BAB II

LANDASAN TEORI

2.1. Tinjauan Terhadap Penelitian Yang Terkait

Di bawah ini adalah beberapa hasil pencarian relevan yang dijadikan acuan riset ini, yang terangkum tabel berikut.

Tabel 2.1 Referensi Riset yang Terkait

No	Penulis	Tahun	Metode	Judul	Kesimpulan
1	Peter Winardi dan Endang Setyati	2021	*Convoluti-onal Neural Network*	Identifikasi Jenis Daging Menggunaka n Algoritma *Convoluti-onal Neural Network*	Dalam penelitian ini melakukan identifikasi jenis daging yang berupa daging mentah ayam, babi, bebek, kambing dan sapi. Dengan metode CNN lalu membagi *dataset training* 70% dan validasi 30%. Pada ukuran citra 50 × 50 Piksel

					dengan 150 *epoch* didapatkan sebagai berikut • *training loss* : 43.89%, • *Training Accuracy* : 82.82%, • *Validation Loss* : 87.74 %, • *Validation Acuuracy* : 72.27%. Pada ukuran citra 100 × 100 Piksel dengan 150 *epoch* yang diperoleh • *Training Loss* : 35.74%, • *Training Accuracy* : 85.75%,

					• *Validation Loss* : 81.08 %, • *Validation Acuuracy* : 71.65%.
2	Sarah Lasniari Jasril, Suwanto Sanjaya, Febi Yanto dan Muhammad Affandes	2022	*Convoluti-onalNeural Network* Arsitektur ResNet50	Klasifikasi Citra Daging Babi dan Daging Sapi Mengguna-kan *Deep Learning* Arsitektur ResNet50 dengan Augmentasi Citra.	Dalam penelitian ini mengklasifikasi daging sapi dan daging babi mendapatkan accuracy 87,64%, *recall* 87,59% dan *precision* 90,90%.
3	Sarah Lasniari, Jasril, Suwanto Sanjaya, Febi Yanto	2022	*Convoluti-onalNeural Network* Arsitektur ResNet50	Pengaruh *Hyperpara-mer Convoluti-onal Neural Network*	Dalam penelitian ini mengklasifikasi daging sapi dan daging babi pada pengujian

	dan Muhammad Affandes			Arsitektur ResNet50 Pada Klasifikasi Citra Daging Sapi dan Daging Babi	hyperparameter. Penerapan pada *Split Dataset* 90% : 10 %, *Batch Size* 32, *Epoch* 75 dan *Learning Rate* 0.001 Dihasilkan nilai presisi, accuracy dan *recall* terbaik dihasilkan pada percobaan ke-71 penggunaan model ResNet-50 memperoleh nilai 100%
4	Dodi Efendi, Jasril, Suwanto Sanjaya, Fadhilah Syafria dan	2022	*Convoluti-onalNeural Network* Arsitektur ResNet50	Penerapan Algoritma *Convoluti-onal Neural Network* Arsitektur ResNet50	Dalam penelitian ini mengklasifikasi daging sapi dan daging babi dengan beberapa *optimizer* bahwa

	Elvia Budianita.			untuk Klasifikasi Citra Daging Sapi dan Babi	SGD adalah pengoptimal terbaik untuk arsitektur CNN ResNet-50 dan pengoptimal ini juga telah terbukti meningkatkan akurasi klasifikasi gambar daging sapi, babi dan daging campuran.
5	Calvin, Ghiri Basuki Putra dan Esa Prakasa	2020	*Convoluti-onal Neural Network* Ayam6Net, AlexNet, VGGNet dan GoogLeNet	*Classification of Chicken Meat Freshness using Convoluti-onal Neural Network Algorithms*	Dalam penelitian ini mengidentifikasi kesegaran daging ayam *boiler* menggunakan algoritma *convolutional neural network* Ayam6Net, AlexNet, VGGNet

					dan GoogLeNet. Ayam6Net memiliki akurasi tertinggi 92,9% dengan *dataset* 400 × 400 piksel.
6	Fachri Ramdhan Al Mubaroq	2022	YOLOv4	Aplikasi Deteksi Dini Penyakit Mata *Pink-Eye* Pada Sapi Berbasis Citra Mengguna-kan Model *Deep Learning* Algoritma YOLOv4	Dalam penelitian ini mendeteksi penyakit mata *pink-eye* pada sapi menggunakan algoritma YOLOv4. pada data *testing* didapatkan nilai *Average Precision* sebesar 88% pada kelas sapi bermata *pink-eye* dan 92% pada sapi bermata sehat.

2.2. Daging Sapi

Daging sapi merupakan hasil dari peternakan sapi yang bagian dapat digunakan sebagai bahan makanan sumber protein hewani mengandung nutrisi yang diperlukan tubuh manusia untuk berkembang. Dalam daging sapi mengandung komposisi nutrisi berupa 19% protein, 70% air, 5% lemak, 2,5% mineral dan sedikit karbohidrat. Dalam daging sapi mempunyai kadar air yang rata-rata 75%. Kadar air ini bisa mempengaruhi pertumbuhan mikroorganisme, sehingga terjadinya kebusukan pada daging sapi.

Daging sapi yang masih segar mempunyai beberapa ciri-ciri seperti pada daging berserabut halus dengan sedikit lemak, warna daging sapi merah terang, dan kenyal serta baunya tidak amis, segar dan sedikit gurih (Siregar, 2018).

Daging sapi segar yang layak konsumsi mempunyai ciri ciri warnanya lemaknya berwarna kekuningan, warna merah terang dan, kadar airnya sedikit dan mempunyai aroma yang amis segar. Daging sapi yang kurang segar ditandai warnanya yang merah pucat, kadar airnya sangat banyak, mengeluarkan aroma amis dan sedikit busuk (Hadi, Setiawan dan Sumardi, 2011).

Pada uji sensoris kualitas daging sapi, untuk daging sapi segar mempunyai ciri-ciri warnanya merah ungu gelap, teksurnya sangat kenyal dan baunya khas daging. Untuk daging yang kesegarannya berkurang pada penyimpanan selama 24 jam dengan daun jati yang mempunyai warna merah, tekturnya kenyal dan baunya agak busuk. Untuk daging busuk mempunyai warnanya coklat tekturnya kurang kenyal atau lunak, berair dan berbau busuk (Wulandari, 2014).

Pada daging sapi yang sudah busuk yang merupakan daging rusak karena seratnya banyak mengandung bakteri, warnanya biru kehitaman atau hijau tua dan lemak lunak dan berbau sangat busuk (Prabowo, Erwanto dan Rahayu, 2021).

Pada daging sapi segar mempunyai beberapa ciri-ciri warnanya merah terang, bersih, berserabut halus serta sedikit lemak, tekturnya kenyal, dan mengkilat, sementara untuk daging sapi yang sudah busuk mempunyai ciri-ciri dagingnya berwarna kehijau-hijauan atau kebiru-biruan, kusam dan berlendir baunya tengik dan tidak enak, berlendir serta permukaannya lengket (Afiati, 2009).

2.3. *Artificial Intelligence*

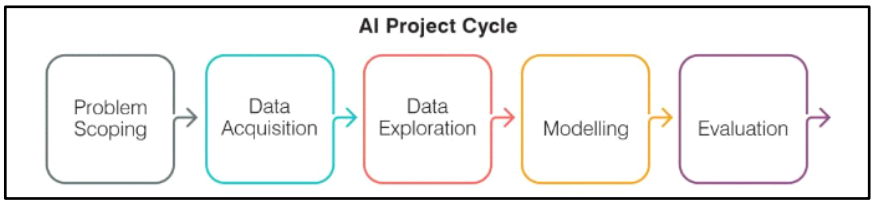
Kecerdasan buatan adalah kecerdasaan yang ditambahkan ke dalam suatu sistem yang bisa dikelola (Siahaan dkk, 2020). Kecerdasaan buatan adalah perangkat lunak atau sebuah program komputer yang memiliki mekanisme untuk mempelajari dan kemudian menggunakan pengetahuan tersebut untuk mengambil keputusan dalam situasi baru, seperti halnya manusia (Supriyadi dan Asih, 2020)

Kecerdasaan buatan atau disebut *artificial intelligence* adalah kecerdasaan yang dibuat melalui program komputer yang akan digunakan di dalam sistem, sehingga bisa membuat keputusan bahkan dapat bertingkah dan berpikir seperti manusia.

2.3.1. *AI Project Life Cycle*

AI Project Life Cycle yang dapat mendefinisikan setiap langkah yang harus diikuti untuk mengembangkan suatu proyek AI (Artificial Intelegence) sehingga

dapat memperoleh nilai maksimum dari kecerdasaan buatan dalam mencapi tujuan yang diinginkan (Sodhi, 2022).



Gambar 2.1 *AI Project Life Cycle*

AI Project Life Cycle dapat dibagi menjadi 5 tahapan yaitu:

1. *Problem Scoping*

   Problem scoping adalah tahap pertama dari siklus AI. itu adalah proses dimana masalah yang perlu dipecahkan diidentifikasi. *Problem scoping* adalah bagian dari fase definisi masalah dari siklus AI. kemudian diikuti dengan *brainstorming*, *designing*, *developing* atau *building*, *testing* dan menyelesaikan proyek.

2. *Data Acquisition*

   Tahap kedua dari siklus proyek AI adalah Akuisisi data. data adalah kumpulan fakta. data adalah elemen utama yang diandalkan oleh setiap proyek AI. data yang diperoleh menjadi dasar proyek karena membantu dalam memahami parameter yang terkait dengan *problem scoping*.

3. *Data Exploration*

   Tahap ketiga dari Siklus Proyek Al adalah Eksplorasi Data. Eksplorasi Data adalah langkah pertama dalam analisis data. Data dieksplorasi dalam tahap ini untuk memahaminya. Ini membantu dalam memahami

karakteristik data dan pola yang mungkin ada dalam data. Eksplorasi data dapat menggunakan kombinasi metode manual dan alat otomatis.

4. *Modelling*

   Tahap ke-empat dari Siklus Proyek AL adalah Pemodelan. Untuk membuat prediksi yang berguna dari data, algoritma yang tepat diimplementasikan untuk membangun model Al. Model Al mencoba menggunakan data dan *input* yang disediakan oleh *programmer* untuk mereplikasi keputusan yang akan dibuat oleh pakar manusia jika dia telah meninjau semua data yang tersedia. Model dapat Berbasis Aturan atau Berbasis Pembelajaran.

5. *Evaluation*

   Tahap kelima dari Siklus Proyek Al adalah Evaluasi. Model setelah dibuat, harus melalui serangkaian pengujian. Setiap kali Model dilatih pada kumpulan data, Model perlu diuji pada kumpulan data uji untuk memeriksa persentase akurasi yang diberikan olehnya. Ini membantu menentukan apakah Model memenuhi persyaratan proyek yang telah dirancang. Model hanya akan di-*deploy* dan digunakan setelah memastikan bahwa model tersebut memenuhi persyaratan seperti yang telah ditentukan.

2.3.2. *Machine Learning*

Pada awalnya *Machine learning* diperkenalkan oleh Arthur Samuel pada tahun 1959. Menurut Arthur Samuel, *machine learning* merupakan cabang ilmu

komputer yang memberikan komputer kemampuan untuk mempelajari sesuatu tanpa memerlukan seorang programmer (Pujoseno, 2018).

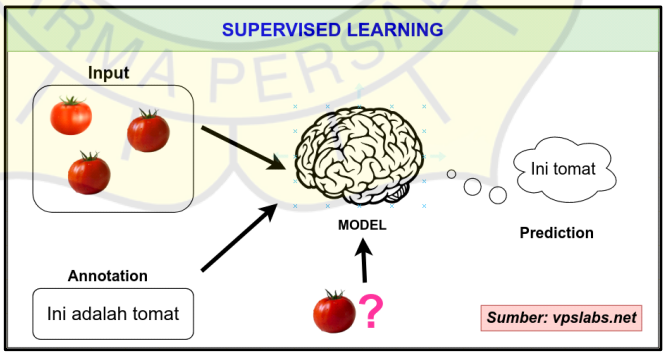
Pembelajaran mesin (ML) adalah cabang kecerdasan buatan yang penelitiannya berfokus pada merancang dan menganalisis algoritma untuk memungkinkan komputer belajar. *Machine learning* (ML) melibatkan algoritma umum (generik), dimana suatu algoritma dapat memperoleh sesuatu yang menarik atau berguna dari data tidak harus membuat kode tertentu (Id, 2021).

2.3.3. Jenis-Jenis *Machine Learning*

Machine Learning dikelompokkan cara pembelajaran masing-masing dengan data dan hasil yang diperoleh. *Machine Learning* dapat dikelompokkan tergantung dari gaya belajarnya, *machine learning* dibagi menjadi tiga kelompok yaitu:

1. *Supervised Learning*

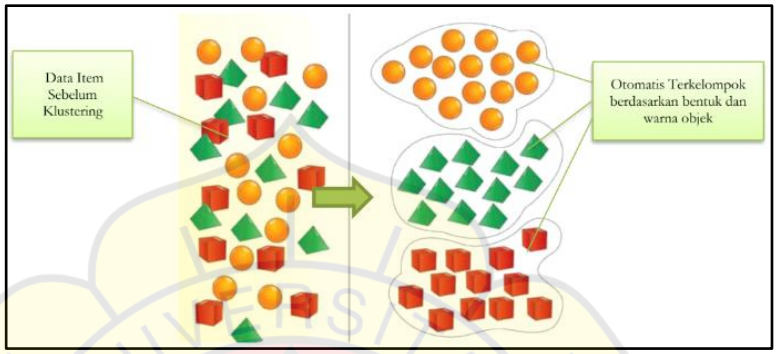
   Supervised Learning adalah pembelajaran yang terbimbing. Komputer akan mempelajari data pelatihan yang sudah berisi label.



Gambar 2.2 Ilustrasi *Supervised Learning*

2. *Unsupervised Learning*

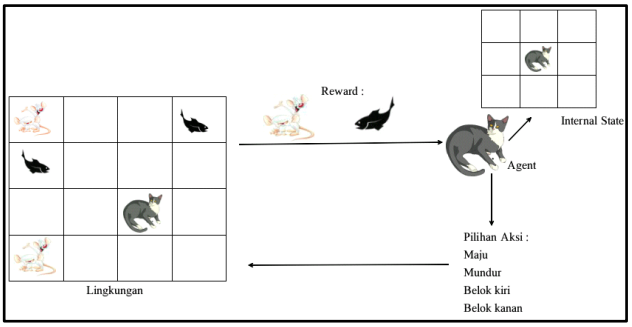
Unsupervised learning adalah pembelajaran mesin yang dimana data masukkan tidak mempunyai label. Oleh karena itu, *machine learning* tipe *unsupervised learning* mengelompokkan data berdasarkan karakteristik setiap bagian data yang ada.



Gambar 2.3 Ilustrasi *Unsepervised Learning*.

3. *Reinforcement Learning*

Reinforcement Learning merupakan pembelajaran mesin yang berbeda dari sebelumnya. Pada *machine learning* tipe *reinforcement learning* ini, Latihan dan ujian dilakukan bersamaan dan dinamis. Informasi dari belajar dikumpulkan secara aktif dalam interaksi dengan lingkungan sehingga memperoleh *feedback* yang berguna untuk pembelajaran yang dapat memperbarui perameternya.



Gambar 2.4 Ilustrasi *Reinforcement Learning*

2.3.4. *Deep Learning*

Deep Learning adalah Sub-bidang dari *artificial intelligence* (AI) atau *machine learning* (ML) yang berfokus kepada pembuatan model *neural network* yang mampu membuat keputusan data yang akurat. (Kelleher, 2019).

Deep learning mempunyai karaktersistiknya yang berdasarkan teknik pembelajarannya dapat dikelompokkan menjadi empat yaitu:

1. *Deep Unsupervised Learning* (DUL)

   Deep learning tipe *unsupervised learning* ini memodelkan sekumpulan data yang dimana inputannya secara otomatis tanpa ada panduan yang berupa *output* atau label yang diinginkan.

2. *Deep Supervised Learning* (DSL)

   Deep learning tipe *supervised learning* ini memodelkan sekumpulan data yang dimana inputannya berdasarkan *output* label setiap kelas.

3. *Deep Semi Supervised Learning* (DSSL)

   Deep Learning tipe semi *supervised learning* ini dimana menggunakan sampel *input* yang sebagian memiliki label dan sebagian lainnya tidak memiliki label.

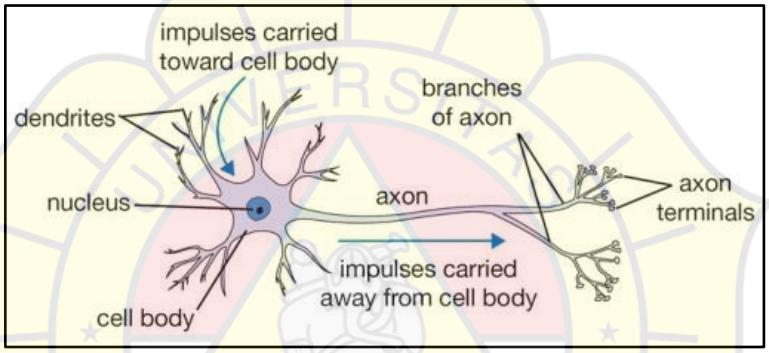
4. *Deep Reinforcement Learning* (DRL)

   Deep Learning tipe *reinforcement learning* merupakan gabungan dari *deep learning* dan *reinforcement learning* yang dimana mempelajari suatu mengambil tindakan berdasarkan hasil pengamatan terhadap lingkungan yang ada. Setiap tindakan menciptakan konsekuensi lingkungan dan memberikan *feedback* yang berguna untuk pembelajaran selanjutnya dengan parameter baru.

2.3.5. *Artificial Neural Network* (ANN)

Artificial neural network atau *neural network* adalah salah satu algoritma dalam *machine learning* yang dimana sebuah model komputasi diadapdasikan dari cara kerja saraf otak manusia.

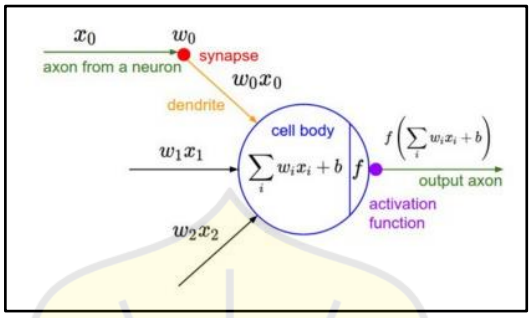
Otak manusia terdiri dari milyaran *neuron* yang dihubungkan oleh *sinapsis*, *dendrit* berfungsi sebagai penerima informasi yang masuk yang ditransmisikan dan diproses oleh *nukleus*. Akson bertugas meneruskan *impuls*, atau hasil pengolahan informasi, dari badan sel ke jaringan lain.



Gambar 2.5 Jaringan Saraf Pada Manusia

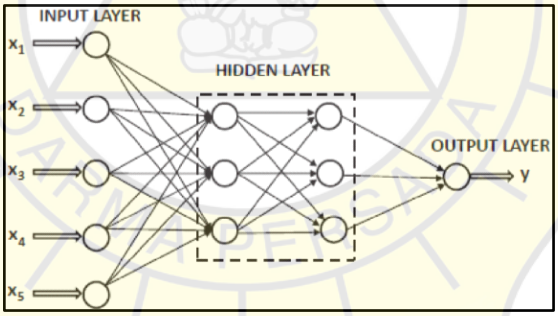
Model *neural network* seperti jaringan saraf pada otak manusia. Jaringan saraf terdiri dari banyak *neuron* yang terhubung satu sama lain. Masing-masing neuron ini mengubah informasi yang diterima melalui hubungannya dengan neuron lain. Setiap *node* atau *input* tersebut mempunyai bobot atau *weight* yang berhubungan dengan masing-masing *node*. Setelah memasuki *neuron*, nilai-nilai input yang ada dijumlahkan dan jumlahkan lagi dengan nilai bias. Kemudian, setelah dijumlahkan, fungsi aktivasi setiap *neuron* memprosesnya. Nilai tambah dibandingkan dengan nilai ambang batas. Jika nilainya di bawah ambang batas, *neuron* dihentikan, jika tidak diaktifkan.

Ketika sebuah *neuron* aktif, ia mengirimkan nilai keluaran ke semua *neuron* yang terhubung dengannya melalui bobot *keluaran*. Proses ini berlanjut dengan pendapatan berikutnya. Jaringan saraf tiruan memiliki banyak *neuron*. *Neuron* dikelompokkan dalam beberapa lapisan.



Gambar 2.6 Ilustrasi *Artificial Neural Network*

Di nilai *input layer* akan dilanjutkan ke dalam *layer-layer* selanjutnya sampai mencapai *layer* terakhir atau disebut *output layer*. Diantara *input layer* dan *output layer* ada *hidden layer*.

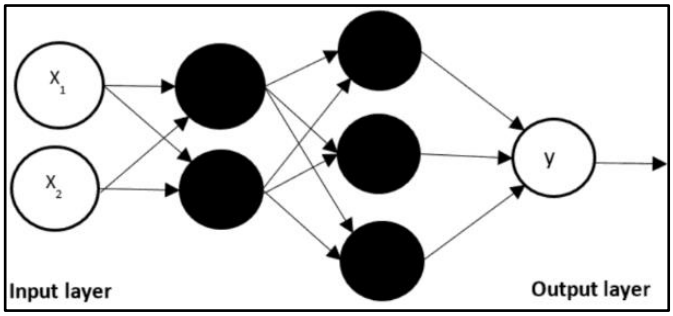


Gambar 2.7 Model Sebuah *Neural Network Multi-Layer*

Berbagai arsitektur *neural network* dapat menjadi dua kategori utama yaitu:

1. *Feed-forward Neural Network*

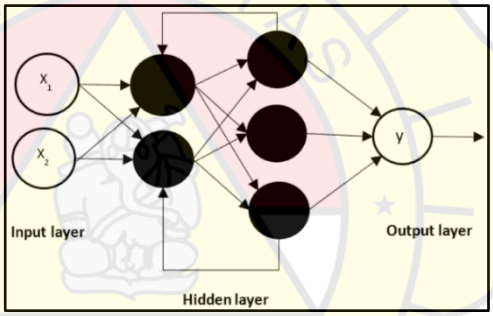
   Sebuah *neural network* dimana koneksi penghubung dua *node* tidak membentuk sebuah siklus.



Gambar 2.8 Model *Feed-forward Neural Network*

2. *Recurrent Neural Network*

   Sebuah *neural network* dimana ada satu atau beberapa koneksi penghubung dua *node* membentuk sebuah siklus untuk mengulang kembali sebuah *node*.



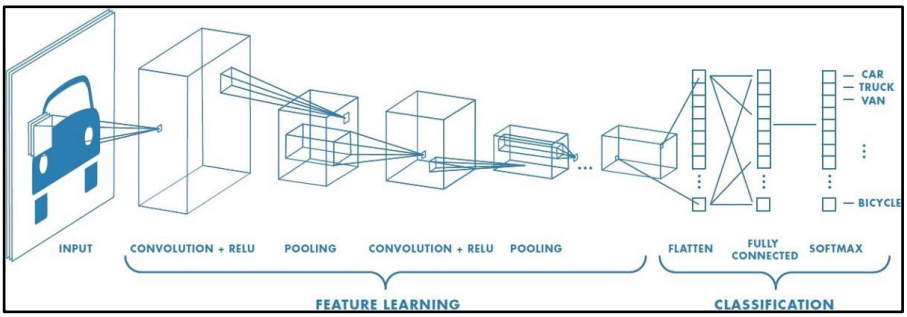
Gambar 2.9 Model *Recurrent Neural Network*

2.3.6. Convolutional Neural Network

Jaringan safat konvolusional merupakan metode pembelajaran mendalam yang berawal dari pengembangan *multi-layer perceptron* (MLP) yang dirancang untuk mengolah data 2D berupa gambar.

Jaringan saraf *convolutional* didasarkan pada konsep dalam biologi yang disebut bidang reseptif. Bidang reseptif ini adalah fitur korteks visual pada hewan,

yang berfungsi sebagai pendeteksi sensitif untuk jenis rangsangan tertentu (Stenroos, 2017).



Gambar 2.10 Ilustrasi Jaringan CNN

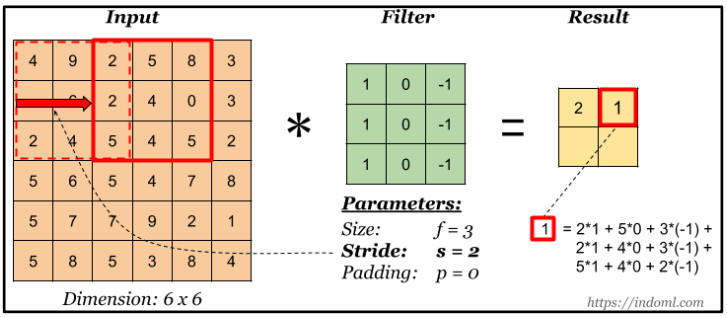
Pada gambar di atas bahwa CNN mempunyai tiga bagian seperti *input* (gambar), pembelajaran fitur dan klasifikasi. Data dipecah menjadi beberapa bagian berdasarkan ukuran *batch*. Informasi ukuran tumpukan diproses dalam fase pembelajaran fitur, yaitu lapisan konvolusi, lapisan pooling, lapisan normalisasi, dan fungsi aktivasi ReLu. Hasil pembelajaran fitur berupa *feature learning* dari citra digunakan sebagai data *input* pada tahap *classification*. Pada tahap *classification* CNN digunakan metode jaringan syaraf tiruan. *fully connected layer* dan *softmax activation* untuk mencari *probability value* pada setiap kelas klasifikasi.

Pada *convolutional neural network* tersusun atas beberapa *feature learning* dan *classification* yaitu sebagai berikut.

1. *Convolution Layer*

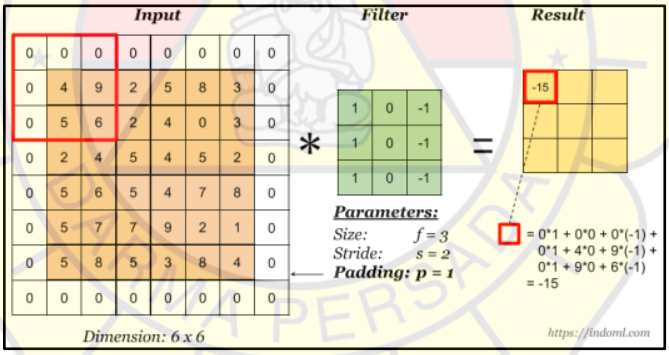
   Convolution dapat diartikan sebagai pengoperasian dua fungsi. Saat melakukan analisis citra atau gambar, fungsi yang pertama adalah nilai *input* berupa nilai *pixel* pada suatu lokasi dalam gambar. Fungsi kedua adalah matriks atau *filter* kernel. Nilai yang dihasilkan diperoleh dari

perkalian titik dua fungsi. *Stride* sebagai pergeseran dari *Filter* atau kernel ke titik berikutnya pada gambar (Ker, 2017).



Gambar 2.11 Gambaran Kalkulasi dengan Pergeseran *Stride*

Dalam lapisan konvolusional dikenal istilah *padding*, yaitu suatu bentuk manipulasi suatu citra atau citra masukkan dengan cara menambahkan piksel-piksel yang mengandung nilai 0 pada setiap tepi citra atau citra tersebut, sehingga bisa disebut *zero padding*.



Gambar 2.12 Gambaran Kalkulasi dengan Penambahan *Zero Padding*

Dalam *convolution layer* untuk mendapatkan ukuran dimensi keluaran dari suatu *feature map* maka dapat ditentukan oleh *input image*, kernel atau *filter*, *stride* dan *padding*. Berikut kalkulasi *feature map* berdasarkan banyaknya *padding* yang digunakan pada gambar.

- *No padding*

$$H = \frac{I - K}{S} + 1$$

- *Half padding*

$$H = \frac{I - K + P}{S} + 1$$

- *Full padding*

$$H = \frac{I - K + 2P}{S} + 1$$

Dimana,

H = dimensi dari *feature map* hasil *convolutional layer*

I = dimensi gambar masukkan

K = dimensi *matrix* dari *kernel* atau *filter*

P = banyaknya *padding*

S = banyaknya *stride*

2. *Batch Normalization*

Normalisasi batch diperkenalkan oleh Loffe pada tahun 2015, untuk menstandarisasi *input layer* dengan mengatur dan menskalakan *activation layer*, yang dapat meningkatkan stabilitas jaringan, kecepatan komputasi dan kinerja (Loffe, 2015)

Layer masukkan yang dinormalisasi dengan *d*-dimensi $x = [x_1, x_2, \dots, x_d]$ dinyatakan dalam:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\delta_B^2 + \epsilon}}$$

Dengan,

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\delta_{\boldsymbol{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

Pada μ_B adalah nilai *average* dari *batch* dan $\delta_{\boldsymbol{B}}^2$ adalah nilai varians *batch* dengan m adalah kumpulan *dataset* pelatihan. ϵ adalah nilai konstanta yang meningkatkan stabilitas *numeric* ketika varians *batch* sangat rendah (loffe, 2015). Maka keluaran dari sebuah lapisan *bacth normalization* ini adalah sebagai berikut.

$$y_i = \gamma\hat{x}_1 + \beta$$

Dimana γ dan β adalah parameter pembelajaran pada saat pelatihan.

3. *Rectified Linear Unit* (ReLu)

   ReLu adalah *activation function* yang sering digunakan di CNN, di mana g menunjukkan fungsi *non-linear pixel-wise*, yang berarti nilai *output* berasal dari fungsi ReLu ini. jika x positif maka akan menjadi bernilai x dan jika x negatif maka akan menjadi bernilai 0.

   $$g(x) = max\ (0, x)$$

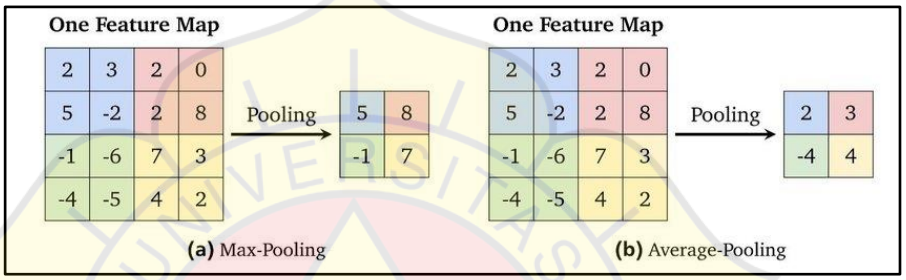
   ReLu mempunyai sifat *non-linier* dengan mengkombinasi *non-linier*, sehingga lapisan yang berbeda dapat ditumpuk satu sama lain. Nilai keluaran berkisar 0 hingga tak terhingga.

4. *Pooling Layer*

   Pooling Layer merupakan komponen lain dalam CNN yang bertujuan mengurangi resolusi lebar dan tinggi dari *feature map* (Castiglioni dkk,

2021). Pada langkah *pooling* dilakukan dengan cara membagi *output* lapisan konvolusional di beberapa *grid* berdasarkan penentuan jumlah *stride* dan jenis *pooling* yang digunakan.

Pada metode *pooling* ini mempunyai dua jenis *pooling*, yaitu *average pooling* serta *max pooling*. Pada *average pooling*, nilai yang dipertahankan adalah nilai rata-rata setiap *grid*. Sedangkan *max pooling*, nilainya diambil sebagai nilai maksimal masing-masing *grid*.



Gambar 2.13 Gambaran Kalkulasi pada *Polling Layer*

Saat menghitung ukuran *output* dari *pooling layer*, hal ini dapat dinyatakan dengan persamaan (Dumoulin, 2018).

$$O = \frac{I - K}{S} + 1$$

Dimana,

O = dimensi dari *output* dari *pooling layer*

I = dimensi *matrix* masukkan

K = dimensi *matrix kernel* atau *filter*

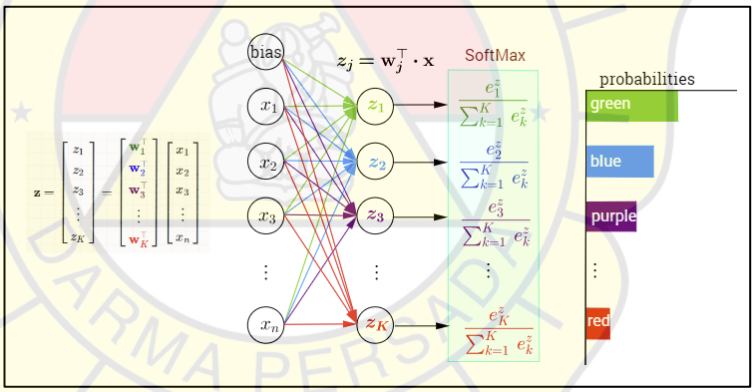
S = banyaknya *stride*

5. *Flatten*

   Pada *feature learning* umumnya pada proses ekstrasi fitur CNN selalu berbentuk *multidimensional array*, maka diperlukan suatu *flatten* agar menjadi nilai *input* untuk proses mengklasifikasikan gambar.

6. *Fully Connected Layer*

   Pada *clasissification* pada arsitektur CNN terdapat lapisan terakhir dalam *convolutional neural network* yaitu *fully connected layer*, artinya setiap *neuron* pada lapisan sebelumnya terhubung ke setiap *neuron*. Pada lapisan ini menggunakan nilai keluaran dari lapisan *feature map* sebagai nilai *input* dan menghitung probalitas untuk diklasifikasi ke dalam berbagai kelas.



Gambar 2.14 Ilustrasi *Fully Connected Layer*

Fully connected layer dapat dijelaskan berikut ini.

$$z_j = \sum_{j=1}^{c} w_{i,j}^T x_i + b_j$$

Pada z_j adalah nilai dari keluaran dari jaringan dengan nilai x_i sebagai nilai masukkan hasil dari *feature learning* dan $w_{i,j}$ merupakan sebuah

bobot jaringan adalah $i \times j$, dimana nilai i jumlah fitur *input* dan j adalah jumlah target kelas dan b_j adalah bias dari jaringan.

7. *Softmax*

   Softmax adalah fungsi aktivasi yang memungkinkan klasifikasi linear dengan mengkalkulasikan nilai probabilitas terhadap suatu kelompok. Nilai *output* setiap kelompok memiliki rentang nilai 0 sampai 1. (Nwankpa dkk, 2018). *Softmax* dapat didefinisikan sebagai berikut ini.

$$s_j = \frac{e^{z_j}}{\sum_{d=1}^{C} e^{z_d}}$$

8. *Cross Entropy*

   Cross entropy merupakan metode yang digunakan oleh jaringan saraf untuk mengkalkulasi serangkaian solusi yang berpusat pada arah optimal. Dalam *cross entropy*, ini dipakai sebagai *loss function* untuk memaksimalkan efisiensi *learning* dengan menghitung nilai kesalahan. (Zhou dkk, 2019). *Cross entropy* dapat menentukan seberapa jauh nilai hasil prediksi dan nilai sesungguhnya (Murphy, 2012). *Cross entropy* dapat didefinisikan sebagai berikut ini.

$$D(s,p) = -\sum_{j} p_j \, log \, s_j$$

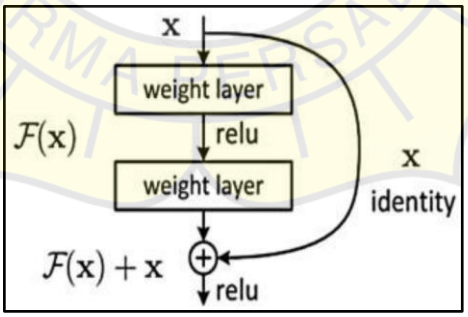
2.3.7. Resnet

Pada tahun 2015 Kaiming He, Xiangyu Zhang, Saqing Ren dan Jian Sun mengusulkan penelitiannya tentang arsitektur baru dari *Convolutional Neural Network* (CNN), khususnya model arsitektur *Deep Residual Network* (ResNet) (He dkk, 2016).

Arsitektur ini memberikan performa yang baik dalam menyelesaikan permasalahan pada model CNN bernama *vanishing gradient*, dimana *gradien* mulai menghilang pada lapisan terakhir dan tidak berubah pada lapisan berikutnya. Masalah ini membuat pelatihan sistem menjadi lambat dan rawan kesalahan. Untuk Mengatasi masalah ini, model ResNet membuat blok residual. Setiap *layer* arsitektur CNN ResNet memiliki blok sebagai fungsi *identity* (He dkk, 2016). Berikut ini persamaan *residual block*

$$y = F(x, W_i) + x$$

Dimana *x* dan *y* merupakan *vector input* dan *output layer*, *F* adalah fungsi residual yang sebelumnya dipelajari dan W_i adalah *weight layer*.



Gambar 2.15 *Residual Block*

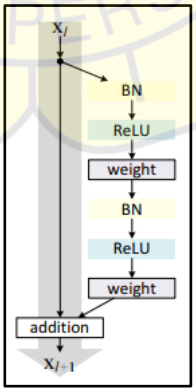
ResNet memiliki arsitektur berbeda dengan kedalaman jaringan yang berbeda, seperti arsitektur ResNet18, Resnet50 dan ResNet 101. Setiap macam-macam ResNet memiliki jenis lapisan yang serupa, perbedaannya hanya pada

banyaknya lapisan konvolusi, banyaknya matriks *filter*, dan jumlah operasi *floating point per detik* (FLOP), yang merupakan panjang *throughput komputer* per detik.

Layer Name	ResNet-18	ResNet-50	ResNet-101	Output Size
Conv1	$7 \times 7, 64, stride\ 2$			112×112
Conv2_x	3×3 max $pool, stride\ 2$			56×56
	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	
Conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	28×28
Conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	14×14
Conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	7×7
Average pool, 1000-d fc, softmax				1×1
FLOPs	$1,8 \times 10^9$	$3,8 \times 10^9$	$7,6 \times 10^9$	

Gambar 2.16 Perbandingan Arsitektur Pada Model ResNet

Selain dalam arsitektur CNN ResNet mempunyai improvisasi dalam *residual block* yang bernama ResNetV2. Yang mempunyai perubahan dalam *residual block* yang dimana menghapus non-linearitas terakhir sehinggan menerapkan *normalisasi batch* dan aktivasi ReLu sebelum perkalian dengan operasi konvolusi (He dkk, 2016).

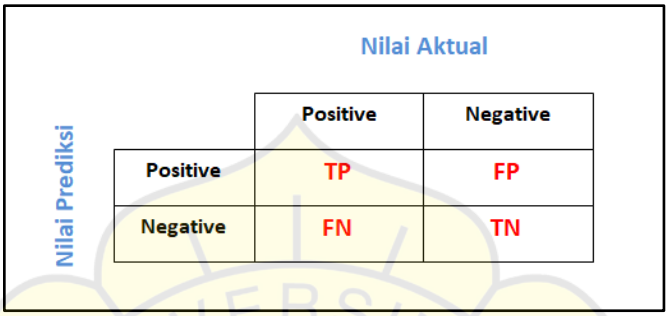


Gambar 2.17 *Residual Block* Versi 2

2.4. Mengukur Kinerja

2.4.1. *Confusion Matrix*

Confusion matrix adalah parameter yang dapat mengukur kinerja dari sebuah model *image classification*. *Confusion matrix* dapat diterapkan untuk klasifikasi biner dan klasifikasi *multi-class*. (Batarseh dan Yang, 2020)



Gambar 2.18 *Confusion Matrix*

Confusion matrix mempunyai beberapa penjelasan sebagai berikut :

1. *True Positive* (TP)

   TP menunjukkan banyaknya sampel positif yang diprediksi secara akurat.

2. *True Negative* (TN)

   TN menunjukkan banyaknya sampel negatif yang diprediksi secara akurat

3. *False Positive* (FP)

   FP menunjukkan banyaknya sampel yang sebenarnya negatif tetapi diprediksi positif

4. *False Negatif* (FN)

   FN menunjukkan banyaknya sampel yang sebenarnya positif tetapi dipresiksi negatif.

2.4.2. Akurasi

Perbandingan antara data yang diklasifikasi secara benar dengan data keseluruhan (Pratiwi, Handayani dan Sarjana, 2020). Yang mengukur seberapa besar tingkat keberhasilan pada klasifikasi yang dilakukan. Berikut ini persamaan Nilai akurasi.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

2.4.3. *Precision (Specificity)*

Precision atau *specificity* merupakan sebuah nilai hasil klasifikasi daging sapi segar yang tidak diidentifikasi sebagai daging setengah segar dan busuk. Untuk menghitung *precision* gunakan persamaan berikut ini.

$$Precision = \frac{TP}{TP + FP}$$

2.4.4. *Recall (sensitifivity)*

Recall atau *sensitifivity* merupakan sebuah nilai hasil klasifikasi daging sapi segar yang diidentifikasi sebagai daging sapi segar. Untuk menghitung *recall* menggunakan persamaan berikut ini.

$$Recall = \frac{TP}{TP + FN}$$

2.4.5. *F1-Score*

Rata-rata harmonik dari *precision* dan *recall* (Saputro dan Sari, 2019). Rumus *F1-Score* dapat dituliskan menggunakan persamaan berikut ini.

$$F1\ Score = 2\ \times \frac{precision\ \times\ recall}{precision + recall}$$

2.5. Pemodelan UML

UML adalah bahasa untuk mendefinisikan, mendokumentasikan, memvisualisasikan dan membuat artefak *software system* (potongan informasi yang dihasilkan oleh proses pembuatan *software*, seperti model, deskripsi dan perangkat lunak) (Destriana dkk, 2021).

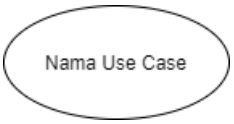
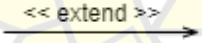
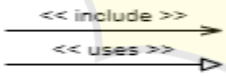
2.5.1. *Use Case Diagram*

Diagram *use case* merupakan model yang mendeskripsikan perilaku *system* yang dibuat (Sugiarti, 2018).

Diagram *use case* menggambarkan hubungan antara satu atau lebih *actor* dan *system* yang akan dibuat. Dengan menggunakan diagram *use case*, Anda dapat dengan cepat melihat fitur apa saja yang tersedia di sistem anda dan siapa yang berhak untuk menggunakan fitur tersebut.

Diagram *usecase* berisi beberapa simbol yang perlu anda dipahami. simbol-simbol pada *usecase* diagaram sebagai berikut.

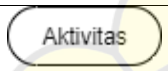
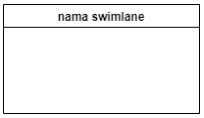
Tabel 2.2 Simbol *Use Case Diagram*

No	Simbol	Deskripsi
1	*Use Case* 	Fungsionalitas yang sebagai suatu entitas untuk bertukar pesan antar entitas atau aktor.
2	*Actor* 	*Actor* merupakan digambarkan sebagai orang, yang berinteraksi dengan sistem informasi.
3	Asosiasi/*association* 	Komunikasi antara *actor* dan *Use Case.* Partisipan dalam *Use Case* berinteraksi dengan *actor*.
4	Ektensi/*extend* 	Hubungan antara *use case* yang ditambahan dapat berdiri sendiri tanpa memerlukan *use case* tambahan.
5	*Include* 	Hubungan antara *use case* yang ditambahkan tersebut mengharuskan *use case* tersebut untuk melakukan tugasnya yang sebagai syarat untuk melakukan *use case* tersebut.

2.5.2. *Activity Diagram*

Menurut Rosa & Shalahuddin (2018), Diagram aktivitas digambarkan seperti operasi suatu sistem, proses bisnis, atau menu perangkat lunak.

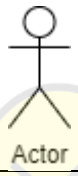
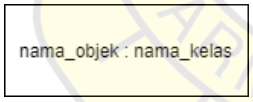
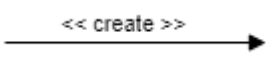
Tabel 2.3 Simbol *Activity Diagram*

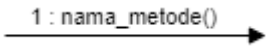
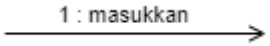
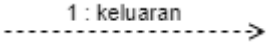
No	Simbol	Deskripsi
1.	 Status awal	Diagram operasi dengan keadaan awal.
2.	 Aktivasi	Aktivasi yang dilakukan oleh sistem.
3.	 Percabangan / *decision*	Percabangan asosiasi dimana terdapat lebih dari satu pilihan kegiatan.
4.	Pengabungan / *join*	Mengabungkan operasi menjadi satu.
5.	 Status akhir	Diagram fungsional memiliki untuk menjelaskan keadaan akhir.
6.	 *Swimlane*	Memisahkan aktivitas yang terjadi dengan *swimlane*.

2.5.3. *Sequence Diagram*

Sequence diagram mendeskripsikan perilaku objek dalam kasus penggunaan dengan menggambarkan masa hidup objek serta pesan yang dikirim dan diterima antar objek. Pembuatan *sequence diagram* juga diperlukan untuk melihat skenario yang ada pada kasus penggunaan (Rosa dan Shalahuddin, 2018).

Tabel 2.4 Simbol *Sequence Diagram*

No	Simbol	Deskripsi
1.	Aktor 	*Actor* merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi.
2.	Garis hidup / lifeline	Menunjukkan kehidupan suatu benda.
3.	Objek 	Mewakili audiens yang berinteraksi dengan kegiatan pesan sistem teresebut.
4.	Waktu aktif 	Dengan mendeklarasikan suatu objek yang aktif dan interaktif, segala sesuatu yang berhubungan dengan masa hidup.
5.	Pesan tipe *create* 	Objek menciptakan objek lain.

6.	Pesan tipe *call* 	Objek memanggil fungsi/metode pada objek lain atau pada objek itu sendiri.
7.	Pesan tipe *send* 	Objek sedang mengirimkan data/*input*/informasi ke objek lain, tanda panah menunjuk ke objek yang dikirim.
8.	Pesan tipe *return* 	Objek yang melakukan operasi yang menghasilkan respon terhadap objek tertentu,
9.	Pesan tipe *destroy* 	Objek mengakhiri umur benda lain, tanda panah menunjukkan benda yang akan diakhiri.