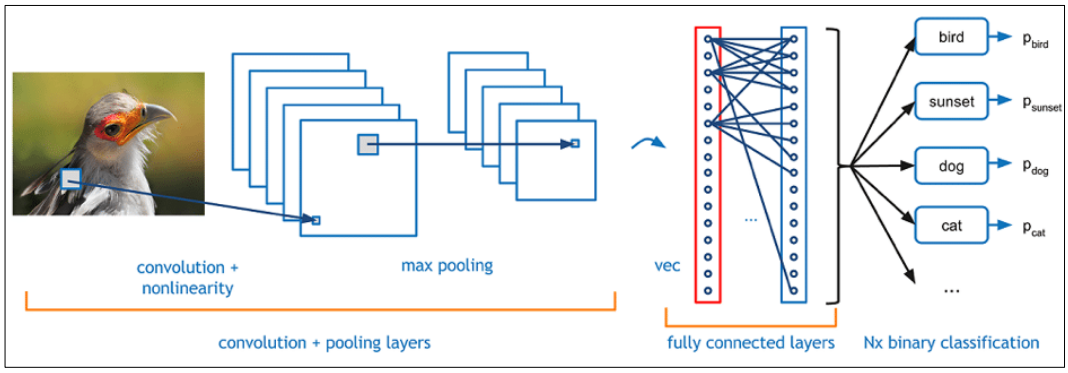
atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x,y, dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Koordinat citra digital dapat dilihat pada gambar 2.6



Gambar 2. 6 Kordinat Citra Digital

2.7.Convolutional Neural Network

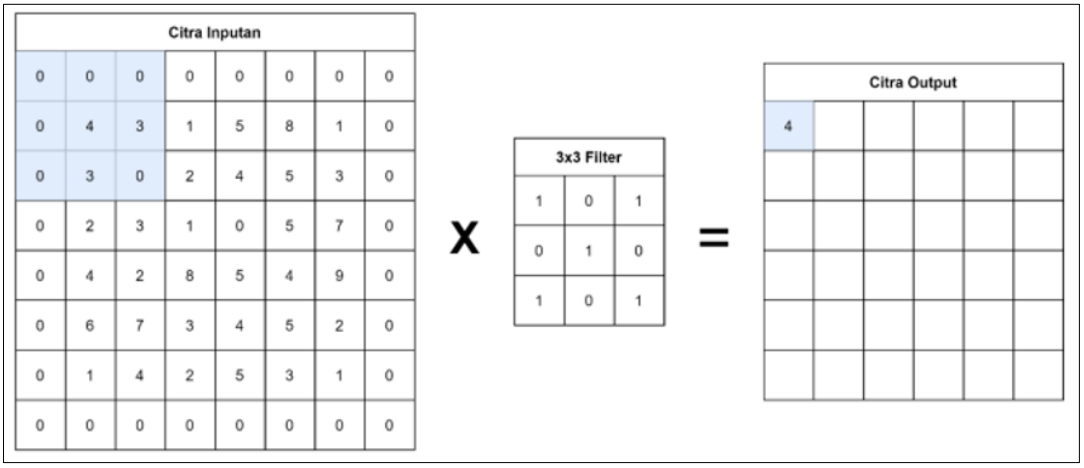
Convolutional Neural Network adalah salah satu jenis perhitungan neural network yang sering digunakan pada pengolahan citra untuk mendeteksi dan mengenali objek pada sebuah image (Mehindra Prasmatio, Rahmat, and Yuniar 2020). *Convolution Layer* (Lapisan konvolusi) merupakan proses menerapkan filter pada gambar. Pada proses konvolusi terdapat perkalian matriks antara filter atau kernel dan area pada citra. Pada arsitektur jaringan CNN bisa memiliki lebih dari satu lapisan *convolution layer*. Lapisan *convolutional* pertama bertanggung jawab untuk mendapatkan informasi dari gambar yang memuat fitur tingkat rendah (*low level features*) seperti tepi horizontal, tepi vertikal, warna, orientasi gradien. Pada lapisan selanjutnya akan bertanggung jawab dengan fitur tingkat tinggi (*high level features*) seperti ketajaman, tekstur, dan lainnya(Budiarto Hadiprakoso and Buana 2021).



Gambar 2. 7 Arsitektur Convolutional Neural Network

Penjelasan pada gambar 2.7 merupakan struktur sederhana dari arstiektur *Convolutional Neural Network.* Pada gambar tersebut menunjukkan dari tiga proses yaitu dari *convolution + pooling layers* lalu ke *fully connected layers* dan proses terakhir yaitu *classification* yang menunjukkan output nya.

Lalu untuk hasil dari operasi perkalian matriks tersebut biasa disebut dengan *feature map*. Proses operasi *convolution* dapat dilihat pada gambar 2.8.



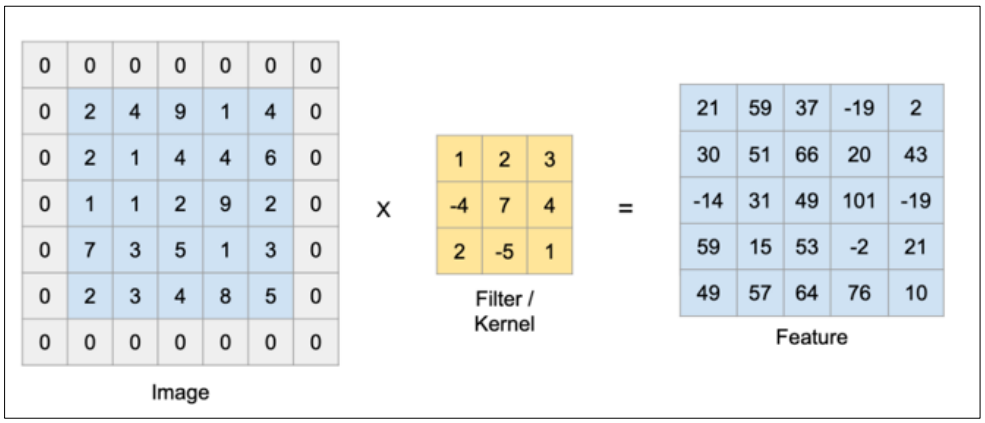
Gambar 2. 8 Proses *Convolution Layer* (Fachrurrozi, 2021)

Pada gambar 2.8 merupakan proses *convolution* pada sebuah matriks gambar, Operasi *convolution* sendiri merupakan perkalian antara matriks input dan filter atau kernel yang akan menghasilkan sebuah *feature map* (Jason Brownlee 2020). Pada gambar 2.8 terdapat matriks input dengan ukuran 8 x 8 dan matriks filter 3 x 3. Dari perkalian kedua matriks tersebut menghasilkan output berupa matriks *feature map*

dengan ukuran 6 x 6 karena hilangnya lapisan terluar dari input akibat proses *convolution.*

2.7.1 Padding dan Stride

Padding adalah penambahan ukuran piksel dengan nilai tertentu pada lapisan terluar dari data input agar hasil dari feature map tidak terlalu kecil sehingga tidak banyak informasi yang hilang. Tanpa padding, sangat sedikit nilai pada lapisan berikutnya yang akan terpengaruh oleh piksel sebagai tepi gambar. Nilai padding biasanya nol sehingga disebut dengan zero padding. Ilustrasi padding dapat dilihat pada gambar 2.9.

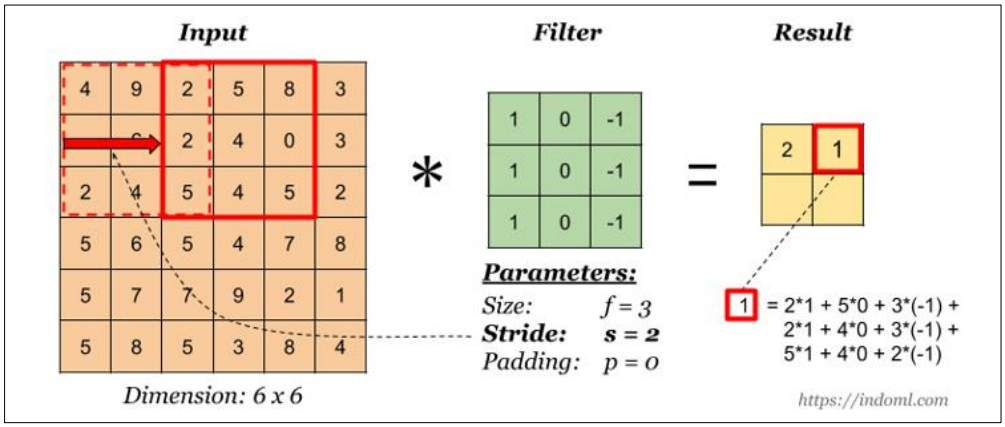


Gambar 2. 9 Ilustrasi Padding

Pada gambar 2.9 terdapat *input* citra dengan ukuran 5 x 5 piksel yang kemudian diberi padding dengan zero padding pada lapisan terluar yang membuat ukuran dari citra tersebut menjadi 7 x 7 piksel. *Input* citra tersebut akan melalui proses *convolution* oleh filter atau kernel yang memiliki ukuran 3 x 3 piksel dan menghasilkan *feature map* dengan ukuran 5 x 5 piksel. Dari gambar tersebut dapat diketahui fungsi dari padding adalah agar *output* atau hasil pada *feature map* memiliki ukuran yang sama dengan citra pada *input.*

Stride adalah parameter yang menentukan berapa jumlah pergeseran filter atau kernel. Jika nilai stride adalah 1, maka pada proses *convolution* filter akan bergeser sebanyak 1 piksel secara horizontal lalu vertikal. Semakin kecil stride maka akan semakin detail informasi yang kita dapatkan dari sebuah input, namun membutuhkan

komputasi yang lebih jika dibandingkan dengan stride yang besar. Menggunakan stride yang kecil kita tidak selalu akan mendapatkan performa yang bagus. Ilustrasi *stride* ditunjukkan oleh gambar 2.10



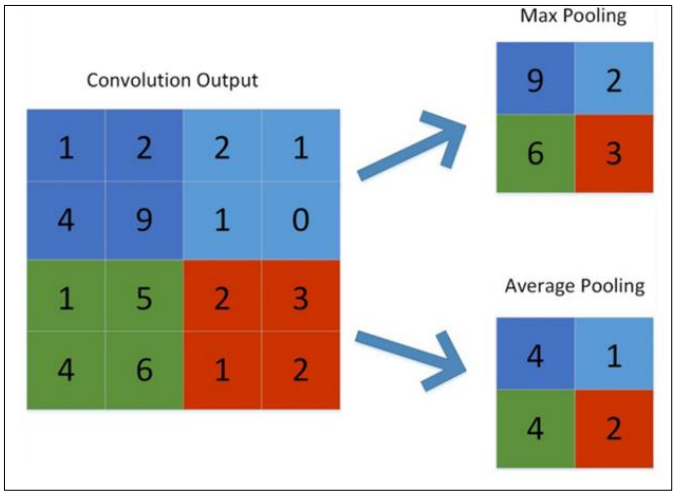
Gambar 2. 10 Ilustrasi Stride

Pada gambar 2.10 dapat dilihat proses operasi *convolution* perkalian antara input citra dengan filter memiliki pergeseran 2 elemen matriks dari input citra

2.7.2 Pooling Layer

Pooling layer merupakan lapisan yang berfungsi untuk mengurangi ukuran matriks dengan mengambil elemen yang dinilai merupakan bagian penting dari fitur konvolusi sehingga dapat mengurangi sumber daya komputasi yang dibutuhkan untuk memproses data melalui pengurangan dimensi dari *features map* (*down sampling*) sehingga mempercepat komputasi karena parameter yang diperbaharui semakin sedikit.

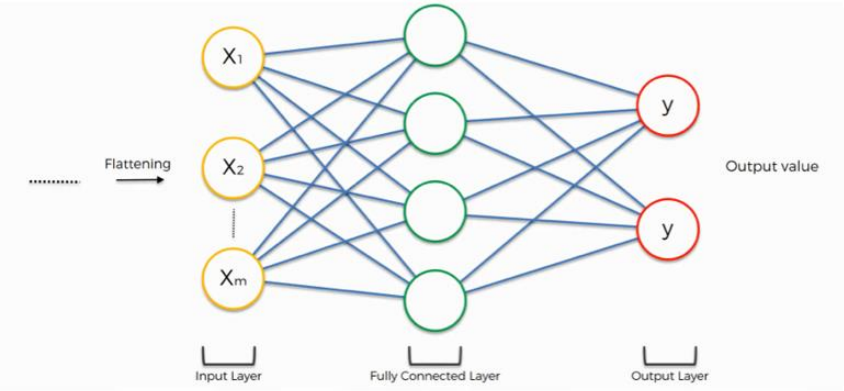
Ada dua jenis tipe dari *pooling* yaitu *max pooling* dan *average pooling*. *Max pooling* akan meringkas data masukan atau *feature map* dengan mencari nilai atau *value* terbesar dari *feature map* berdasarkan penggeseran *window pooling*. *Average pooling* akan menghitung nilai rata-rata dari setiap *feature map* pada setiap kali penggeseran *window pooling*(Kholik 2021). Proses *pooling layer* dapat dilihat pada gambar 2.11



Gambar 2. 11 Ilustrasi Pooling (Jiang, Lu, and Wang 2020)

2.7.3 Fully Connected Layer

Lapisan *fully connected* layer adalah lapisan di mana semua input dari satu lapisan terhubung ke setiap unit aktivasi lapisan berikutnya. Lapisan ini mendapatkan input dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua node menjadi satu dimensi (Romario, Ihsanto, and Kadarina 2020). Layer ini mendapatkan input (convolution, ReLU, atau pooling layer) dan menghasilkan vector dimensi NxN adalah jumlah kelasnya. Tujuannya adalah menggunakan fitur-fitur tersebut untuk mengklasifikasi gambar ke dalam berbagai kelas berdasarkan inisialisasi pada data training. Ilustrasi *fully connected layer* dapat dilihat pada gambar 2.12



Gambar 2. 12 Fully Connected Layer

2.7.4 Batch Normalisazation

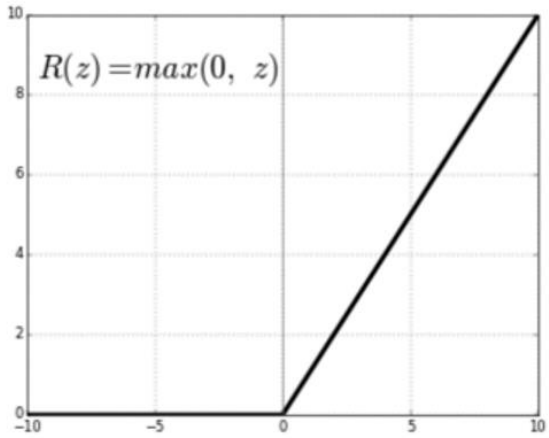
Normalisasi adalah prosedur dalam mengubah nilai numerik dalam *dataset* untuk menjadikan skala tertentu tanpa menghapus informasi yang ada pada *dataset* tersebut (Chai, Pilanci, and Murmann 2020). Namun pada *machine learning* Teknik dalam melakukan normalisasi *input* tiap lapisan saat proses pelatihan model disebut *batch normalization* (Peccia, 2018). Penggunaan *batch normalization* saat proses pelatihan model dapat menghasilkan *output* yang bagus tanpa menggunakan banyak nilai pada *epoch*. Ini karena pada lapisan tingkat menengah pada model, distribusi nilai dari fungsi aktivasi sering berubah sehingga mengakibatkan proses pelatihan menjadi lama karena pada tiap lapisan model harus mengaplikasikan nilai distribusi baru pada tiap *epoch.*

2.7.5 Fungsi Aktivasi

Fungsi aktivasi merupakan persamaan matematis yang digunakan sebagai penentu nilai output dari suatu neuron pada neural network (Dinesh Kumawat 2019). Tiap neuron akan dilakukan normalisasi output dengan skala antara 1 dan 0 atau antara -1 dan 1. Fungsi aktivasi memiliki banyak jenis, namun yang sering digunakan dalam pemrosesan yaitu *non linear functions*, karena memungkinkan model melakukan kemampuan operasi dalam penanganan yang kompleks pada input dan output suatu jaringan yang besar (Bhushan Kapkar 2020). Terdapat beberapa fungsi aktivasi n*on linear functions*:

1. Rectified Linear Unit (ReLU)

Dalam fungsi aktivasi ReLU, semua masukan atau *input* yang bernilai kurang dari nol, maka luaran atau *output* yang dihasilkan bernilai nol, dan jika nilai masukan atau *input* bernilai lebih dari nol, maka luaran atau *output* yang dihasilkan akan bernilai sama dengan nilai masukan atau *input*. Grafik fungsi aktivasi ReLU terdapat pada gambar 2.13



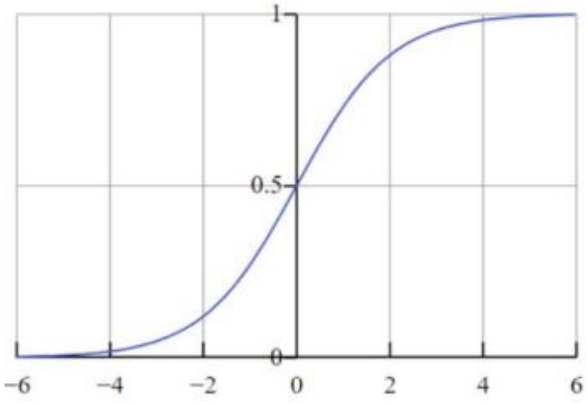
Gambar 2. 13 Grafik Aktivasi ReLU

Berdasarkan gambar 2.13, berikut merupakan rumus persamaan dari fungsi aktivasi ReLU:

$$f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}$$

2. Sigmoid

Sigmoid merupakan salah satu fungsi aktivasi yang dipakai untuk pembelajaran mesin paling utama pada aspek logistic regression, peranan aktivasi ini kerap dipakai untuk identifikasi pembelajaran mesin, akan tetapi tidak sering digunakan untuk model pembelajaran mesin pada tingkat lanjut. Grafik fungsi aktivasi sigmoid dapat dilihat pada Gambar 2.14



Gambar 2. 14 Grafik Aktivasi Sigqmoid

Berikut rumus persamaan dari fungsi aktivasi sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$

3. Softmax

Fungsi Softmax merupakan fungsi yang mengganti nilai riil K menjadi vector nilai rill K yang berjumlah 1. Nilai masukan dapat berupa nilai positif, negatif, nol, atau lebih besar dari satu. Namun fungsi Softmax mengubahnya jadi angka antara 0 dan 1, alhasil bisa dimaksud sebagai probabilitas. Apabila salah satu masukan kecil (negatif) fungsi Softmax mengubahnya jadi probabilitas kecil dan jika masukan besar (positif) fungsi Softmax mengubahnya menjadi probabilitas besar, namun nilai bakal selalu antara 0 dan 1 (Wood n.d.).

$$\sigma(\vec{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Keterangan:

$\vec{z}$ = input vector untuk fungsi softmax, mulai dari z0 hingga zK

z_i= semua nilai z_i dari input vector untuk fungsi softmax

e^{z_i}= standar fungsi eksponensial diaplikasikan ke tiap elemen dari input vector

$\sum_{j=1}^{K} e^{z_j}$= normalisasi agar nilai keluaran jika ditotal menghasilkan 1 atau 0

2.7.6 Optimization Function

Optimization Function ataupun fungsi optimizer adalah fungsi yang dipakai guna untuk memperbarui parameter dari bobot, ini memiliki tujuan agar meminimalisir angka loss yang diterima (Renu Khandelwal 2019). Fungsi ini masih berhubungan dekat dengan fungsi loss. Hal itu disebabkan optimizer memerlukan optimasi pada fungsi loss supaya memperoleh hasil terbaik (Renu Khandelwal 2019). Dalam deep learning terdapat berberapa optimizer, yang kerap dipakai antara lain:

1. Adaptive Movement Estimation (Adam)

Adam adalah algoritma optimasi yang dapat digunakan sebagai pengganti prosedur penurunan gradien stokastik klasik untuk memperbarui bobot jaringan secara iteratif berdasarkan data pelatihan. Adam dipresentasikan oleh Diederik Kingma dari

OpenAI dan Jimmy Ba dari University of Toronto dalam makalah (poster) ICLR 2015 berjudul " Adam : A Method for Stochastic Optimization "(Kingma and Ba 2015).

Adam merupakan pengembangan dari Adagrad serta kombinasi antara RMSProp dan Stochastic Gradient Descent (SGD) dengan momentum (Jais, Ismail, and Nisa 2019). Metode ini sangat efisien secara komputasi dan dalam prosesnya sedikit memori yang dibutuhkan. Adam optimizer salah satu algoritma gradient descent yang sering digunakan. Berikut adalah proses langkah – langkah ADAM sebagai berikut:

1. Menambah t pada setiap iterasi

$$t = t + 1 \tag{2.1}$$

2. Menghitung gradient

$$g_t = \nabla_\theta f_t \theta_{t-1} \tag{2.2}$$

3. Memperbaharui bias moment pertama

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)_{.gt} \tag{2.3}$$

4. Memperbaharui bias moment kedua

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)_{gt2} \tag{2.4}$$

5. Menghitung koreksi bias moment pertama

$$\hat{mt}_t = \frac{mt}{1 - \beta_t^1} \tag{2.5}$$

6. Menghitung koreksi bias moment kedua

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.6}$$

7. Memperbaharui parameter

$$v_t = \frac{\theta_{t-1} - \alpha . \hat{mt}_t + (1 - \beta_2)_{gt2}}{\sqrt{\hat{v}_t} + \in} \tag{2.7}$$

Keterangan:

g = Gradien

m = Momen pertama

v = Momen kedua

$\beta 1, \beta 2$ = Exponential decay rates

α = Stepsize

θ = Parameter yang diperbaiki

2. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent biasa disebut sebagai incremental gradient descent. Metode ini mencari suatu bobot baru dengan metode pengambilan salah satu data dari semua data training, sesudah itu SGD melakukan Analisa dari setiap data yang diambil. Menfaat memakai SGD yaitu mengurangi penggunaan memori yang diperlukan disaat pemtosesan bobot baru.

3. Adaptive Gradient Algorithm (Adagrad)

Adagrad merupakan suatu metode adaptive learning rate. Pada metode ini learning rate akan dijadikan parameter. Adagard menghilangkan keperluan untuk melakukan adaptasi learning rate secara manual. Tetapi, algoritma optimasi ini memiliki kekurangan yaitu learning rate yang terus menjadi kecil hingga tidak dapat lagi dilatih

4. Root Mean Square Propagation (RMSProp)

Metode ini merupakan penyempurnaan dari kekurangan yang ada pada Adagrad dengan memakai moving average dari gradient kuadrat. Metode RMSProp memakai besarnya gradient descent terbaru guna menormalkan gradient. Learning rate pada metode ini deselaraskan dengan cara otomatis serta memakai learning rate yang berlainan untuk setiap parameter. RMSProp memisah learning rate dengan rata-rata keseluruhan eksponensial dari gradient descent.

2.7.7 Loss Function

Loss function adalah fungsi yang mengukur seberapa bagus performa yang dihasilkan oleh model dalam prediksi terhadap target. Fungsi *loss* yang baik adalah fungsi yang menghasilkan *error* yang diharapkan paling rendah. Apabila model memiliki kelas yang cukup banyak perlu adanya cara untuk mengukur perbedaan antara probabilitas hasil hipotesis dan probabilitas hasil kebenaran asli. Berikut jenis loss function:

1. Mean Absolute Error

Mean Absolute Error (MAE), juga disebut L1 *Loss*, menghitung rata-rata jumlah perbedaan mutlak antara nilai aktual dan nilai prediksi. Ini memeriksa ukuran kesalahan dalam satu set nilai yang diprediksi, tanpa melihat arah positif atau negatifnya. Jika nilai absolut dari kesalahan tidak digunakan, maka nilai negatif dapat membatalkan nilai positif. MAE digunakan ketika distribusi variabel target memiliki *outlier*, seperti nilai-nilai kecil atau besar yang sangat jauh dari nilai rata-rata. Rumus persamaan *mean absolute error* adalah sebagai berikut

$$loss(x, y) = |x - y| \quad 2.1$$

2. Mean Squared Error

Mean Squared Error (MSE), juga disebut L2 Loss, menghitung rata-rata perbedaan kuadrat antara nilai aktual dan nilai prediksi. Pytorch MSE Loss selalu menghasilkan hasil positif, terlepas dari tanda nilai aktual dan prediksi. Untuk meningkatkan akurasi model, kita harus mencoba mengurangi L2 Loss sekecil mungkin atau mencapai nilai sempurna adalah 0,0. MSE digunakan saat melakukan regresi, percaya bahwa target Anda, dikondisikan pada input, terdistribusi normal, dan ingin kesalahan besar secara signifikan (kuadrat) lebih diberikan perhitungan daripada kesalahan kecil. Rumus persamaan mean squared error adalah sebagai berikut

$$loss(x, y) = (x, y)^2 \quad 2.2$$

3. Cross-Entropy Loss Function

Loss function ini menghitung perbedaan antara dua distribusi probabilitas untuk serangkaian kejadian atau variabel acak yang disediakan. Ini digunakan untuk menghitung skor yang merangkum perbedaan rata-rata antara nilai yang diprediksi dan nilai sebenarnya. Untuk meningkatkan akurasi model, kalian harus mencoba meminimalkan skor-skor lintas entropi antara 0 dan 1, dan nilai sempurna adalah 0. Fungsi Cross-Entropy memiliki variasi yang luas, jenis yang paling umum adalah Binary Cross-Entropy (BCE). BCE Loss terutama digunakan untuk model klasifikasi

biner; yaitu, model hanya memiliki 2 kelas. Persamaan matematika dari cross entropy sebagai berikut

$$loss(x, y) = -\sum x.\log(y) \quad (2.4)$$

Keterangan:

x = nilai actual

y = nilai prediksi