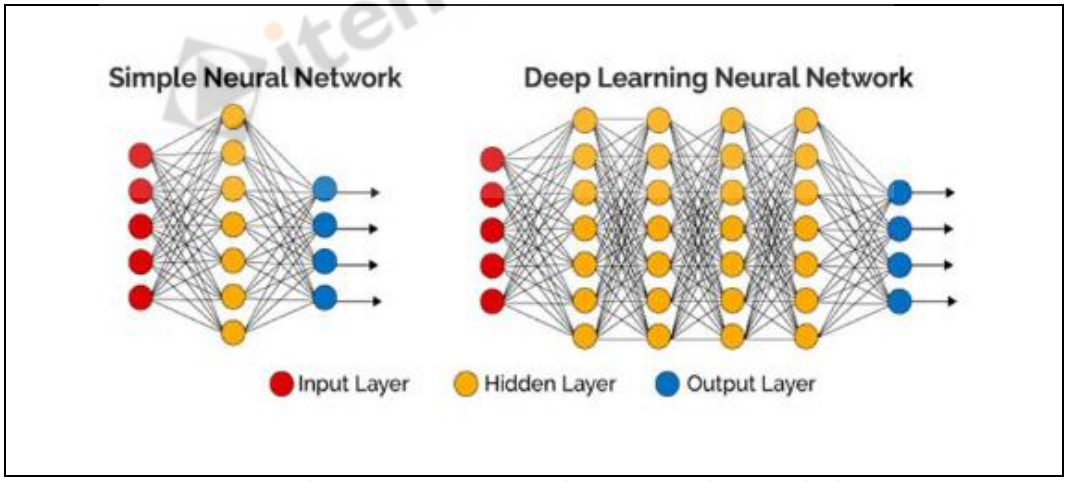
BAB II

LANDASAN TEORI

Berikut ini merupakan beberapa teori yang digunakan dalam penelitian yang dilakukan.

2.1 *Deep Learning*

Pada gambar 1 menampilkan bahwa setiap *deep learning* bisa disebut *machine learning* tetapi setiap *machine learning* tidak bisa disebut *deep learning*. Dengan menggunakan *deep learning* pembelajaran data akan lebih banyak layer dan berlapis-lapis dibandingkan dengan *machine learning* yang hanya menggunakan satu atau dua layer saja. Menurut (Chollet, 2018) dengan pembelajaran hanya satu atau dua layer bisa disebut pembelajaran yang belum mendalam benar. Metode *deep learning* adalah metode pembelajaran dengan berbagai layer, diperoleh dengan menyusun modul sederhana tetapi non-linear yang masing-masing mengubah representasi pada satu tingkat (dimulai dengan *input* mentah) menjadi representasi pada tingkat yang lebih tinggi.

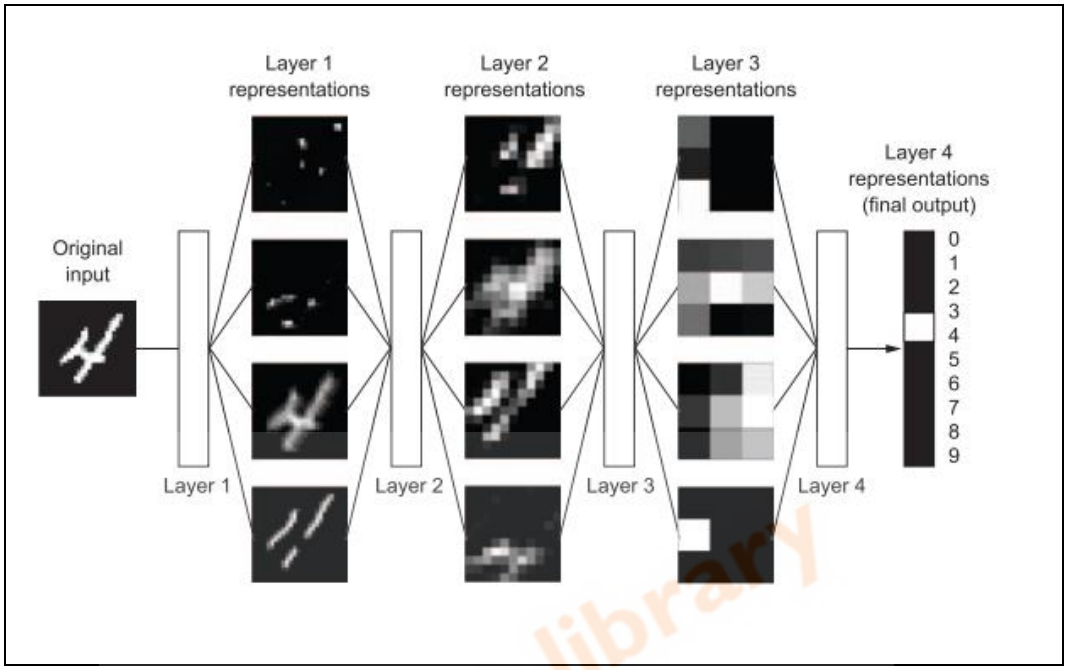


Gambar 1 Perbandingan layer machine learning (kiri) dan deep learning (kanan)

Sumber: (Savalia & Emamian, 2018)

Penerapan *deep learning* telah digunakan dalam beberapa bidang seperti klasifikasi video, klasifikasi gambar, *object detection, object recognition, text-to-speech, natural language processing, robotic, text classification,* dan *singing*

synthetis (Minar & Naher, 2018). Gambar 2 menjelaskan sebagian kecil proses klasifikasi gambar.



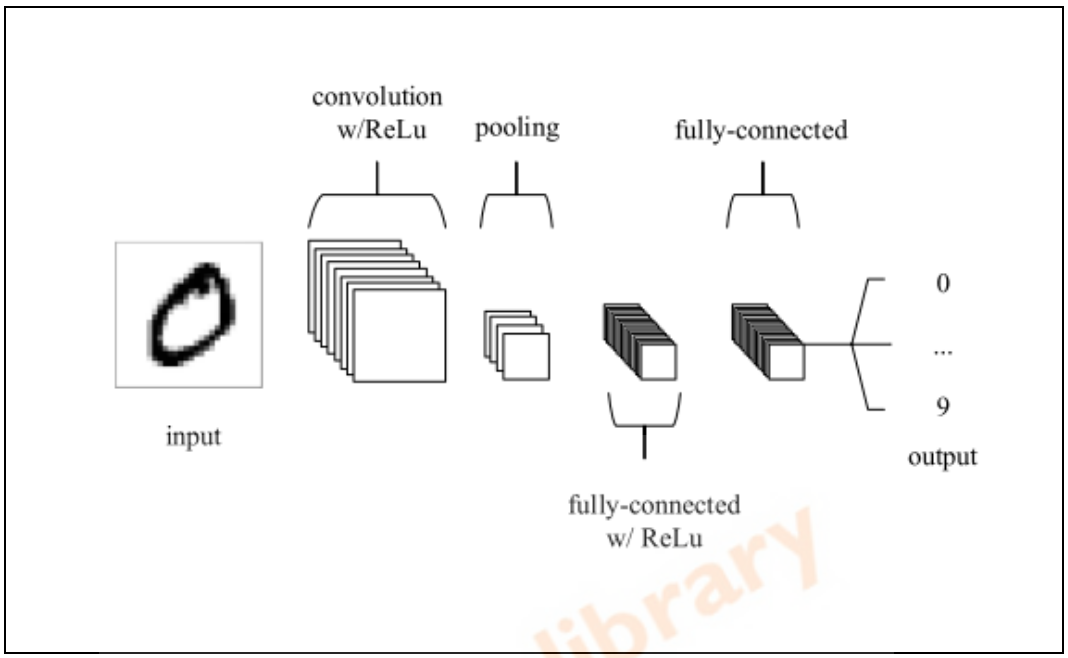
Gambar 2 Proses *Training Deep Learning*

Sumber: (Chollet, 2018)

2.2 Convolution Neural Network

Algoritma convolutional neural network adalah *perceptron multilayer* yang merupakan didesain khusus untuk identifikasi informasi gambar dua dimensi. Selalu memiliki lebih banyak lapisan: lapisan *input*, lapisan konvolusi, lapisan sampel dan lapisan keluaran. Keuntungan dari *algoritma* CNN adalah untuk menghindari eksplisit ekstraksi *fitur*, dan secara implisit untuk belajar dari data pelatihan; neuron yang sama pada bobot permukaan pemetaan *fitur*, sehingga jaringan dapat belajar secara paralel, mengurangi kompleksitas jaringan; mengadopsi struktur sub-sampel berdasarkan waktu atau ruang, dapat mencapai tingkat ketahanan, skala, dan deformasi perpindahan; informasi *input* dan *topologi* jaringan dapat menjadi pasangan yang sangat baik, ini memiliki keunggulan unik dalam pengenalan suara dan pemrosesan gambar (T. Liu et al., 2015). Pada perkembangannya terdapat banyak arsitektur cnn yang sering digunakan seperti r-*cnn*, *faster* r-cnn, ssd (*single shot multibox detector*), *alexnet*, *vggnet*, *googlenet*,

resnet, dll. Secara umum terdapat 3 *layer* utama pada cnn, yaitu *convolutional layers, pooling layers,* dan *fully connected layers.*



Gambar 3 Arsitektur Dasar CNN

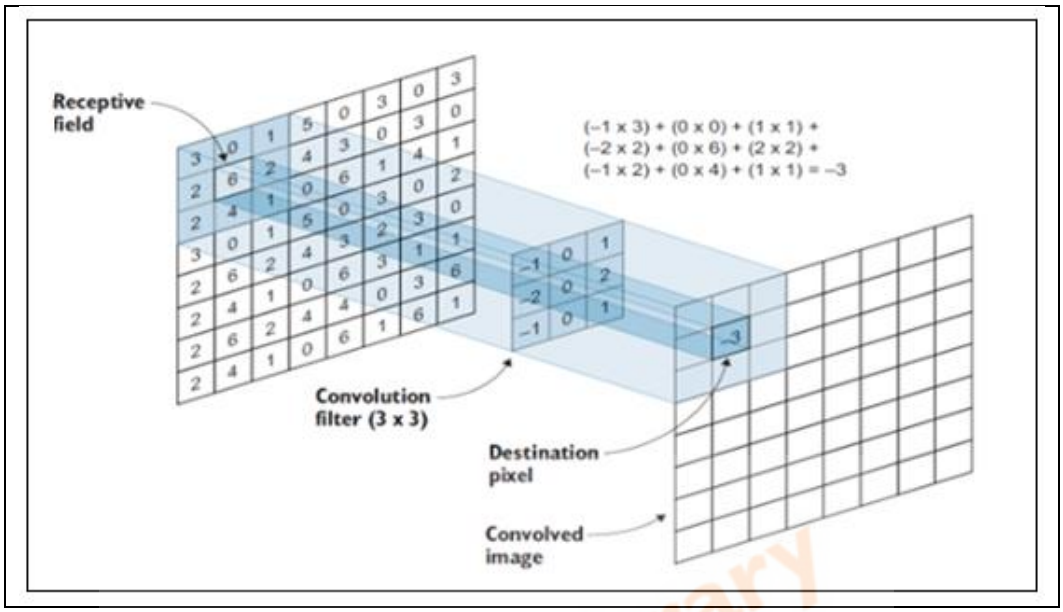
Sumber: (O'Shea & Nash, 2015)

Terdapat operasi-operasi tambahan lain yang dilakukan pada CNN untuk meningkatkan akurasi klasifikasi seperti zero *padding, ReLU activation,* dan *batch normalization*. Pada *convolutional layers,* dilakukan proses operasi konvolusi pada citra masukan terhadap *kernel* atau filter untuk mendapatkan *fitur* pada citra. Kemudian fungsi *pooling layer* adalah untuk memperkecil dimensi dari *fitur* yang telah didapat dari *convolutional layer*. Sedangkan pada *fully connected layer*, citra yang telah diperkecil pada *pooling layer* di rubah menjadi satu dimensi dan diklasifikasikan berdasarkan data latih yang dimiliki pada *database* (Guo et al., 2016).

2.2.1 *Convolution*

Convolution adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Gambar 4 menunjukkan *input,* filter dan *result*. *Input* secara keseluruhan adalah citra yang akan di

konvolusi. filter bergerak dari sudut kiri atas ke kanan bawah. Kemudian *result* gambar kotak kuning menghasilkan nilai konvolusi.

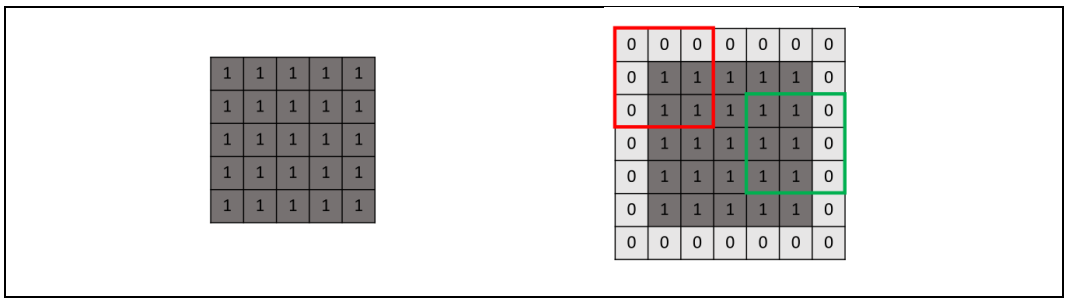


Gambar 4 Operasi konvolusi

(Elgendy, 2019a)

2.2.2 Zero Padding

Zero Padding sangat dibutuhkan oleh *convolution* ketika bagian dari filter melampaui gambar *input* atau peta *fitur* (G. Liu et al., 2018). Fungsi dari pada zero *padding* adalah penambahan operasi citra dengan angka 0. Gambar 5 menunjukkan penambahan zero *padding*.



Gambar 5 Zero Padding

(Liu, G et al, 2018)

2.2.3 *Batch Normalization*

Digunakan untuk mengurangi pergeseran *kovarian* atau menyamakan distribusi setiap nilai *input* yang selalau berubah karena perbuahan pada *layer*

sebelumnya selama proses *training* (Ioffe & Szegedy, 2015). Operasi *batch normalization* dilakukan sebelum fungsi aktivasi setiap lapisan input. Proses *batch normalization* terdiri dari proses *zero-center input* dengan menghitung *mini batch mean* dan *minibatch varian* lalu dinormalisasi kemudian menghitung *scale & shift*. Adapun penjelasan tiap proses yang dilakukan yaitu sebagai berikut:

1. *Zero-center input*

Proses ini menghitung rata-rata input dan standar deviasi (*input mini batch*). Maka dari itu disebut *batch normalization* karna menormalkan data pada *minibatch* untuk mempercepat proses *training* dan meningkatkan *learning rates* pada saat pembuatan model (Ioffe & Szegedy, 2015). Terdapat empat proses yaitu menghitung *mini batch mean* dan *mini batch varian*, normalisasi, dan *scale and shift*.

Rumus 1 *mini batch mean*

$$\mu B \leftarrow \frac{1}{m}\sum_{i=1}^{m} xi \qquad \text{.....(1)}$$

Rumus 2 *mini batch varian* :

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(xi - \mu B)^2 \qquad \text{.....(2)}$$

Keterangan :

m = jumlah kolom pixel

μB = nilai rata-rata dari batch

σB = standar deviasi dari *minibatch*

2. Rumus (3) Normalisasi *Input*

$$\hat{x}i \leftarrow \frac{xi - \mu B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad \text{.....(3)}$$

Keterangan :

$\hat{x}$ = *zero-centered* dan normalisasi *input*

ϵ = *variabel* tambahan, biasanya bernilai 10^{-8} untuk menghindari pembagian dengan 0 jika σ = 0 dalam beberapa estimasi.

2.2.4 *ReLU Activation*

ReLU (*Rectified Linear Unit*) merupakan fungsi linear yang digunakan untuk mengubah nilai x menjadi 0 jika x bernilai asal negatif dan tetap mempertahankan nilai x jika bernilai lebih dari 0. Tujuannya adalah untuk mempercepat pelatihan. Fungsi *ReLu* pertama kali diperkenalkan oleh Geoffrey Hinton dan Vinod Nair untuk menggantikan fungsi *aktivasi sigmoid* (Agarap, 2018). Fungsi *ReLU* dirumuskan pada rumus (4). Pada Gambar 6 diilustrasikan operasi aktivasi *ReLU*.

$$f(x\text{i}) = \max(0, x\text{i}) \begin{cases} xi, if\ xi \geq 0 \\ 0, if\ xi < 0 \end{cases} \quad \text{.....(4)}$$

Keterangan:

$f(x\text{i})$ = nilai dari *ReLU activation*

Xi = nilai matriks dari citra

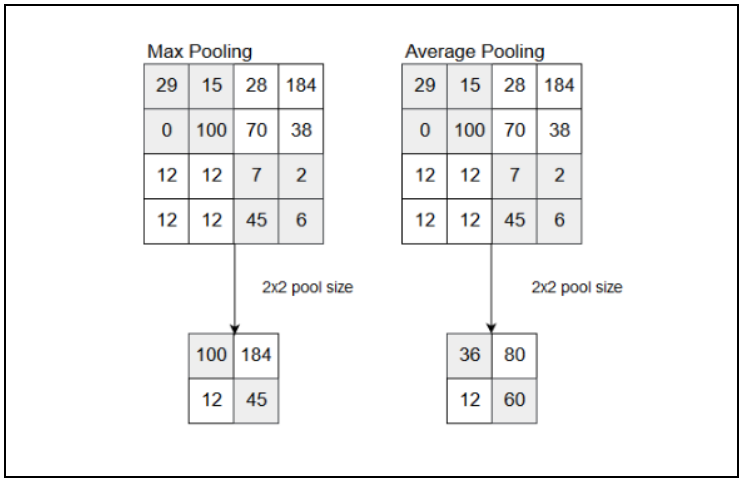
Sumber : (Nwankpa et al., 2018)

1,2	**2,5**	**2**		**1,2**	**2,5**	**2**
-2,1	**2**	**-1,2**		**0**	**2**	**0**
-1,1	**4**	**5**		**0**	**4**	**5**

Gambar 6 Ilustrasi Operasi *ReLU Activation*

2.2.5 *Pooling*

Pooling layer adalah layer yang bertugas untuk mengurangi resolusi dari suatu gambar yang telah diproses, *pooling layer* berfungsi untuk mengurangi *noise* yang ada dalam gambar tersebut, terdapat dua jenis *pooling* yang ada, yaitu *max pooling* dan *average pooling* (Mahmud et al., 2019). *Pooling* layer bertujuan untuk secara bertahap mengurangi dimensi citra, dan dengan demikian mengurangi jumlah parameter dan kompleksitas komputasi model (O'Shea & Nash, 2015)..

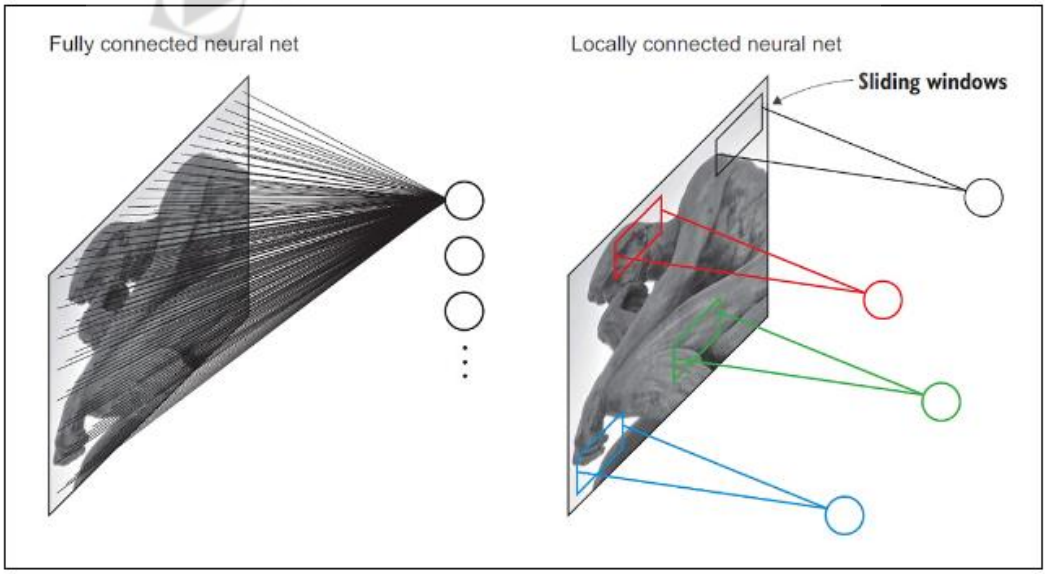


Gambar 7 Ilustrasi *Max Pooling* dan *Avg Pooling*

Sumber: (Mahmud et al., 2019)

2.2.6 *Fully Connected*

Lapisan terakhir adalah lapisan yang sepenuhnya terhubung, cara kerjanya mengambil gambar yang sudah dilakukan filter tingkat tinggi dan menerjemahkannya ke label dengan kategori. Lapisan ini mendapatkan *input* dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu (Elgendy, 2019b). Fungsi dari lapisan ini adalah untuk menyatukan semua *node* menjadi satu dimensi seperti yang diilustrasikan pada Gambar 8.



Gambar 8 Ilustrasi Fully Connected Layer

Sumber : (Elgendy, 2019b)

2.2.7 *Softmax*

Digunakan untuk mendapatkan hasil klasifikasi. Nilai *softmax* berada ada *range* 0 – 1 dan memiliki jumlah 1 jika seluruh elemennya dijumlahkan. Fungsi ini biasanya digunakan diujung *layer* dari *fully connected layer* untuk klasifikasi lebih dari dua kelas yang digunakan pada CNN untuk mendapat nilai probabilitas suatu objek terhadap kelas yang ada (Nwankpa et al., 2018). Rumus (5) *softmax*

$$\hat{y}=\frac{\exp(xi)}{\sum_j exp^{(xi)}} \quad \text{.....(5)}$$

Keterangan :

$\hat{y}_l$ = Hasil dari aktifasi fungsi *softmax*

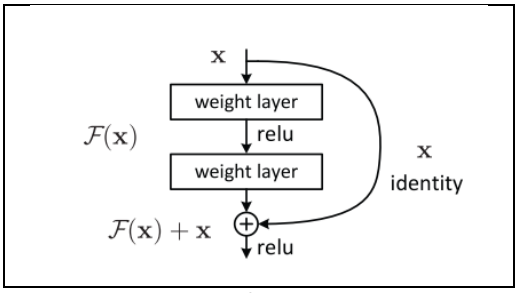
Xi = kelas ke i (i=1,2..dst). pada penelitian ini memakai 4 kelas

j = nilai dari vektor

Sumber : (Nwankpa et al., 2018)

2.3 Arsitektur *ResNet* 50

ResNet, kependekan dari *Residual Networks* adalah jaringan saraf klasik. Model ini adalah pemenang tantangan *ImageNet* pada tahun 2015. Terobosan mendasar dengan *ResNet* adalah memungkinkan untuk melatih jaringan saraf yang sangat dalam dengan 150+ lapisan. Sebelum pelatihan *ResNet*, jaringan saraf yang dalam sangat sulit karena masalah hilangnya gradien. Arsitektur *Resnet* menunjukkan bahwa *neural network* ini lebih mudah dioptimalkan, dan dapat memperoleh akurasi dari kedalaman jauh yang jauh meningkat (He et al., 2016). Gambar 10 menunjukkan lapisan *residual*. *ResNet* merupakan solusi dari *neural network* yang dalam, semakin dalam pelatihan maka semakin rumit dan kedalaman sangat penting untuk pelatihan agar parameter atau neuron dapat mengingat atau menyimpan nilai pelatihan yang optimal. Agar pelatihan tidak jenuh dan menghindari kesalahan yang cukup tinggi maka terdapat fungsi *resnet* yaitu penambahan identitas lapisan seperti pada gambar 9.



Gambar 9 Identity Block ResNet

(He et al., 2016)

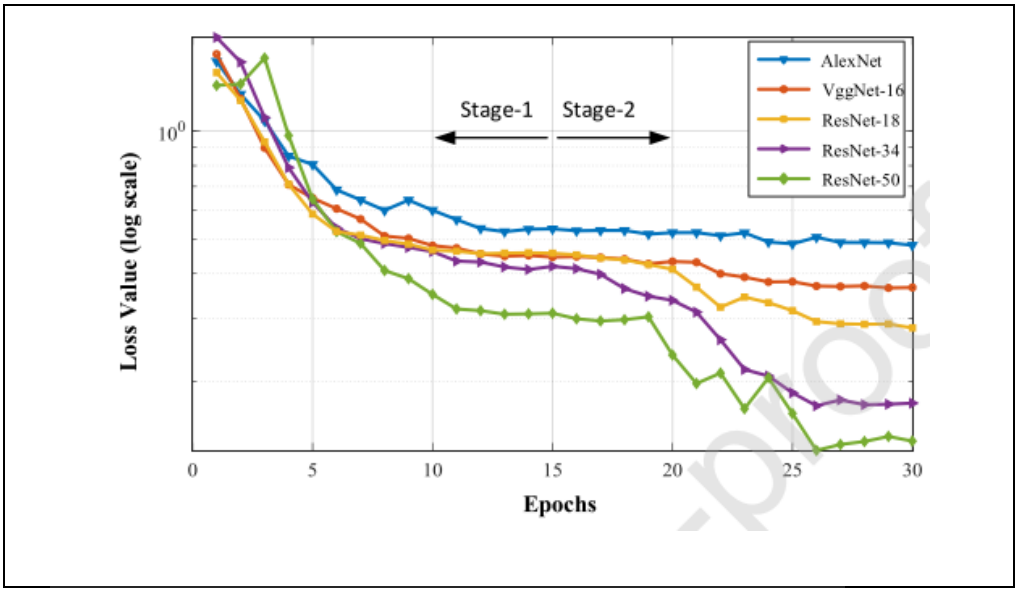
Solusi ini menunjukkan bahwa model ini mengurangi kesalahan pelatihan seperti yang telah dilakukan penelitian oleh (Talo et al., 2019) dari perbandingan model yang dilakukan resnet-50 memiliki nilai eror terkecil dan akurasi terbaik dari model lainnya. *ResNet* 50 merupakan arsitektur CNN terbaik, itu terlihat pada penelitian yang dilakukan (Talo et al., 2019) yaitu melakukan penelitian tentang klasifikasi penyakit otak dengan gambar MRI. Arsitektur yang digunakan adalah *AlexNet*, *Vgg*-Net 16, *ResNet*-18, *ResNet*-34 dan *ResNet*-50. *ResNet*-50 memberikan nilai *loss* terkecil daripada dengan arsitektur CNN yang lain ditampilkan pada gambar 10. Dan akurasi yang sangat baik dengan mencapai

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				

Gambar 10 Arsitektur ResNet

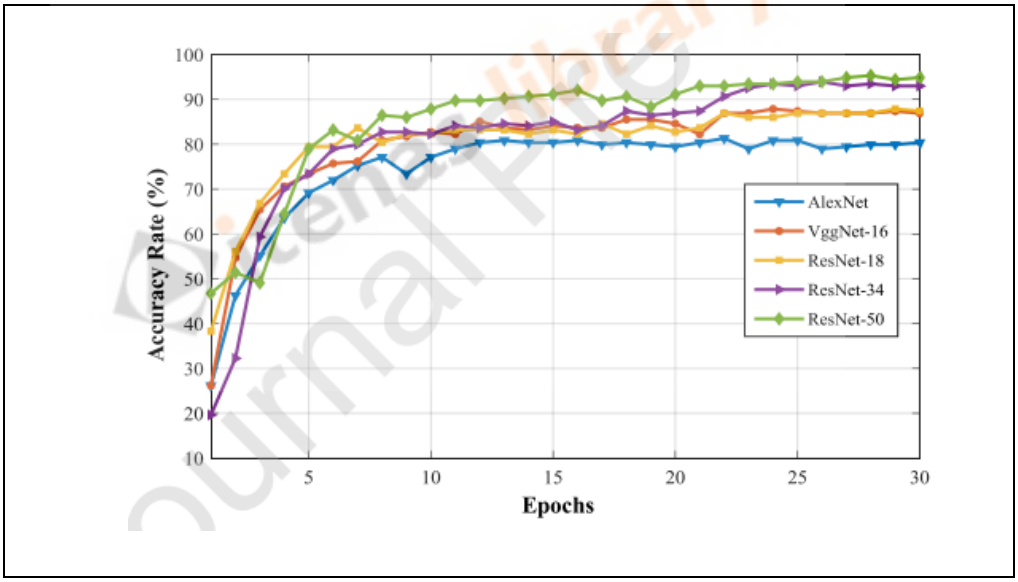
(He Kaiming., et al, 2016)

95.23% terbaik di antara arsitektur CNN yang lain. Perbandingan Nilai tersebut dapat dilihat pada gambar 11. Namun beberapa bermasalah dalam penelitian ini adalah sedikitnya jumlah gambar pasien penyakit otak. Sehingga membutuhkan meningkatkan klasifikasi.



Gambar 11 Perbandingan Nilai Loss

Sumber: (Talo et al., 2019)



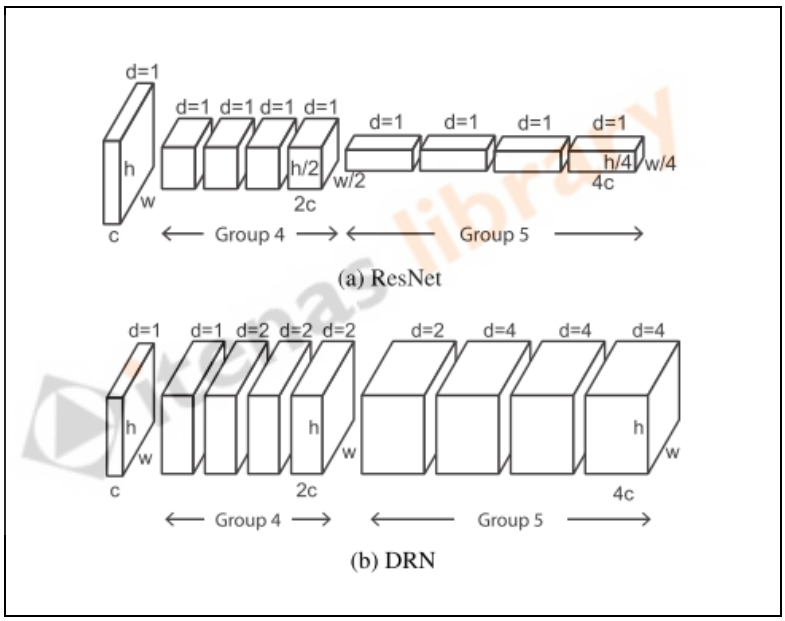
Gambar 12 Perbandingan Nilai Akurasi

Sumber: (Talo et al., 2019)

2.4 *Residual Network* 50 *Dilated*

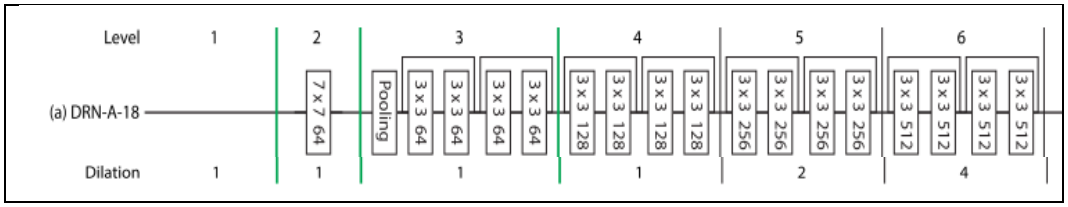
Menurut (F. Yu et al., 2017) *neural network* bagian klasifikasi gambar secara progresif mengurangi resolusi gambar yang diwakili oleh *fitur* kecil yang menyimpan sedikit informasi gambar. Meskipun jaringan konvolusi berjalan dengan baik, penghapusan ketajaman gambar yang paling lengkap dapat mencegah model ini mencapai akurasi yang lebih tinggi, misalnya dengan mempertahankan

kontribusi benda kecil dan tipis yang mungkin penting untuk memahami gambar dengan benar. Mempertahankan semacam itu mungkin tidak penting dalam konteks klasifikasi digit tulisan tangan, di mana satu objek mendominasi gambar, tetapi dapat membantu dalam analisis pemandangan alam yang kompleks di mana banyak objek dan konfigurasi relatifnya harus diperhitungkan. *Dilated Residual Network* (DRN) menunjukkan menghasilkan kinerja klasifikasi gambar yang ditingkatkan. Secara khusus, DRN menghasilkan akurasi yang lebih tinggi dalam klasifikasi *Image*-Net daripada model yang tidak dilatasi, tanpa peningkatan kedalaman atau kompleksitas model. DRN yang dikonversi memiliki jumlah lapisan dan parameter yang sama dengan *ResNet* asli.

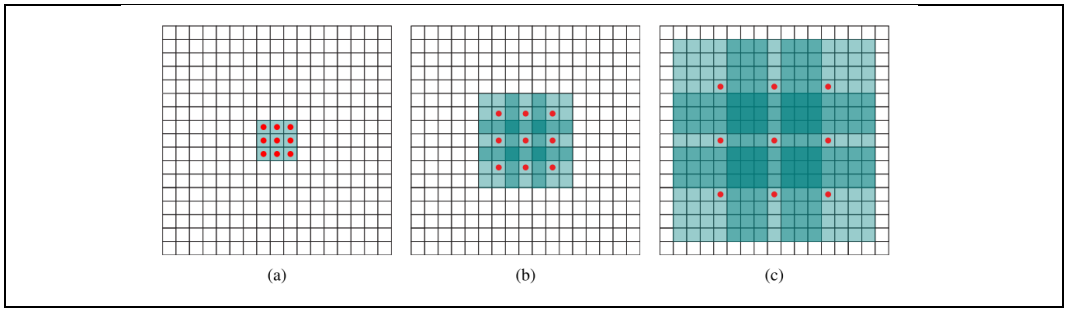


Gambar 13 Conversi ResNet menjadi Dilated ResNet

Gambar 14 menjelaskan rekonstruksi *ResNet dilated* sama seperti *resnet* asli hanya saja menghapus *max pooling* dengan *pooling* kemudian menambahkan dilatasi pada *conv_blok* 4 dan 5.



Gambar 14 Rekontruksi ResNet menjadi DRN



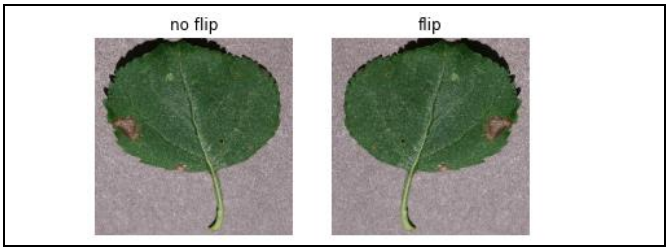
Gambar 15 *Dilation Rate*

(Yu.F., et al, 2017)

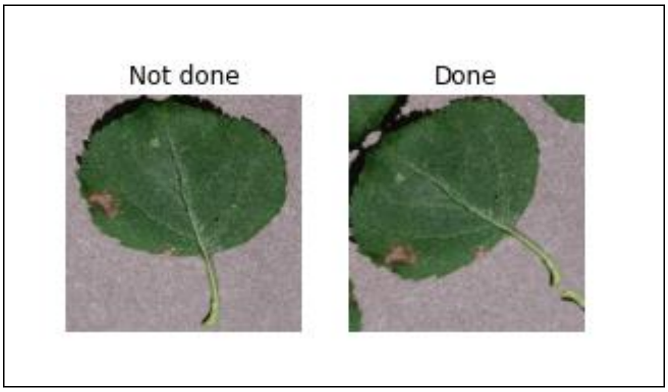
Gambar 15 menggambarkan konvolusi ketika diberikan *dilation rate*, gambar 15 (a) ukuran 3x3 filter dengan *dilation rate* 1, gambar 15 (b) ukuran filter sama 3x3 dengan *dialtion rate* 2 ukuran filter menjadi melebar dan ukuran filter berubah menjadi 7x7. Gambar 15 (c) ukuran filter 3x3 dengan *dilation rate* 4 ukuran filter menjadi 15x15.

2.5 *Preprocessing*

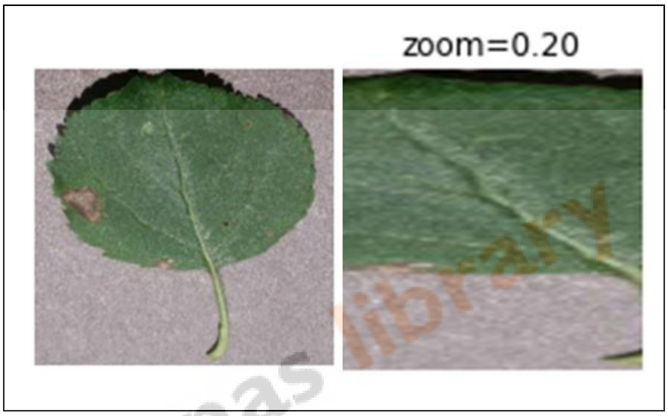
Preprocessing merupakan hal yang sangat penting untuk *deep learning* terutama pada *neural network*. Sebelum melakukan pelatihan gambar atau data penting untuk melakukan *preprocessing*. Menurut (Chollet, 2018) *Preprocessing* data bertujuan untuk membuat data mentah lebih mudah diterima oleh *Neural Networks*. *Preprocessing* yang diambil hanya tiga proses, karena beberapa proses *preprocessing framework* keras tidak tersedia pada *framework* fast.ai. Beberapa proses *preprocessing* digunakan oleh penelitian (Chollet, 2018) yaitu *flip*, *zoom* sebesar 0,2 dan rotasi sebesar 40. Hal tersebut telah terbukti dapat mengurangi *overfitting*. Contoh *flip* gambar ditampilkan pada gambar 16, *zoom* gambar ditampilkan pada gambar 17 dan rotasi gambar di tampilkan pada gambar 18.



Gambar 16 *Flip*



Gambar 17 Rotasi



Gambar 18 *Zoom*

Kemudian ditambahkan normalisasi *imagenet* dengan rumus (6) dan rumus (7) sebagai berikut (Ponakala, 2013).

Mean = [0.485, 0.456, 0.406]	……..………….……...….(6)

Nilai Mean R, G dan B pada Rumus (7) merupakan *Normalize ImageNet*.

std = [0.229, 0.224, 0.225]	....……………..(7)

Nilai std R, G dan B pada Rumus (8) merupakan *Normalize ImageNet*.

Normalisasi adalah komponen yang efektif dalam pembelajaran mendalam, memajukan banyak bidang penelitian seperti pemrosesan bahasa alami, visi komputer, dan pembelajaran mesin (Luo et al., 2019). Normalisasi yang digunakan adalah *ImageNet Stats* yaitu menggunakan nilai RGB *mean* dan standar deviasi. Nilai *mean* RGB di rumus (6) dan rumus (7) merupakan nilai standar deviasi *ImageNet*.

Proses resize dilakukan ukuran gambar dataset yang di tampilkan tabel 5 seluruhnya berukuran 256 x 256, kemudian diubah menjadi 224 x 224 hal tersebut sama seperti yang dilakukan (Simonyan & Zisserman, 2015) karena ukuran tersebut bisa benar-benar menjangkau sisi terkecil dari gambar pelatihan. Perhitungan resize rumus (8) dan rumus (9).

$$\text{Scale} = \sqrt{Pixel\ /NewPixel} \quad \text{.....(8)} \qquad \text{Nh} = \frac{Height}{Scale} \quad \text{.....(9)}$$

$$\text{Nw} = \frac{Weight}{Scale} \quad \text{.....(10)}$$

Nh = New Height

Nw = New Weight

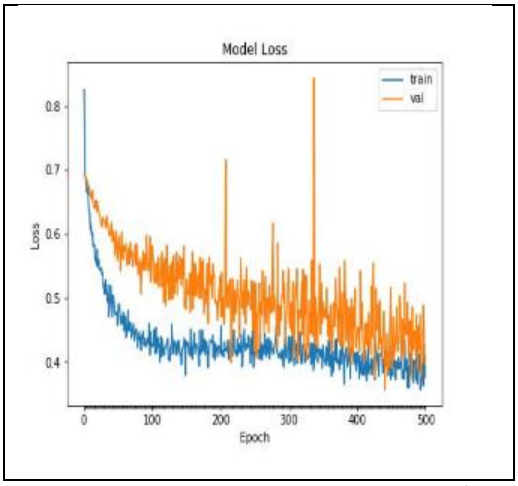
Sumber: (Suresh et al., 2013)

2.6 *Overfitting* dan *Underfitting*

Overfitting adalah keadaan ketika kinerja model bernilai baik untuk *training* tetapi buruk pada prediksi data (Putra, 2019). *Overfitting* terjadi jika jarak antara nilai kesalahan yang dihasilkan data pelatihan (*training loss*) dan uji (*valid loss*) terlalu besar. Model yang dibuat terlalu fokus pada *training* data set tertentu, hingga tidak bisa melakukan prediksi dengan tepat jika diberikan data set lain yang serupa

Underfitting adalah saat model tidak dapat mengurangi *error* untuk set validasi dan pelatihan (Smith, 2018). *Underfitting* terjadi ketika suatu model tidak dapat memperoleh nilai kesalahan yang rendah pada pelatihan.

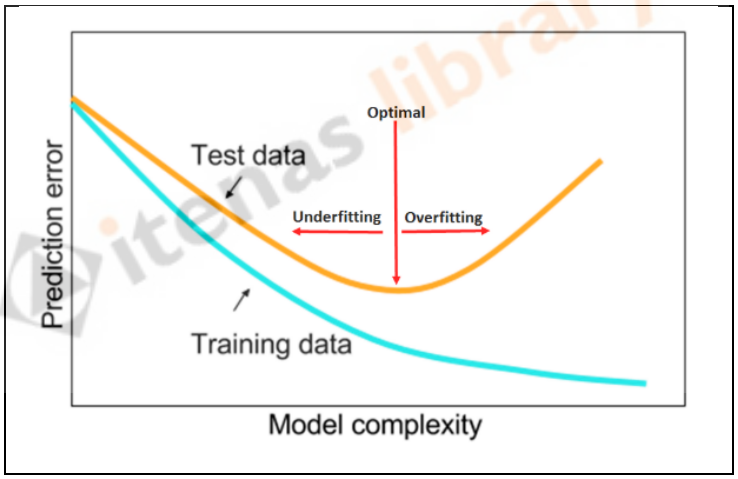
Gambar 20 menjelaskan garis biru *training loss* dan garis oranye *validasi loss*. Pada gambar 20 terlihat semakin banyak iterasi atau *epoch* maka *training loss* semakin kecil, tetapi untuk *valid loss* jika nilai *loss* tinggi data termasuk *underfitting* dan jika nilai *training loss* dan *valid loss* berada dinilai terkecil dan memiliki nilai selisih tidak terlalu jauh maka bisa dikatakan *training* tersebut optimal seperti gambar 19.



Gambar 19 Contoh *training* optimal

(Dogo et al., 2018)

Tetapi jika data telah optimal kemudian nilai *valid loss* naik secara signifikan maka data tersebut bisa dikatakan *overfitting*.



Gambar 20 *Overfitting* dan *Underfitting*

(Smith, 2018)

2.7 Daun Apel

Apel (*Mallus sylvestris Mill*) merupakan tanaman buah tahunan yang berasal dari daerah Asia barat dengan iklim subtropis. Apel adalah salah satu buah yang banyak diminati oleh masyarakat di Indonesia, karena memiliki rasa yang enak dan banyak mengandung vitamin sehingga bermanfaat.

Gambar 21 Daun apel sehat

(*PlantVillage-Dataset*, 2017)

Terdapat tiga jenis penyakit daun apel pada dataset plant-village yaitu;

- *Apple Scab*

Apple scab disebabkan oleh jamur *Venturia inaequalis* dengan memiliki gejala-gejala bercak-bercak hijau di kedua sisi daun yang seiring berkembangnya penyakit, daun menjadi berwarna hitam keunguan (Wicaksono et al., 2020).

Gambar 22 Penyakit daun *apel scab*

(*PlantVillage-Dataset*, 2017)

- *Apple Black Rot*

Apple Black Rot disebabkan oleh jamur *Botryosphaeria obtusa* dengan memiliki gejala-gejala Bercak-bercak ungu di permukaan daun dengan diameter 0,2 sampai 0,125 inc (Wicaksono et al., 2020).

Gambar 23 *Penyakit daun apel black rot (PlantVillage-Dataset*, 2017)

- *Cedar Apple Rust*

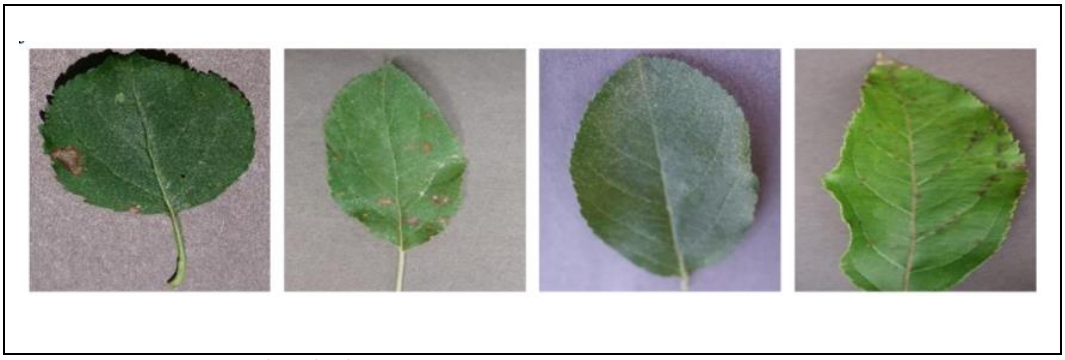
Apple Cedar Rust disebabkan oleh jamur *Gymnosporangium juniperi-virginianae* dengan memiliki gejala-gejala Bercak-bercak coklat dan membuat daun menjadi rapuh (Wicaksono et al., 2020).

Gambar 24 Penyakit *Cedar Apple Rust* (*PlantVillage-Dataset*, 2017)

Pengumpulan data dilakukan dengan sekunder yaitu sebuah data yang langsung diperoleh dari peneliti berupa data set gambar beserta dokumen (*journal*). Terdapat dua data set gambar dan dua *journal* yang berbeda. Dataset yang didapat dari (Hughes & Salathe, 2015) langsung diambil oleh ahli patologi tanaman dengan tiga kelas penyakit daun apel dan satu kelas daun apel sehat. Semua gambar diambil di stasiun penelitian eksperimen yang terkait dengan *Land Grant Universities in the USA (Penn State, Florida State, Cornell, and others)*. Diambil pada website PlantVillage (tersedia di www.plantvillage.org) dan juga tersedia di (www.kaggle.com/abdallahalidev/plantvillage-dataset) di download pada 19 Maret 2020. Pemilihan setiap kelas daun dilakukan langsung oleh ahli patologi untuk menentukan status penyakit dan para ahli bekerja langsung di lapangan. Jumlah dan sampel gambar ditujukan pada Tabel 1 dan Gambar 25.

Tabel 1 Data set Gambar *Training*

NO	Nama Penyakit Daun Apel	Jumlah Citra
1	*Apple Scab*	630
2	*Black Rot*	621
3	*Cedar Apple Rust*	275
4	*Health*	1645
Total Citra		**3171**



Gambar 25 dari kiri Apple Black Rot, Apple Cedar Rust, Apple Healthy dan Apple Scab

Pada data set gambar yang dimiliki (Hughes & Salathe, 2015) terdapat data set yang telah di *augmentasi* dengan cara rotasi 90^0, 45^0, 180^0. *Augmentasi* dalam

pelatihan cukup penting karena dapat memvariasikan *input* gambar, selain memvariasikan data pelatihan menurut (Chollet, 2018) tujuan dari *augmentasi* adalah menghindari *overfitting* dari pada data *input* yang sedikit. Dataset pada tabel 2 tersedia pada (www.kaggle.com/vipoooool/new-plant-diseases-dataset) didownload pada tanggal 19 Maret 2020. Jumlah dan sampel data set gambar *training augmentasi* ditampilkan pada Tabel 2 dan Gambar 26.

Tabel 2 Data set Gambar *Training Augmentasi*

NO	Nama Penyakit Daun Apel	Jumlah Citra
1	*Black Rot*	2484
2	*Cedar Apple Rust*	2200
3	*Health*	2510
4	*Apple Scab*	2520
Total Citra		**9467**

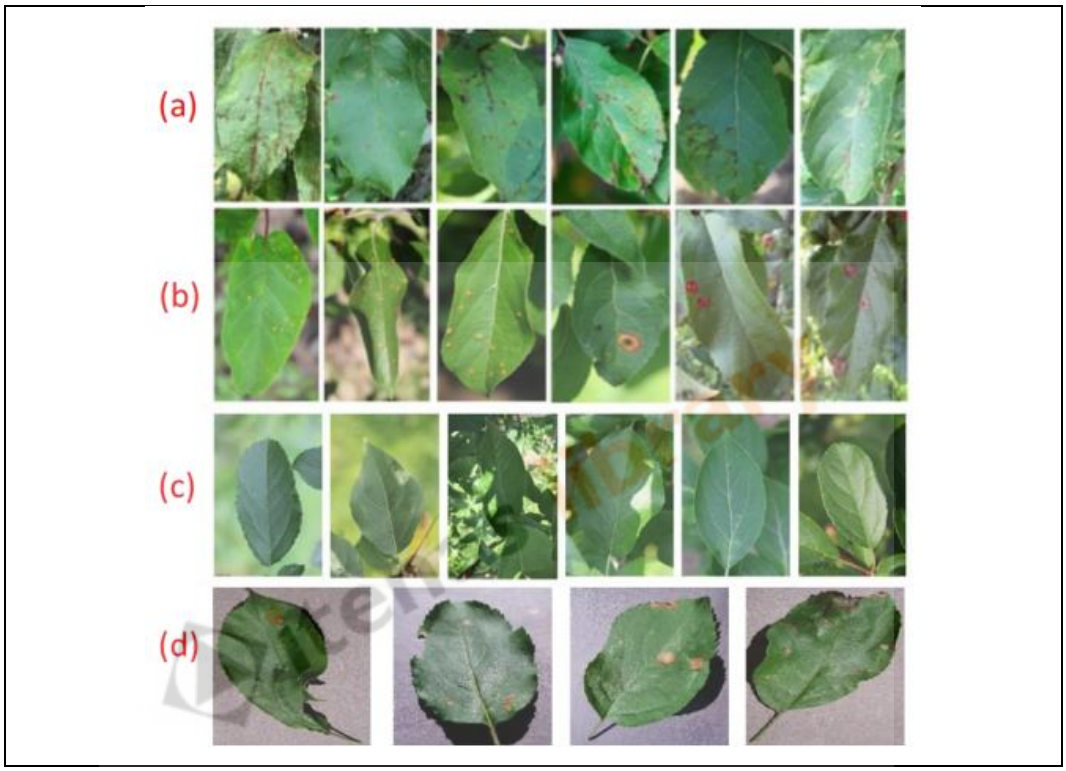
Gambar 26 Augmentasi Rotasi

Kemudian untuk data set gambar uji digunakan untuk mengukur hasil data set *training*, seberapa besar hasil *training* yang dilakukan. Jumlah dan gambar sampel data set gambar uji yang didapat pada (Moghbelli et al., 2020) tersedia pada (https://www.kaggle.com/c/plant-pathology-2020-fgvc7/data) di download pada bagian data diambil pada 30 Mei 2020. Jumlah dan gambar ditampilkan pada tabel 3 dan gambar 27.

Tabel 3 Data set Gambar Uji

NO	Nama Penyakit Daun Apel	Jumlah Citra
1	*Apple Scab*	592

2	*Multiple Disease*	91
3	*Cedar Apple Rust*	622
4	*Health*	516
Total Citra		**1821**



Gambar 27 (a) daun apel dengan penyakit Apple Scab, Gambar 27 (b), apel dengan penyakit Cedar Rust, Gambar 27 (c) daun Apel Sehat dan Gambar 27 (d) Apple Black Rot

2.8 Pengujian Model

Dalam rangka mengukur kinerja dari sebuah sistem deteksi objek sama halnya dengan pengujian pada sistem *information retrieval*, yaitu dengan mengukur *precision*, *recall*, dan *accuracy*. *Precision* merupakan bagian dari dokumen/citra yang terambil oleh sistem dan terbukti benar/relevan. Sedangkan *recall* merupakan bagian dari dokumen/citra relevan yang terambil oleh sistem. Dan *accuracy* didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai actual

(Christopher D. Manning & Hinrich, 2009). Rumus tersebut sama dengan yang dilakukan (Mohtar et al., 2019).

Tabel 4 Confusion Matriks

Kejadian	**Positif**	**Negatif**
	True Positive (TP)	False Negative (FN)
	False Positive (FP)	True Negative (TN)

$$precision = \frac{TP}{TP + FP} \quad \text{.....(11)} \qquad recall = \frac{TP}{TP + FN} \quad \text{.....(12)}$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad \text{.....(13)}$$

Dimana:

TP = Merupakan jumlah kelas yang benar/relevan yang terdeteksi oleh sistem

TN = Merupakan jumlah yang bukan kelas yang benar tidak didapat.

FP = Merupakan jumlah kelas yang salah dideteksi oleh sistem

FN = Merupakan jumlah kelas yang tidak didapat oleh sistem

sumber : (Mohtar et al., 2019)

Precision untuk mengukur pola positif yang berhasil diprediksi dengan benar (TP) dari total pola prediksi dalam kelas positif baik itu data yang berhasil diprediksi dengan benar (TP) dengan data yang berhasil diprediksi bukan kelas positif (FP). Recall, untuk melihat ukuran keberhasilan dengan membagi data yang berhasil diprediksi dengan benar (TP) dibagi dengan jumlah dari data yang berhasil diprediksi dengan benar (TP) dan data yang berhasil diprediksi negatif salah (FN). Akurasi digunakan untuk melihat gambaran seberapa akurat sistem mengklasifikasikan dengan tepat. (Jumeilah, 2018). Untuk menghitung pengujian model dengan empat kelas rumus tersbut di tampilkan pada rumus (14).

$$precision = \frac{\text{TP}}{\sum_{i=1}^{l}(\text{TP} + \text{FP})} \quad (14)$$

$$recall = \frac{\text{TP}}{\sum_{i=1}^{l}(\text{TP} + \text{FN})} \quad (15)$$

$$accuracy = \frac{\sum_{i=1}^{l}\frac{tp+tn}{tp+fn+fp+tn}}{l} \quad (16)$$

Sumber: (Sokolova & Lapalme, 2009)