

**IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI BUNGA**

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Strata Satu

Program Studi Sistem Informasi



Disusun oleh :

REZKY FIRMANSYAH

11150930000029

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH

JAKARTA

2021 M/1442 H

HALAMAN JUDUL

**IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI BUNGA**

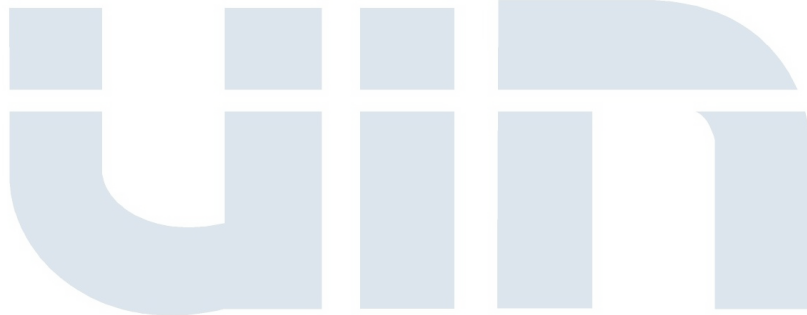
SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Strata Satu
Program Studi Sistem Informasi

Disusun oleh :

REZKY FIRMANSYAH

11150930000029



PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH

JAKARTA

2021 M/1442 H

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI *DEEP LEARNING* MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK* UNTUK KLASIFIKASI BUNGA

Disusun oleh :

REZKY FIRMANSYAH

11150930000029

Menyetujui,

Dosen Pembimbing I

Dosen Pembimbing II



Dr. Omroful Aini, M.T.

NIP. 19730325 200901 2 001



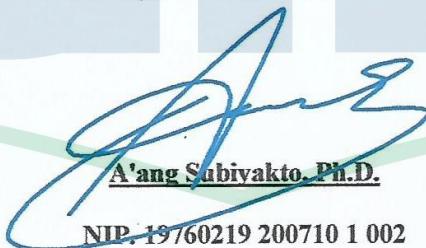
Zulfiandri, MMSI

NIP. 19700130 200501 1 003

Mengetahui,

Ketua Program Studi Sistem Informasi Fakultas Sains dan Teknologi

UIN Syarif Hidayatullah Jakarta



A'ang Subiyakto, Ph.D.

NIP. 19760219 200710 1 002

PENGESAHAN UJIAN

Skripsi yang berjudul *Implementasi Deep Learning Menggunakan Convolutional Neural Network* untuk Klasifikasi Bunga telah diuji dan dinyatakan lulus dalam sidang munaqosyah Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta pada hari Jumat tanggal 17 April 2020. Skripsi ini telah diterima sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu (S1) Program Studi Sistem Informasi.

Meneytujui,

Penguji I



Yuni Sugiarti, M.Kom

NIDN. 2006067602

Penguji II



Eva Khudzaeva, M.Si

NIDN. 0306108301

Dosen Pembimbing I



Dr. Omeratul Aini, M.T.

NIP. 19730325 200901 2 001

Dosen Pembimbing II



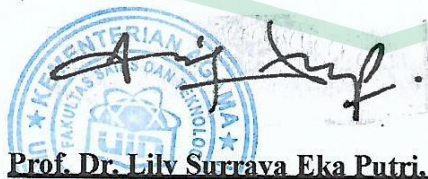
Zulfiandri, MMSI

NIP. 19700130 200501 1 003

Mengetahui,

Dekan Fakultas Sains dan Teknologi

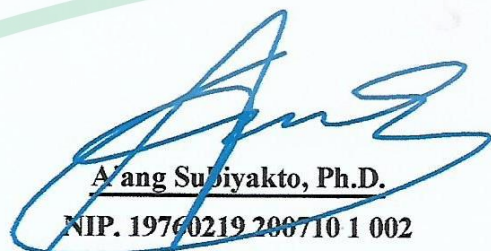
Ketua Prodi Sistem Informasi



Prof. Dr. Lily Surrava Eka Putri,

M.Env.Stud.

NIP. 1969040 4200501 2 005



A'ang Subiyakto, Ph.D.

NIP. 19760219 200710 1 002

PERNYATAAN

DENGAN INI SAYA MENYATAKAN BAHWA SKRIPSI INI BENAR-BENAR HASIL KARYA SENDIRI DAN BELUM PERNAH DIAJUKAN SEBAGAI SKRISI ATAU KARYA ILMIAH PADA PERGURUAN TINGGI ATAU LEMBAGA MANAPUN.

Jakarta, 09 Maret 2021



REZKY FIRMANSYAH

11150930000029

ABSTRAK

REZKY FIRMANSYAH – 11150930000029, Implementasi *Deep Learning* Menggunakan *Convolutional Neural Network* Untuk Klasifikasi Bunga di bawah bimbingan **Qurrotul Aini** dan **Zulfiandri**.

Bunga merupakan struktur reproduksi yang ditemukan pada tanaman berbunga (*angiospermae* atau *magnoliophyta*). Terdapat 295.383 spesies atau jenis bunga dengan beragam bentuk, warna dan struktur. Untuk dapat mengklasifikasi jenis bunga dari semua input (citra bunga) dengan *machine learning*, maka dibutuhkan teknik pembelajaran *supervised learning*. Terdapat beberapa penelitian untuk mengklasifikasikan bunga dengan menggunakan *machine learning*, namun kinerja akurasi model klasifikasi bunga dalam penelitian sebelumnya belum mencapai hasil yang sangat baik. Selain itu, masalah yang dihadapi dalam mengklasifikasi bunga adalah karena bunga memiliki warna yang beragam serta terkadang terlihat mirip dengan warna latar belakang objek bunga, dan untuk mengidentifikasi objek memiliki latar belakang warna juga sulit untuk dilakukan. Dalam penelitian ini teknik *convolutional neural network* digunakan untuk mengklasifikasi bunga. Tujuan dari penelitian ini adalah untuk membuat model CNN untuk mengklasifikasikan bunga, serta menguji dan mengukur kinerja dengan model ANN dan model SVM. Metode penelitian ini menggunakan menggunakan model yang dikembangkan dari Kozłowski et al (2019) dan Steinbrener et al (2019), dengan menggunakan CNN dan CNN *transfer learning*. Data yang digunakan adalah oxford17 dengan 17 jenis bunga dan oxford102 dengan 102 jenis bunga. Hasil penelitian ini mendapatkan akurasi 60% dan 84% dengan menggunakan pendekatan transfer learning. Sedangkan untuk bunga pada oxford102 mendapatkan hasil akurasi sebesar 42% dan 64% untuk akurasi dengan *transfer learning*. CNN dapat mengungguli SVM dan ANN pada *dataset* Oxford17, namun pada *dataset* oxford102 CNN tidak dapat mengungguli ANN, tetapi dapat mengungguli SVM. Manfaat penelitian ini menghasilkan model *machine learning* yang mampu untuk mengklasifikasikan bunga secara optimal.

Kata kunci: Bunga, *Deep Learning*, *Convolutional Neural Network*, Klasifikasi, Implementasi.

BAB 1-5 + 93 Halaman + xiv Halaman + 56 Gambar + 9 Tabel + Daftar Pustaka + Lampiran
Pustaka Acuan (60, 1990-2019)

KATA PENGANTAR

Assalamualaikum Wr. Wb

Puji dan syukur kehadiran Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga peneliti dapat menyelesaikan skripsi ini dengan baik. Shalawat serta salam tak lupa selalu tucurahkan kepada junjungan nabi besar kita yaitu Nabi Muhammad SAW yang telah memberikan tuntunan dan petunjuk kepada umat manusia menuju kehidupan dan peradaban, serta para keluarga, serta para sahabat Nabi. Penulis menyadari bahwa dalam menyelesaikan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, perkenankanlah penulis untuk dapat mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. Lily Surayya Eka Putri, M.Env.Stud selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah.
2. Bapak A'ang Subiyakto, Ph.D. selaku Ketua Program Studi Sistem Informasi Fakultas Sains dan Teknologi dan Bapak Nuryasin, M.Kom. selaku Sekretaris Program Studi Sistem Informasi Fakultas Sains dan Teknologi.
3. Ibu Dr. Qurrotul Aini, M.T. sebagai Dosen Pembimbing I dan Bapak Zulfiandri, MMSI sebagai Dosen Pembimbing II yang tak pernah lelah memberikan ilmu dan bimbingannya sehingga skripsi ini dapat terselesaikan.

4. Dosen-dosen Program Studi Sistem Informasi yang telah memberikan ilmunya selama penulis menuntut ilmu di UIN Jakarta.
5. Kedua orang tua penulis, Bapak Ade Ramli dan Ibu Enung Nurmalasari yang selalu memberikan semangat, dukungan, dan doa sehingga penulis dapat menyelesaikan laporan ini.
6. Kepada Gerry, Rahmadi, Arga, Fathur, Abdur, Anka, Rafi yang telah memberikan motivasi, dukungan, dan ilmu kepada penulis.
7. Teman-teman Program Studi Sistem Informasi yang telah menjadi motivasi penulis untuk menyelesaikan laporan ini.
8. Serta seluruh pihak-pihak yang terkait dan telah berjasa membantu dalam proses penyelesaian skripsi ini yang tidak dapat disebutkan satu persatu namun tidak mengurangi sedikitpun rasa terimakasih dari penulis.

Akhir kata, peneliti menyadari bahwa masih terdapat kekurangan dalam penyusunan skripsi ini, untuk itu saran dan kritik yang sifatnya membangun sangat peneliti harapkan. Peneliti berharap skripsi ini dapat bermanfaat bagi penulis khususnya dan bagi para pembaca pada umumnya, Aamin.

Wassalamualaikum Wr.Wb.

Jakarta, 09 Maret 2021

Rezky Firmansyah

11150930000029

DAFTAR ISI

HALAMAN JUDUL.....	ii
LEMBAR PENGESAHAN SKRIPSI	iii
PENGESAHAN UJIAN	iv
PERNYATAAN.....	v
ABSTRAK.....	vi
KATA PENGANTAR	vii
DAFTAR ISI.....	ix
DAFTAR GAMBAR	xiii
DAFTAR TABEL.....	xvi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	5
1.3 Perumusan Masalah.....	5
1.4 Batasan Masalah.....	5
1.5 Tujuan Penelitian.....	6
1.6 Manfaat Penelitian.....	6
1.7 Metode Penelitian.....	7

1.7.1	Metode Pengumpulan Data	7
1.7.2	Pemodelan dengan <i>Machine Learning</i>	7
1.8	Sistematika Penulisan	8
BAB 2 TINJAUAN PUSTAKA		10
2.1	<i>Artificial Intelligence</i>	10
2.1.1	<i>Machine Learning</i>	11
2.1.2	<i>Deep Learning</i>	14
2.1.3	Klasifikasi	19
2.2	<i>Convolutional Neural Network (CNN)</i>	19
2.2.1	Lapisan <i>Convolutional Neural Network</i>	21
2.2.2	Kelebihan CNN	27
2.2.3	Model CNN Kozłowski	27
2.2.4	Model CNN Steinbrener	28
2.3	Augmentasi Data	28
2.4	<i>Confusion Matrix</i>	29
2.5	<i>Nadam Optimization</i>	31
2.6	<i>Transfer Learning</i>	33
2.7	<i>Deep Residual Networks (ResNets)</i>	34
2.8	Bunga	34
2.9	Citra Digital	36

2.10	<i>Computer Vision</i>	38
2.10.1	<i>Pattern Recognition</i>	38
2.10.2	<i>Object Recognition</i>	39
2.11	<i>Tools</i>	39
2.11.1	Python	39
2.11.2	Tensorflow	39
2.11.3	Jupyter Notebooks.....	40
2.12	Penelitian Sejenis	40
BAB 3 METODE PENELITIAN		49
3.1	Populasi dan sampel penelitian	49
3.2	Kerangka Penelitian	49
BAB 4 HASIL DAN PEMBAHASAN		53
4.1	<i>Preprocessing</i> Data	53
4.1.1	Pengumpulan data citra.....	53
4.1.2	Pelabelan Data.....	56
4.1.3	Augmentasi Data.....	56
4.2	Pemodelan dan Pelatihan.....	59
4.2.1	Pemodelan <i>Convolutional Neural Network</i>	60
4.2.2	Optimasi	66
4.2.3	Pelatihan.....	66

4.2.4	Pengujian Model	78
4.3	Aplikasi Klasifikasi Bunga.....	84
4.3.1	Perancangan <i>User Interface</i>	84
4.3.2	Tahapan Pengembangan Aplikasi dan Pengujian	85
4.3.3	Simulasi dan Hasil	88
4.4	Interpretasi dan Evaluasi	92
BAB 5	PENUTUP	95
5.1	Kesimpulan.....	95
5.2	Saran	96
DAFTAR PUSTAKA		97
LAMPIRAN.....		105

DAFTAR GAMBAR

Gambar 2.1 Posisi <i>deep learning</i> pada <i>artificial intelligence</i> (Zaharchuk et al., 2018).....	11
Gambar 2.2 Struktur dari <i>neural networks</i> (Kim, 2017)	15
Gambar 2.3 Fungsi aktivasi ReLU (Moolayil, 2019).....	17
Gambar 2.4 Sebelah kiri adalah <i>neural networks</i> biasa, sebelah kanan setelah melakukan DropOut (Srivastava et al., 2014)	18
Gambar 2.5 <i>Flowchart</i> dari CNN (You et al., 2017).....	20
Gambar 2.6 Proses pada lapisan konvolusi (Kim, 2017)	23
Gambar 2.7 Proses pada lapisan konvolusi dan lapisan <i>pooling</i> (Shukla, 2018)	24
Gambar 2.8 Proses <i>max pooling</i> (Vasilev et al., 2019)	25
Gambar 2.9 Proses <i>average pooling</i> (Vasilev et al., 2019)	26
Gambar 2.10 Perbandingan algoritma optimasi untuk <i>machine learning</i> (Dozat, 2016).....	32
Gambar 2.11 Proses pada <i>transfer learning</i> (Torrey & Shavlik, 2010)	33
Gambar 2.12 Perbandingan kinerja dengan dan tanpa <i>transfer learning</i> (Torrey & Shavlik, 2010).....	34
Gambar 2.13 Struktur umum bunga (The Editors of Encyclopaedia Britannica, 2019).....	36
Gambar 2.14 Penelitian Sejenis.....	48
Gambar 3.1 Kerangka Penelitian.....	52

Gambar 4.1 Oxford17 (Nilsback & Zisserman, 2006)	55
Gambar 4.2 Oxford102 (Nilsback & Zisserman, 2008)	55
Gambar 4.3 Data yang telah diberi label	56
Gambar 4.4 Hasil gambar yang telah diargumentasi	58
Gambar 4.5 <i>Flowchart</i> pemodelan CNN	59
Gambar 4.6 <i>Flowchart training</i> CNN	60
Gambar 4.7 Model CNN yang digunakan	64
Gambar 4.8 Model CNN <i>transfer learning</i> yang digunakan	66
Gambar 4.9 Salah satu data citra pada input <i>layer</i>	68
Gambar 4.10 Label citra	69
Gambar 4.11 Input <i>layer</i>	70
Gambar 4.12 <i>Channel</i> RGB pada citra input	70
Gambar 4.13 Input <i>channel</i> merah	71
Gambar 4.14 Contoh <i>filter</i>	71
Gambar 4.15 <i>Output</i> proses konvolusi	72
Gambar 4.16 Visualisasi filter pada CNN	72
Gambar 4.17 <i>Feature map</i> pada lapisan konvolusi	73
Gambar 4.18 Pola pada <i>feature map</i>	73
Gambar 4.19 Hasil dari <i>activation function</i> pada <i>layer</i> 1 dan 2	74
Gambar 4.20 Pemilihan area objek pada citra bunga	74
Gambar 4.21 Input pada <i>pooling layer</i>	75
Gambar 4.22 <i>Output</i> pada <i>pooling layer</i>	75
Gambar 4.23 Input <i>pooling layer</i> (a), <i>output pooling layer</i> (b)	75

Gambar 4.24 Contoh salah satu <i>neural network</i>	76
Gambar 4.25 <i>Loss</i> dan akurasi pada pelatihan	78
Gambar 4.26 Model <i>loss</i> untuk <i>dataset</i> oxford17 dengan CNN (a) dan CNN <i>transfer learning</i> (b)	79
Gambar 4.27 Model <i>loss</i> untuk <i>dataset</i> oxford102 dengan CNN (a) dan CNN <i>transfer learning</i> (b)	79
Gambar 4.28 Model akurasi untuk <i>dataset</i> oxford17 dengan CNN (a) dan CNN <i>transfer learning</i> (b)	80
Gambar 4.29 Model akurasi untuk <i>dataset</i> oxford102 dengan CNN (a) dan CNN <i>transfer learning</i> (b)	81
Gambar 4.30 <i>Confusion matrix</i> oxford17 (a) dan dengan <i>transfer learning</i> (b) .	82
Gambar 4.31 <i>Confusion matrix</i> oxford102.....	82
Gambar 4.32 <i>Confusion matrix</i> oxford102 <i>transfer learning</i>	83
Gambar 4.33 Tampilan klasifikasi.....	84
Gambar 4.34 Tampilan pengaturan	85
Gambar 4.35 <i>Flowchart</i> proses pengujian.....	87
Gambar 4.36 <i>Flowchat</i> program <i>scrapping</i> citra.....	88
Gambar 4.37 Contoh citra gambar untuk pengujian.....	89
Gambar 4.38 Pengujian dengan prediksi benar	90
Gambar 4.39 Pengujian dengan prediksi salah.....	90

DAFTAR TABEL

Tabel 2.1 <i>Confusion Matrix</i>	29
Tabel 2.2 Penelitian sejenis	42
Tabel 4.1 Perbandingan skor klasifikasi CNN	83
Tabel 4.2 Perbandingan skor klasifikasi CNN	84
Tabel 4.3 Hasil pengujian model untuk oxford17	91
Tabel 4.4 Hasil pengujian model untuk oxford17 transfer learning.....	91
Tabel 4.5 Hasil pengujian model untuk oxford102	91
Tabel 4.6 Hasil pengujian model untuk oxford102 <i>transfer learning</i>	92
Tabel 4.7 Hasil perbandingan metode CNN dengan SVM dan ANN.....	94

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Alam ini memiliki berbagai jenis makhluk hidup yang tinggal di dalamnya, yang terdiri atas flora dan fauna atau tumbuhan dan hewan. Terdapat sekitar 374.000 spesies tanaman, di mana sekitar 308.312 adalah tanaman vaskular, dengan 295.383 tanaman berbunga (*angiospermae*; monokotil: 74.273; spesies *eudikotil*: 210.008). Tanaman berbunga memiliki 416 famili, 13.164 genus, dan 295.383 spesies, menjadikan tanaman berbunga sebagai kelompok tanaman di darat dengan ragam terbanyak di dunia (Christenhusz & James, 2016). Bunga merupakan struktur reproduksi yang ditemukan pada tanaman berbunga (yang biasa dikenal *angiospermae* atau *magnoliophyta*). Dalam ilmu botani (ilmu yang mempelajari tentang tumbuh-tumbuhan), bunga merupakan bagian dari tanaman untuk menghasilkan biji, Pembuahan serta penyerbukan berlangsung pada bunga. Setelah dilakukan pembuahan, selanjutnya bunga akan berkembang lebih lanjut membentuk buah. Terdapat ratusan ribu spesies atau jenis bunga di dunia ini, dengan beragam bentuk, warna dan struktur. Pengklasifikasian bunga secara tradisional dilakukan oleh botanis. Masalah yang dihadapi dalam mengklasifikasi bunga adalah karena bunga memiliki warna yang beragam serta terkadang terlihat mirip dengan warna latar belakang objek bunga, dan untuk mengidentifikasi objek memiliki latar belakang warna juga sulit untuk dilakukan (Saitoh, Aoki, & Kaneko, 2004).

Dalam beberapa dekade terakhir *machine learning* banyak digunakan untuk berbagai tujuan dan kebutuhan, seperti memprediksi hasil pertandingan olahraga sepak bola (Bunker & Thabtah, 2019), memprediksi keadaan polusi udara (Xiao, Fang, Zheng, Pain, & Navon, 2019), memprediksi keadaan ombak (James, Zhang, & O'Donncha, 2018), memprediksi faktor *biokonsentrasi* pada ikan dan hewan invertebrata (Miller et al., 2019) hingga membuat model dari *machine learning* yang dapat mengalahkan para pemain profesional dari permainan kompleks “Go” (Silver et al., 2016). Oleh karena dari itu, penggunaan *machine learning* untuk proses pengklasifikasian bunga diharapkan dapat memangkas waktu serta memiliki keakuratan yang tinggi dalam hal akurasi klasifikasi.

Untuk dapat mengklasifikasi jenis bunga dari semua input (citra bunga) dengan *machine learning*, maka dibutuhkan teknik *supervised learning*. Dalam *supervised learning*, label (jenis bunga) diberikan kepada algoritma sebagai dasar kebenaran (Rozenblit et al., 2018). Teknik *supervised learning* dengan metode klasifikasi juga dapat ditemukan di berbagai aplikasi seperti pendeteksi web penipuan, penilaian kartu kredit, klasifikasi email dan banyak aplikasi lainnya (Abdelhamid & Thabtah, 2014). Dengan berkembangnya *machine learning* maka lahirlah *deep learning*. *Deep learning* merupakan sub bidang dari *machine learning* yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak manusia yang disebut jaringan saraf (*neuron*) tiruan (Brownlee, 2016). Memanfaatkan jaringan saraf (*neural network*) secara hierarkis dan berskala besar dengan koneksi yang kuat, *deep learning* dapat menghasilkan prediksi maupun klasifikasi dengan akurasi yang tinggi, yang mampu melampaui model *machine*

learning tradisional dalam beberapa bidang seperti pengenalan gambar dan suara (LeCun, Bengio, & Hinton, 2015).

Terdapat beberapa penelitian untuk mengklasifikasikan bunga dengan menggunakan *machine learning* (Albadarneh & Ahmad, 2017; Almogdady, Manaseer, & Hiary, 2018; Tiay, Benyaphaichit, & Riyamongkol, 2014). Penelitian-penelitian tersebut menggunakan berbagai algoritma *machine learning*, seperti menggunakan algoritma *K-nearest neighbor* (Tiay et al., 2014), algoritma *artificial neural networks* (Almogdady et al., 2018), algoritma *support-vector machines* (Albadarneh & Ahmad, 2017). Hasil dari penelitian-penelitian tersebut rata-rata memperoleh hasil akurasi dalam pengklasifikasian jenis bunga mencapai 81.57%. Namun, belum ada penelitian untuk mengklasifikasikan bunga yang berfokus pada *deep learning* dengan *convolutional neural network*. Dalam penelitian ini teknik *convolutional neural network* digunakan untuk mengklasifikasi bunga berdasarkan jenis yang telah diberi label sebelumnya. *Convolutional Neural Networks* (CNN) telah merevolusi bidang ilmu *computer vision* dan *pattern recognition*, dengan memiliki tingkat akurasi pengenalan pola yang lebih tinggi (Ptucha et al., 2019). Karena jika dibandingkan dengan *machine learning*, *convolutional neural network* pada *deep learning* memiliki tiga kelebihan; Pertama, mendukung pembelajaran berdasarkan representasi selama proses pelatihan, sehingga dapat secara otomatis beradaptasi dengan data dan tugas prediksi pada bidang tertentu. Kedua, dengan memiliki peningkatan kekuatan dalam proses komputasinya, CNN dapat memfasilitasi pembuatan *neural network* yang lebih dalam serta menunjukkan kemampuan dalam merepresentasikan model dengan kinerja yang lebih kuat dan

prediksi yang lebih akurat. Ketiga, *deep learning* mendukung pelatihan *end-to-end* dan tidak mengharuskan pengguna memiliki pengetahuan dibidang data atau model yang dianalisisnya (Yu & Shi, 2018).

Peneliti menggunakan *dataset* yang diperoleh dari universitas Oxford, yaitu oxford17 (Nilsback & Zisserman, 2006) dan oxford102 (Nilsback & Zisserman, 2008) serta data yang berasal dari hasil *scrapping* internet dengan google images. Proses pembuatan model dalam penelitian ini menggunakan dua pendekatan, yang pertama penggunaan arsitektur model CNN dari awal dan pendekatan kedua adalah penggunaan metode *transfer learning* dengan ResNet yang memiliki kinerja paling baik (Saikia et al., 2019). Sedangkan untuk proses *training*, algoritma optimasi Nadam digunakan untuk mencapai hasil yang optimal, dikarenakan algoritma ini memiliki kelebihan, yaitu hemat sumber daya komputasi dan memiliki kinerja yang lebih baik dari algoritma optimasi *deep learning* lainnya (Dogo et al., 2018). Penelitian ini ditujukan untuk membandingkan kinerja metode *convolutional neural networks* dengan metode *machine learning* lainnya seperti *Support Vector Machine* (SVM) yang memperoleh hasil akurasi 83.52% (Almogdady et al., 2018) dan *Artificial neural networks* (ANN) dengan hasil akurasi 81.19% (Albadarneh & Ahmad, 2017), kedua penelitian tersebut juga menggunakan *preprocessing* data dengan *image segmentation* dimana pemisahan objek bunga dengan latar belakang dilakukan secara semi otomatis, atau dengan bantuan *software image processing*. Penelitian ini menghasilkan model yang mampu belajar dari hasil input data berupa gambar-gambar bunga dan mampu menghasilkan prediksi jenis bunga. Serta mampu menghasilkan kinerja model yang lebih baik menggunakan *Convolutional*

Neural Networks. Oleh karena itu, Berdasarkan hal-hal tersebut, maka peneliti tertarik untuk melakukan penelitian berjudul **“Implementasi *Deep Learning* Menggunakan *Convolutional Neural Network* untuk Klasifikasi Bunga”**.

1.2 Identifikasi Masalah

Berdasarkan latar belakang terdapat beberapa masalah sebagai berikut:

1. Kinerja akurasi model klasifikasi bunga dalam penelitian sebelumnya (SVM dan ANN) belum mencapai hasil yang sangat baik. Serta belum adanya model dengan metode CNN untuk mengklasifikasi bunga sehingga belum diketahui kinerja akurasi CNN untuk mengklasifikasikan bunga.
2. Belum adanya model serta pengujian terhadap kinerja akurasi dari model CNN terhadap klasifikasi bunga.

1.3 Perumusan Masalah

Berdasarkan identifikasi masalah yang telah disebutkan sebelumnya, perumusan masalah dalam penelitian ini adalah bagaimana pemodelan CNN, proses pengujian dan kinerja *convolutional neural networks* untuk mengklasifikasi citra bunga.

1.4 Batasan Masalah

Adapun batasan masalah yang dikemukakan dalam penelitian ini adalah sebagai berikut:

- a. Data untuk penelitian ini menggunakan dua *dataset*, yaitu oxford17 (terdiri dari 17 jenis bunga) dan oxford102 (terdiri dari 102 jenis bunga).
- b. Bunga yang menjadi data dalam proses penelitian berada pada fase bunga mekar (*Antesis*).

- c. Penelitian ini menggunakan *deep learning* arsitektur CNN (*Convolutional Neural Network*) dan CNN dengan *transfer learning* dengan optimasi Nadam.
- d. Model dari *deep learning* dibuat dengan menggunakan bahasa pemrograman Python versi 3 dengan *library* Numpy, Pandas, Tensorflow, dan Keras.

1.5 Tujuan Penelitian

Penelitian ini memiliki tujuan khusus dan tujuan umum, yaitu:

1.5.1 Tujuan Umum

Adapun tujuan umum dari penelitian ini adalah untuk menghasilkan model CNN, menguji model CNN, dan mengukur kinerja model CNN dalam melakukan klasifikasi bunga..

1.5.2 Tujuan Khusus

- a. Membuat dan merancang model CNN untuk mengklasifikasikan jenis bunga yang terdapat pada dataset Oxford17 dan Oxford102.
- b. Menguji model CNN untuk klasifikasi bunga dan mendapatkan kinerja akurasi dari model CNN berdasarkan pengujian terhadap model dan membandingkannya dengan kinerja akurasi model dengan metode SVM dan ANN.

1.6 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah sebagai berikut:

- a. Memberikan model CNN yang dapat digunakan untuk mengklasifikasikan jenis bunga yang terdapat pada *dataset* Oxford17 dan Oxford102, dan sebagai referensi dalam penelitian selanjutnya.
- b. Menghasilkan perbandingan nilai akurasi model CNN, SVM, dan ANN yang dapat dijadikan nilai acuan dalam menentukan model yang baik untuk digunakan khususnya untuk mengklasifikasikan jenis bunga .

1.7 Metode Penelitian

1.7.1 Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan dalam penelitian ini adalah sebagai berikut:

a. Studi Pustaka

Studi pustaka dilakukan dengan membaca buku-buku maupun jurnal yang terkait dengan pokok permasalahan dalam penelitian ini.

b. Observasi

Observasi merupakan teknik pengumpulan data dengan cara mengamati objek data secara langsung.

c. Studi Literatur

Studi literatur dilakukan dengan membaca buku-buku maupun jurnal penelitian yang berhubungan dengan topik *machine learning*, *deep learning* dan klasifikasi.

1.7.2 Pemodelan dengan *Machine Learning*

Peneliti menggunakan metode yang dikembangkan dari Kozłowski et al. (2019) terdiri atas pengumpulan *dataset*, *pre-processing* data, pemodelan CNN,

training strategy, pengujian, interpretasi, evaluasi, serta menggunakan Python versi 3 dengan *library* Numpy, Pandas, Matplotlib, Seaborn sebagai *tools* dalam melakukan penelitian ini.

1.8 Sistematika Penulisan

Laporan dari penelitian ini terdiri atas 5 (lima) bab. Berikut gambaran umum pokok pembahasan yang akan dibahas tiap-tiap bab tersebut:

BAB 1 PENDAHULUAN

Pada bab ini menjelaskan latar belakang, identifikasi masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB 2 TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori relevan terkait implementasi *deep learning* menggunakan *convolutional neural network* untuk klasifikasi bunga dan penelitian sejenisnya.

BAB 3 METODE PENELITIAN

Bab ini menjelaskan mengenai metode yang digunakan dalam proses penelitian, yang terdiri atas populasi dan sampel, kerangka penelitian, prosedur penelitian.

BAB 4 HASIL DAN PEMBAHASAN

Pada bab ini menjelaskan proses dan hasil yang diperoleh dari implementasi *deep learning* menggunakan *convolutional neural network* untuk klasifikasi bunga.

BAB 5 PENUTUP

Pada bab ini diuraikan kesimpulan dari uraian yang telah dituangkan pada bab sebelumnya serta saran yang dapat digunakan sebagai dasar dalam pengembangan selanjutnya.



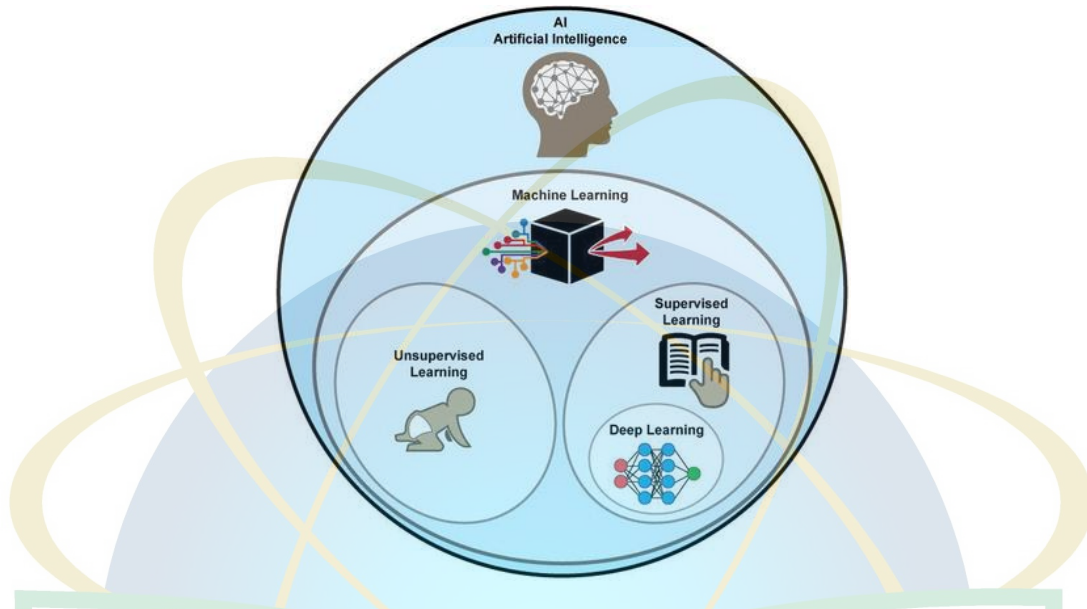
BAB 2

TINJAUAN PUSTAKA

2.1 *Artificial Intelligence*

Inteligensi buatan (*Artificial Intelligence*) adalah sistem simulasi mekanik untuk mengumpulkan pengetahuan dan informasi dan memproses kecerdasan: (menyusun dan menafsirkan) dan menyebarkannya ke dalam bentuk kecerdasan yang dapat ditindaklanjuti (Grewal, 2014). Menurut Zaharchuk et al. (2018) salah satu definisi untuk *Artificial Intelligence* adalah setiap metode komputer yang melakukan tugas-tugas yang biasanya membutuhkan kecerdasan manusia. Pembelajaran mesin (*machine learning*) adalah salah satu jenis kecerdasan buatan yang mengembangkan algoritma untuk memungkinkan komputer belajar dari data yang ada tanpa pemrograman eksplisit. *Deep learning* adalah metode pembelajaran mesin yang diawasi yang menggunakan arsitektur tertentu, yaitu beberapa bentuk jaringan saraf. Kekuatan dari teknik ini adalah dalam skalabilitasnya, yang sebagian besar didasarkan pada kemampuan untuk secara otomatis mengekstraksi fitur yang relevan. Hubungan antara *artificial intelligence*, *machine learning* dan *deep learning* ditunjukkan pada Gambar 2.1 dimana *artificial intelligence* atau kecerdasan buatan merupakan akar dari ilmu *machine learning* dan *deep learning*, dalam membangun kecerdasan buatan dibutuhkan pengetahuan (*knowledge*) pengetahuan ini dibuat dengan pembelajaran (*learning*) dengan mesin *machine learning*. Dalam *machine learning* sendiri terdapat dua teknik utama pembelajaran

supervised dan *unsupervised learning*, dalam *supervised learning* terdapat teknik *deep learning*.



Gambar 2.1 Posisi *deep learning* pada *artificial intelligence* (Zaharchuk et al., 2018)

2.1.1 *Machine Learning*

Machine Learning merupakan sekumpulan teknik yang berfungsi untuk menangani dan memprediksi sekumpulan data dengan cara merepresentasikan data-data tersebut dengan algoritma untuk pembelajaran. Dengan adanya *machine learning*, komputer dapat melakukan pembelajaran secara mandiri dari data-data yang telah diberikan (Danukusumo, 2017). Sedangkan menurut Shukla (2018) *machine learning* atau pembelajaran mesin ditandai dengan perangkat lunak yang belajar dari pengalaman sebelumnya. Program komputer seperti itu dapat meningkatkan kinerjanya karena semakin banyaknya data yang tersedia maka kinerjanya semakin baik. Harapannya adalah jika data yang ada cukup, ia akan mempelajari pola dan menghasilkan kecerdasan buatan untuk data yang baru

dimasukkan. Nama lain dari *machine learning* adalah pembelajaran induktif, karena kode dari mesin mencoba menyimpulkan struktur dari data saja.

a. **Komponen *Machine Learning***

Menurut Vasilev et al. (2019) *machine learning* terdiri atas beberapa komponen berikut ini:

1. Pembelajaran

Pada bagian ini algoritma digunakan sebagai dasar atau filosofi untuk proses pembelajaran mesin. Algoritma dipilih berdasarkan masalah yang ingin diselesaikan atau dipecahkan, karena setiap algoritma hanya cocok untuk menyelesaikan masalah tertentu.

2. Kumpulan data

Kumpulan data atau *dataset* merupakan kumpulan data-data mentah yang menjadi perhatian untuk dipelajari. Kumpulan data ini dapat berlabel maupun tidak berlabel. Memiliki data sampel dengan jumlah yang cukup sangat penting untuk pembelajaran agar dapat memahami struktur masalah.

3. Representasi

Merupakan proses bagaimana merepresentasikan atau menggambarkan data-data berdasarkan pada fitur yang dipilih, sehingga dapat digunakan untuk proses pembelajaran. Misalnya, untuk dapat mengklasifikasikan gambar tulisan tangan, maka gambar akan diwakilkan sebagai nilai *array*. Setiap sel akan berisi nilai dalam warna dalam satu piksel. Pemilihan representasi yang baik akan menentukan untuk mendapatkan hasil yang lebih baik.

4. Tujuan atau sasaran

Bagian ini menggambarkan atau merepresentasikan alasan untuk belajar dari data-data untuk menyelesaikan masalah. Tujuan ini membantu untuk menentukan bagaimana dan apa yang harus digunakan dalam proses pembelajaran dan representasi apa yang digunakan.

5. Target

Target merepresentasikan apa yang dipelajari serta hasil akhir yang didapatkan. Target dapat berupa klasifikasi data yang tidak memiliki label, representasi data yang sesuai dengan pola, memprediksi masa depan, dan respons terhadap stimulus atau strategi.

b. Tipe *Machine Learning*

Menurut Shukla (2018) *machine learning* terbagi menjadi 3 tipe berdasarkan cara pembelajarannya:

1. *Supervised learning*

Supervised learning secara keseluruhan adalah tentang proses pembelajaran dari contoh-contoh data yang diberikan label sebelumnya. *Supervised learning* membutuhkan data berlabel untuk dapat melakukan pelatihan data, yang disebut modelnya. Sebagai contoh, memberikan banyak data berupa foto dan rekaman yang sesuai, kita dapat melatih model untuk mengklasifikasikan etnis dari individu yang ada dalam foto.

2. *Unsupervised learning*

Unsupervised learning adalah tentang memodelkan data yang diinput tanpa label. Dengan data yang cukup, dimungkinkan untuk menemukan pola dan struktur dari data. Dua alat paling yang banyak digunakan praktisi *machine*

learning untuk belajar dari data saja adalah pengelompokan (*clustering*) dan pengurangan dimensi.

3. *Reinforcement learning*

Reinforcement learning melatih informasi yang dikumpulkan dengan mengamati bagaimana lingkungan bereaksi terhadap tindakan. *Reinforcement learning* adalah jenis dari *machine learning* yang berinteraksi dengan lingkungan untuk belajar kombinasi tindakan yang paling menghasilkan hasil yang menguntungkan.

2.1.2 *Deep Learning*

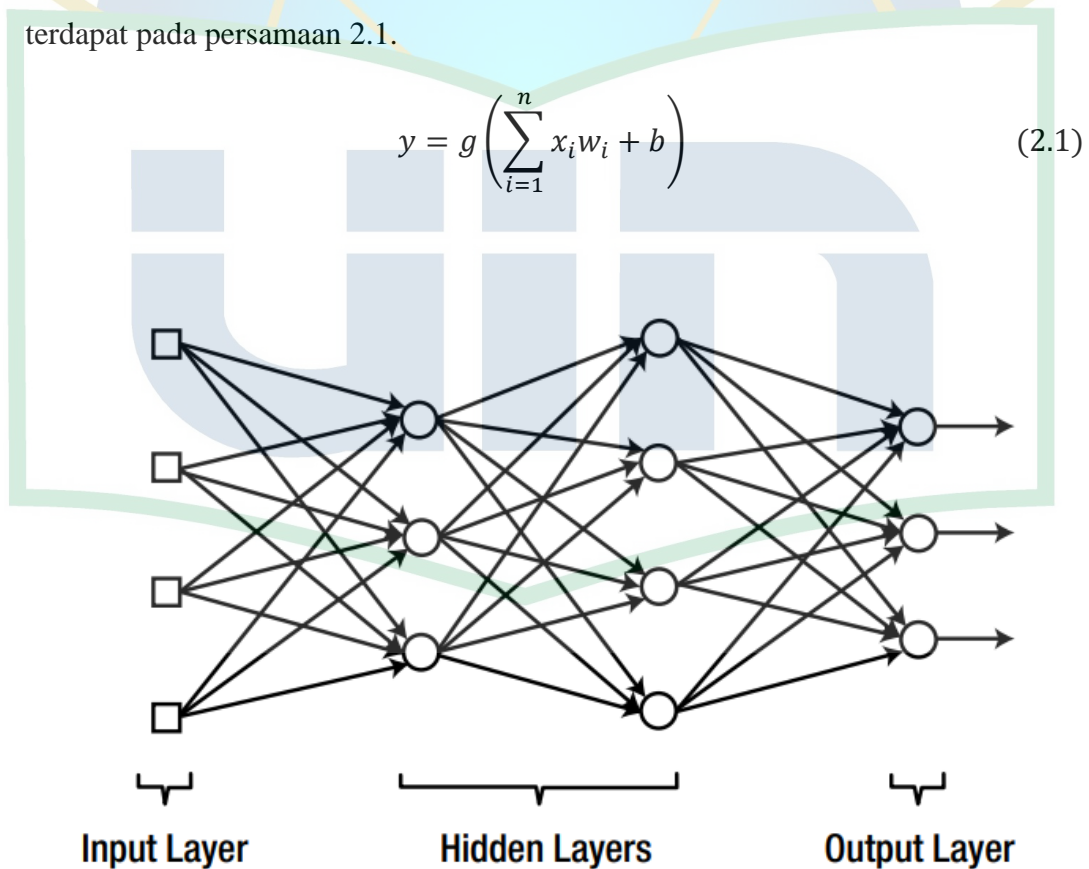
Deep learning memungkinkan model komputasi yang terdiri atas beberapa lapisan pemrosesan untuk mempelajari representasi data dengan berbagai tingkat abstraksi. *Deep learning* menemukan struktur rumit dalam kumpulan data besar dengan menggunakan algoritma *backpropagation* untuk menunjukkan bagaimana mesin harus mengubah parameter internal yang digunakan untuk menghitung representasi di setiap lapisan dari representasi di lapisan sebelumnya (LeCun et al., 2015). *Deep learning* terdiri atas:

a. *Neural Network*

Istilah *neural networks* pertama kali digunakan oleh McCulloch & Pitts (1990) dalam percobaan untuk menemukan representasi matematis dari pemrosesan informasi dalam sistem biologis. Jaringan saraf (*neural networks*) merupakan jaringan dari *node* (simpul), yang meniru struktur neuron otak dari makhluk hidup. *Node* menghitung jumlah nilai bobot dari masukan dan memprosesnya pada lapisan tersembunyi, lalu mengeluarkan hasil dari fungsi pengaktifan dengan nilai bobot.

Neural networks telah dikembangkan dari arsitektur sederhana menjadi struktur yang semakin kompleks. Awalnya, pelopor *neural networks* memiliki arsitektur yang sangat sederhana dengan hanya lapisan input dan *output*, yang disebut jaringan saraf *single-layer*. Ketika lapisan tersembunyi (*hidden layer*) ditambahkan ke jaringan saraf *single-layer*, maka akan menghasilkan jaringan saraf *multi-layer*. Oleh karena itu, jaringan saraf *multi-layer* terdiri atas lapisan input, lapisan tersembunyi, dan lapisan *output* seperti pada Gambar 2.2 (Kim, 2017).

Untuk mendapatkan neuron tujuan (y) maka nilai yang ada pada neuron (x) dikalkulasi dengan bobot (w) dan ditambahkan dengan bias (b) lalu diaktivasi dengan fungsi (g), yang akan menentukan neuron selanjutnya (y). Rumusnya terdapat pada persamaan 2.1.



Gambar 2.2 Struktur dari *neural networks* (Kim, 2017)

b. Fungsi Aktivasi

Fungsi aktivasi adalah sebuah fungsi yang berguna untuk menentukan aktif tidaknya *neuron* di dalam *neural networks*.

1. *Softmax*

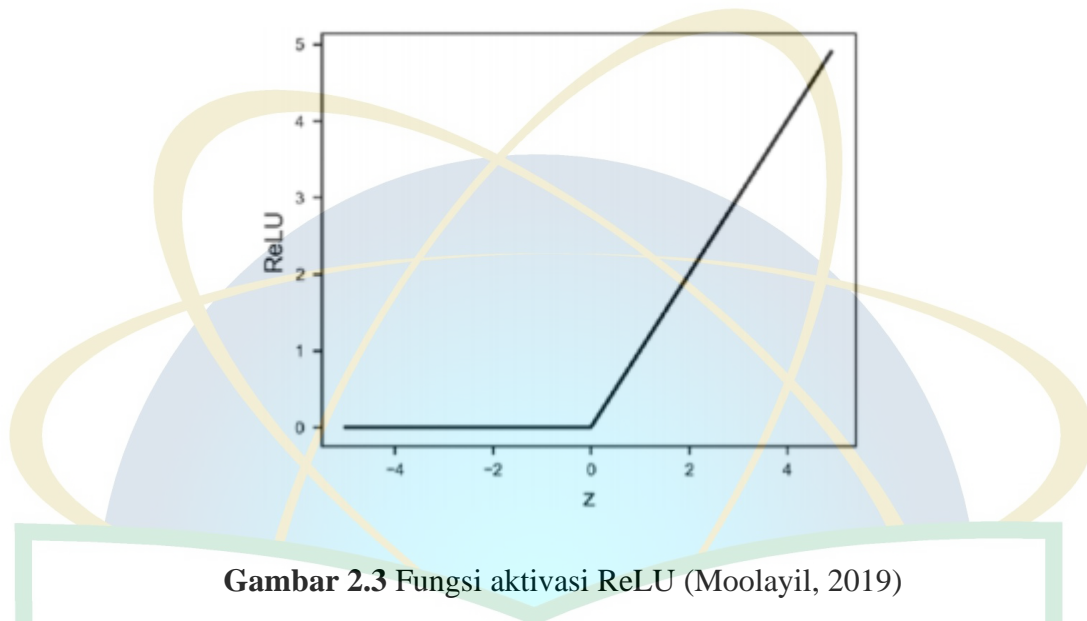
Fungsi *Softmax* adalah fungsi eksponensial yang dinormalisasi untuk mengubah vektor asli D-dimensi dengan nilai riil yang berubah-ubah menjadi vektor probabilitas D-dimensi dengan nilai riil dalam kisaran [0,1]. Fungsi *Softmax* biasanya diterapkan ke bidang pembelajaran mesin, seperti regresi logistik, jaringan saraf tiruan, pembelajaran penguatan. Fungsi *Softmax* dapat digunakan untuk menghitung nilai dari probabilitas untuk semua label. Rumus dari softmax dapat dilihat pada persamaan (2.2) dimana nilai probabilitas (S) pada kelas ke (y) diambil dari neuron pada layer klasifikasi terakhir yang berupa angka eksponensial (e) yang dibagi jumlah nilai eksponensial itu sendiri. Hasil dari label yang ada mengubahnya, akan diambil sebuah vektor nilai yang memiliki nilai riil dan mengubahnya menjadi vektor dengan nilai dengan kisaran angka nol dan satu. Jika semua hasil dijumlah maka akan bernilai satu (He et al., 2018).

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad (2.2)$$

2. *Rectified Linear Unit* (ReLU)

ReLU menggunakan fungsi $f(z) = \max(0, z)$, yang artinya jika *output* positif maka akan menghasilkan nilai yang sama, jika tidak maka akan menghasilkan nilai 0. ReLU tidak hanya meningkatkan kinerja secara signifikan tetapi juga membantu mengurangi jumlah perhitungan selama fase pelatihan. Hal ini

terjadi akibat dari nilai 0 dalam *output* ketika nilai z negatif, sehingga menonaktifkan neuron (Moolayil, 2019). Hasil dari fungsi ditampilkan dalam Gambar 2.3.



c. *DropOut*

DropOut adalah salah satu teknik yang digunakan untuk menghindari terjadinya *overfitting* dalam model. Dalam metode ini, aktivasi beberapa neuron yang dipilih secara acak dalam jaringan diambil sebagai nol selama pelatihan. *Neuron* yang dipilih diubah dalam setiap iterasi pelatihan. Proses pembelajaran menjadi lebih andal dan *overfitting* dikurangi dengan metode ini.

Istilah “*DropOut*” mengacu pada pemutusan *neuron* (tersembunyi dan terlihat) dalam *neural network*. Dengan mengeluarkan unit (*neural*) untuk sementara menghapusnya dari jaringan (*network*), bersama dengan semua koneksi masuk dan keluarnya, seperti yang ditunjukkan pada Gambar 2.4. Pemilihan unit yang dijatuhkan secara acak (Srivastava et al., 2014).



Gambar 2.4 Sebelah kiri adalah *neural networks* biasa, sebelah kanan setelah melakukan Dropout (Srivastava et al., 2014)

4. *Loss Function*

Goodfellow et al. (2016) menjelaskan *loss function* atau *cost function* adalah metode untuk mengevaluasi seberapa baik algoritma dalam memodelkan data yang diberikan. Jika hasil prediksi menyimpang terlalu banyak dari hasil aktual, *loss function* akan memiliki nilai dalam jumlah yang sangat besar. Secara bertahap, dengan bantuan beberapa fungsi pengoptimalan, *loss function* belajar untuk mengurangi kesalahan dalam prediksi.

5. *Backpropagation*

Menurut Cilimkovic (2015) *backpropagation* merupakan algoritma untuk mencari nilai minimum dari *loss function* dalam bobot (*weight*) menggunakan teknik yang disebut aturan delta atau *gradient descent*. Bobot yang meminimalkan *loss function* kemudian dianggap sebagai solusi untuk masalah pembelajaran.

Algoritma dapat dibagi kedalam empat langkah berikut:

- a. Perhitungan *feed-forward*
- b. *Backpropagation* ke lapisan *output*

- c. *Backpropagation* ke lapisan tersembunyi (*hidden*)
- d. Pembaruan bobot (*weight*)

Algoritma dihentikan ketika nilai fungsi kesalahan menjadi cukup kecil

2.1.3 Klasifikasi

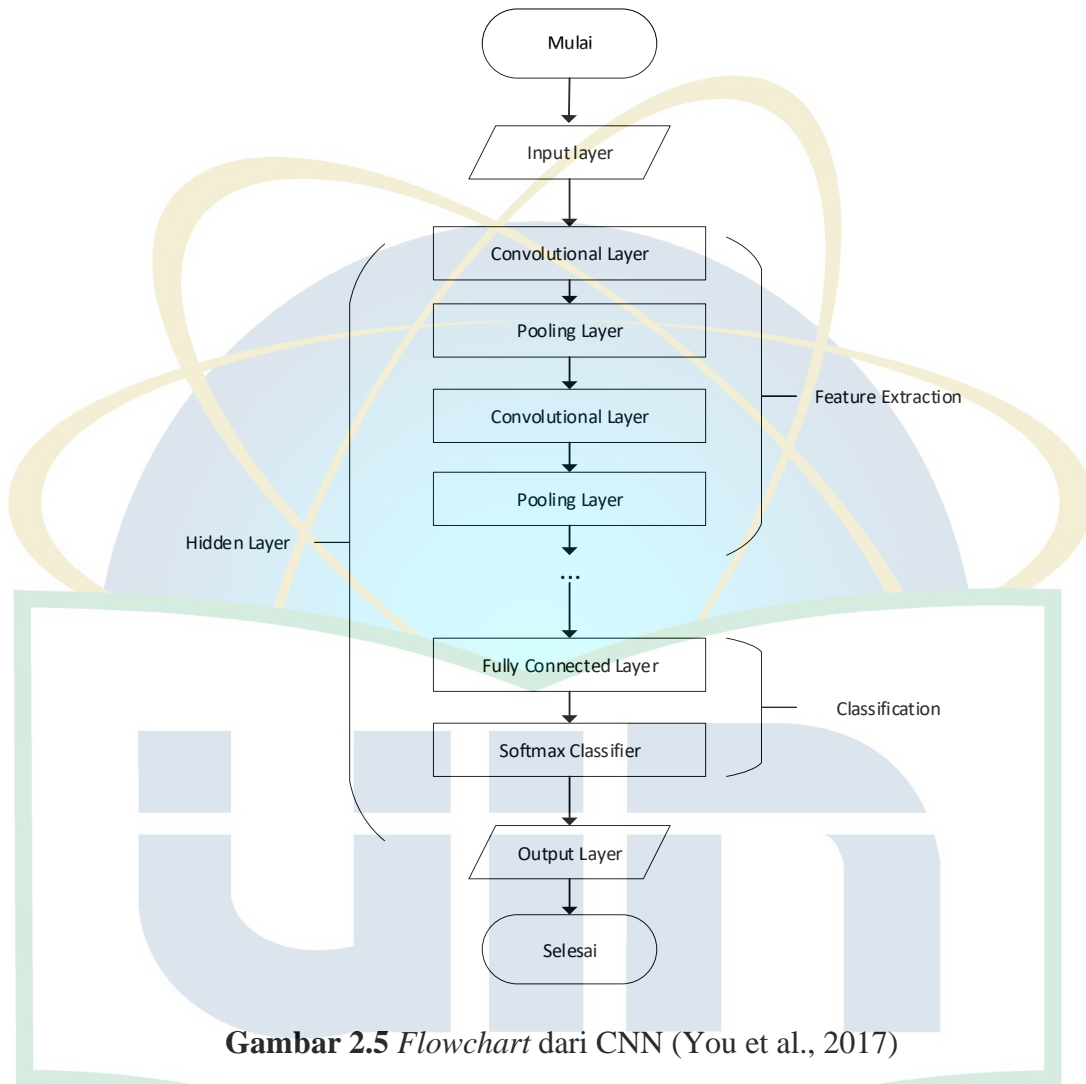
Klasifikasi adalah suatu bentuk dari analisis data yang mengekstraksi model untuk menggambarkan mengategorikan atau kelas dari data. Dalam klasifikasi, pengklasifikasian atau model yang dibangun untuk memprediksi label kelas (kategorial), misalnya sebuah cuaca hujan atau terik. Kategori-kategori ini dapat diwakilkan oleh nilai diskrit, pengurutan antar nilai tidak mempunyai arti. Klasifikasi sendiri terdiri atas dua langkah atau dua proses, proses yang pertama adalah proses pembelajaran (proses pengklasifikasian dibangun), sedangkan proses kedua adalah proses klasifikasi (model yang dibangun digunakan untuk memprediksi label dari data yang telah diberikan) (Han et al., 2011).

2.2 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah sebuah arsitektur dari *deep learning*. CNN mencakup banyak lapisan representasi. Karena struktur yang dalam ini, CNN dapat secara otomatis mendapatkan karakteristik representasi dari data melalui transformasi nonlinier dan perkiraan fungsi nonlinier.

Struktur CNN terdiri atas ekstraksi fitur yang terdiri atas *convolutional layer* yang biasanya diikuti oleh *pooling layer* dan pengklasifikasi softmax. Pada lapisan konvolusional mengekstraksi fitur dari citra gambar, sedangkan pada *pooling layer* mengurangi dimensi dan mengurangi waktu komputasi. Arsitektur ini dapat mencapai bentuk regularisasi dengan sendirinya. Fitur yang diekstraksi kemudian

dimasukkan ke dalam lapisan *softmax* atas untuk proses klasifikasi (You et al., 2017).



Gambar 2.5 Flowchart dari CNN (You et al., 2017)

Proses pada CNN ditunjukkan pada Gambar 2.5 dimulai dari input, pada proses ini data berupa citra gambar dimasukkan, data yang diambil dari tiap *pixel* citra panjang×lebar×1 untuk citra hitam putih (*grayscale*) dan panjang×lebar×3 untuk citra dengan warna (RGB). Tahap selanjutnya adalah ekstraksi fitur dari citra, pada bagian ini dilakukan “*encoding*” dari sebuah citra gambar menjadi *features* yang berupa angka-angka yang merepresentasikan citra gambar tersebut. *Feature*

extraction memiliki dua bagian utama yaitu *convolutional layer* dan *pooling layer*. Pada proses *Feature extraction* jumlah *convolutional layer* dan *pooling layer* dapat disesuaikan dengan kebutuhan, semakin banyak jumlahnya maka semakin dalam arsitektur sehingga meningkatkan akurasi untuk klasifikasi.

Proses *feature extraction* menghasilkan *feature map* yang berbentuk multidimensional array, sehingga harus melalui proses “flatten” atau *reshape* *feature map* menjadi sebuah vektor sebagai input dari *fully connected layer*. Proses *classification* memiliki beberapa *hidden layer*, *activation function*, dan *loss function*. Proses pada *softmax classifier* mengubah angka alias log menjadi probabilitas yang berjumlah satu, *softmax classifier* menghasilkan vektor yang mewakili probabilitas dari daftar hasil (label) yang potensial.

2.2.1 Lapisan *Convolutional Neural Network*

Convolutional Neural Network (CNN) terdiri atas tiga lapis (*layer*) yaitu lapis masukan (*input layer*), lapis keluaran (*output layer*), dan beberapa lapis tersembunyi (*hidden layers*). Lapis tersembunyi (*hidden layer*) umumnya berisi *convolutional layers*, *pooling layers*, *normalization layers*, *ReLU layer*, *fully connected layers*, dan serta *loss layer* (Alom et al., 2018).

a. *Convolutional Layer*

Lapisan konvolusional (*convolutional layer*) merupakan lapisan inti CNN, pada lapisan ini sebagian besar proses komputasi dilakukan. Tujuan utama konvolusi dalam kaitannya dengan *ConvNet* adalah untuk mengekstraksi fitur dari gambar yang dimasukkan (Karim, 2018). Lapisan konvolusi terdiri atas struktur dengan sejumlah filter dengan ukuran tetap yang memungkinkan fungsi kompleks

diterapkan pada gambar yang telah dimasukkan. Proses ini dilakukan dengan cara menggeser filter di atas gambar. Setiap filter memiliki bobot dan nilai bias yang sama di seluruh gambar selama proses ini. Proses ini disebut mekanisme pembagian nilai berat dan mekanisme ini memberikan kemampuan untuk mewakili fitur yang sama pada seluruh gambar (Sarigül, Ozyildirim, & Avci, 2019). Lapisan konvolusi menghasilkan gambar baru yang disebut peta fitur. Peta fitur menonjolkan fitur unik dari gambar asli. Lapisan konvolusi beroperasi dengan cara yang sangat berbeda dibandingkan dengan lapisan jaringan saraf lainnya. Pada Gambar 2.6 menunjukkan proses dari lapisan konvolusi, di mana tanda * menunjukkan operasi konvolusi, dan tanda ϕ adalah fungsi aktivasi. Ikon dengan skala abu-abu (*greyscale*) di antara operator ini menunjukkan filter konvolusi. Lapisan konvolusi menghasilkan jumlah peta fitur yang sama dengan filter konvolusi. Karena itu, misalnya, jika lapisan konvolusi berisi empat filter, itu akan menghasilkan empat peta fitur (Kim, 2017).

Terdapat dua komponen penting lainnya dalam *convolutional layer*:

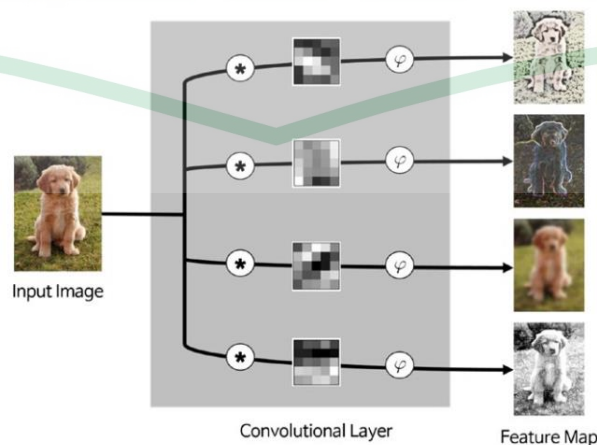
1. *Strides*

Stride adalah jumlah yang digunakan oleh *filter* untuk menggeser gambar. Jika *stride* bernilai 1 maka *filter* akan berpindah 1 piksel secara horizontal dan vertikal. Hingga konvolusi menjadi sama dengan lebar dan kedalaman dari gambar yang dimasukkan. *Stride* bernilai 2 membuat lapisan *convolutional* dengan setengah lebar dan tinggi gambar. Jika *filter* meluas di luar dari ukuran gambar, maka dapat mengabaikan nilai yang tidak diketahui ini atau menggantinya dengan nilai nol (Karim, 2018).

2. *Padding*

Menurut Karim (2018) *padding* merupakan operasi untuk menambah ukuran dari data yang input. Dalam data satu dimensi, cukup menambahkan *array* dengan konstanta; dalam data dua dimensi, data dapat ditambahkan dengan mengelilingi matriks dengan konstanta. Dalam n -dimensional, data dikelilingi *hypercube* n -dimensional dengan konstanta. Penggunaan *padding* bertujuan untuk memanipulasi dimensi *output* dari peta fitur. Penggunaan *padding* dapat untuk mengatur dimensi *output* agar tetap sama seperti dimensi input atau setidaknya tidak berkurang drastis sehingga dapat dilakukan ekstraksi *feature* yang lebih mendalam. Dalam sebagian besar kasus, nilai konstanta ini adalah nol dan disebut *zero padding*:

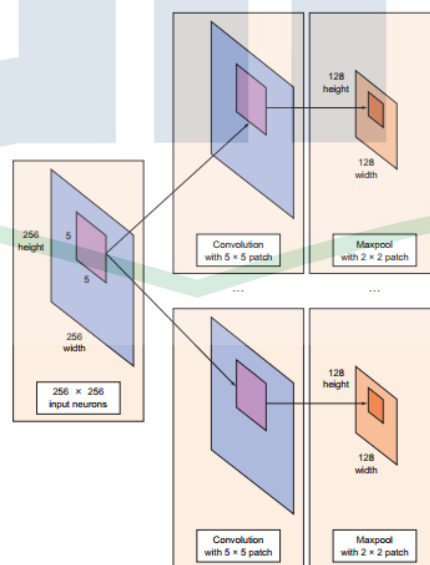
- *Valid padding*: Hanya menjatuhkan nilai dari kolom paling kanan (atau baris paling bawah)
- *Same Padding*: Mencoba untuk meratakan kiri dan kanan data, tetapi jika jumlah kolom yang akan ditambahkan adalah ganjil, maka akan menambahkan kolom tambahan ke kanan.



Gambar 2.6 Proses pada lapisan konvolusi (Kim, 2017)

b. Pooling Layer

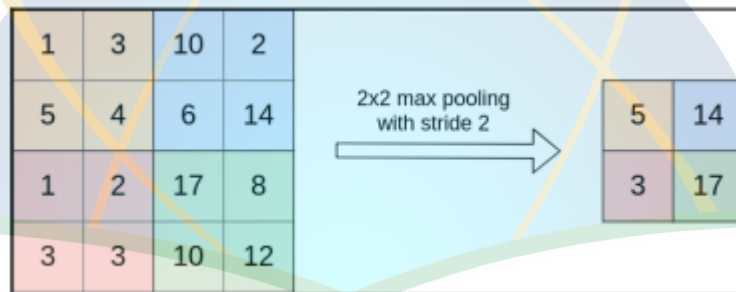
Pooling layer berfungsi menjaga ukuran data ketika *convolution* dilakukan, yaitu dengan cara melakukan *downsampling* (produksi sampel), dengan adanya *pooling layer* data dapat direpresentasikan menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overfitting*. *Pooling layer* mengambil layer *convolutional* sebagai input. Proses pada *pooling layer* diterapkan ke *feature maps* yang telah melewati fungsi konvolusi dan aktivasi. Pada proses ini menghasilkan *feature map* yang lebih kecil, yang merupakan ringkasan dari *feature map* yang dimasukkan. *Pooling* dilakukan dengan cara menggeser filter pada gambar untuk menerapkan operasi yang dipilih, seperti yang ditunjukkan pada Gambar 2.7. Operasi *pooling* yang biasa digunakan adalah *max pooling*, *average pooling* dan *L2-norm pooling* (Sarigül et al., 2019). Keuntungan terbesar yang diberikan oleh operasi *pooling* adalah pengurangan ukuran gambar dan ekstraksi fitur visual secara independen pada gambar (Nielsen, 2015).



Gambar 2.7 Proses pada lapisan konvolusi dan lapisan *pooling* (Shukla, 2018)

1. *Max Pooling*

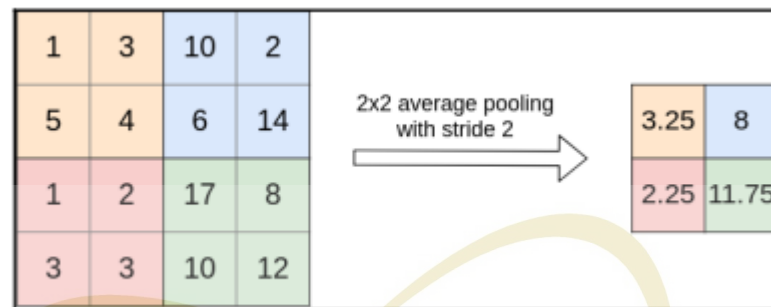
Max pooling merupakan metode *pooling* yang paling populer. Operasi ini dilakukan dengan cara mengambil neuron dengan nilai aktivasi tertinggi di setiap bidang reseptif lokal (*grid cell*), dan hanya mengambil nilai tertinggi dan mengirim nilai itu ke proses selanjutnya. Pada Gambar 2.8 merupakan contoh *max pooling*, dimana bidang yang berjumlah 4×4 , diperkecil menjadi bidang reseptif 2×2 dengan cara mengambil nilai yang terbesar (Vasilev et al., 2019).



Gambar 2.8 Proses *max pooling* (Vasilev et al., 2019)

2. *Average Pooling*

Average Pooling adalah metode *pooling* lainnya, di mana *output* dari masing-masing bidang reseptif adalah nilai rata-rata dari semua aktivasi dalam bidang tersebut. Pada Gambar 2.9 merupakan contoh *average pooling* dengan cara mengambil nilai rata-rata untuk memperkecil ukuran (Vasilev et al., 2019).



Gambar 2.9 Proses *average pooling* (Vasilev et al., 2019)

c. *Normalization Layer*

Lapisan normalisasi (*Normalization Layer*) dibuat untuk mengatasi adanya perbedaan rentang nilai yang signifikan pada citra yang dimasukkan. *Normalization layer* tidak banyak digunakan secara praktis karena dampaknya yang relatif kecil (Suyanto, 2018).

d. *ReLU (Rectified Linear Unit) Layer*

Rectified Linear Units (ReLU) layer berfungsi untuk meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang reseptif pada *convolution layer* (Suyanto, 2018).

e. *Fully Connected Layer*

Fully connected layer merupakan lapisan setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Setelah proses konvolusi dan *pooling*, data ditransformasikan menjadi vektor satu dimensi. Vektor ini menjadi input dari jaringan yang sepenuhnya terhubung. Struktur yang terhubung sepenuhnya dapat berisi satu atau lebih lapisan tersembunyi. Setiap neuron mengalikan bobot koneksi dengan data dari lapisan sebelumnya dan menambahkan

nilai bias. Nilai yang dihitung melewati fungsi aktivasi sebelum dikirim ke lapisan berikutnya (Sarigül et al., 2019).

f. Loss Layer

Loss layer merupakan lapisan yang menentukan bagaimana proses pelatihan memberikan penalti atas penyimpangan antara hasil prediksi dan label (Suyanto, 2018).

2.2.2 Kelebihan CNN

Menurut Yu & Shi (2018) *deep learning* dengan metode CNN memiliki beberapa kelebihan diantaranya:

1. Proses CNN yang hierarkis dan menyederhanakan fitur ini mendukung pembelajaran representasi selama pelatihan, sehingga dapat secara otomatis beradaptasi dengan data dan tugas prediksi di bidang tertentu.
2. Dengan peningkatan dalam kekuatan komputasinya, ia memfasilitasi pembangunan jaringan saraf yang lebih dalam dan menunjukkan kemampuan representasi model yang lebih kuat serta kinerja prediksi yang lebih baik.
3. Metode pembelajaran pada CNN mendukung pelatihan *end-to-end* dan tidak mengharuskan pengguna memiliki pengetahuan domain pada objek data yang digunakan.

2.2.3 Model CNN Kozłowski

Model *convolutional neural network* yang dibangun oleh Kozłowski et al. (2019) digunakan dalam penelitian klasifikasi berbagai jenis pada tanaman gandum. Model

CNN terdiri dari lapisan-lapisan atau *layer* sebagai berikut ini: (1) *Input layer* dengan ukuran sebesar 80×170 piksel dan dengan 3 saluran atau *channel* warna RGB (*Red, Green, Blue*). (2) Lapisan *convolutional* dengan jumlah 32, 64 atau 128 filter dengan ukuran 3×3 , 5×5 atau 9×9 . Dengan menggunakan lapisan fungsi aktivasi RELU dan diikuti *pooling layer* dengan ukuran 3×3 dan langkah (*stride*). (3) Lapisan *convolutional* dengan 128 filter ukuran 3×3 dengan fungsi aktivasi RELU dan diikuti *pooling layer* dengan ukuran 3×3 dan *stride* 2. (4) yang dihubungkan *fully connected layer* dengan 1024 *output* dan diakhiri *fully connected layer* dengan 6 output.

2.2.4 Model CNN Steinbrener

Pada penelitiannya Steinbrener et al. (2019) menggunakan dataset citra dengan ukuran 256×256 piksel, lalu *resize* menjadi 224×224 piksel yang kemudian dijadikan sebagai data input. Penelitian ini menggunakan model *machine learning* GoogleNet sebagai dasar pada transfer learning, pada GoogleNet terdiri dari terdiri dari 9 modul *inception* serta beberapa lapisan konvolusional sederhana dan beberapa lapisan yang sepenuhnya terhubung (*fully connected layer*). Pada proses pelatihan menggunakan *learning rate* sebesar 0.0001 dengan batch sebesar 32.

2.3 Augmentasi Data

Augmentasi data bertujuan untuk meningkatkan jumlah sampel, secara artifisial. Proses ini dilakukan untuk mencegah situasi *overfitting* dalam proses pelatihan. Mengubah ukuran gambar, orientasinya, kondisi pencahayaan adalah

contoh operasi untuk augmentasi data. Data buatan yang dihasilkan ini juga termasuk dalam *dataset* pelatihan (Sarigül et al., 2019).

2.4 Confusion Matrix

Confusion matrix adalah salah satu metode pengukuran keputusan yang paling banyak digunakan dalam *supervised machine learning*. *Confusion matrix* memvisualisasikan nilai tingkat kebingungan dari algoritma pada setiap kelas yang berbeda dan tidak tergantung pada algoritma klasifikasi. Tujuan dari *confusion matrix* adalah untuk melakukan perhitungan akurasi pada konsep data mining. Evaluasi dengan *confusion matrix* menghasilkan nilai akurasi, presisi dan *recall*. Nilai akurasi adalah persentase ketepatan *record* data yang diklasifikasikan secara benar setelah dilakukan pengujian pada hasil klasifikasi. Presisi atau *confidence* merupakan proporsi kasus yang diprediksi positif yang juga hasilnya positif benar pada data yang sebenarnya. *Recall* atau *sensitivity* adalah proporsi kasus positif yang sebenarnya yang diprediksi positif secara benar (Mayadewi & Rosely, 2015).

Tabel 2.1 Confusion Matrix

Aktual	Prediksi	
	+	-
+	<i>True positives (A)</i>	<i>False negatives (B)</i>
-	<i>False positives (C)</i>	<i>True negatives (D)</i>

Untuk dapat menghitung akurasi pada tabel *confusion matrix* dapat menggunakan rumus sebagai berikut:

$$\text{Akurasi} = \frac{(A + D)}{(A + B + C + D)} \quad (2.3)$$

Presisi (*Precision*) merupakan rasio item relevan yang dipilih terhadap semua item yang terpilih. Sehingga presisi dapat diartikan sebagai kecocokan antara permintaan

informasi dengan jawaban terhadap permintaan tersebut. Untuk dapat menghitung presisi dapat digunakan rumus sebagai berikut:

$$Precision = \frac{A}{(C + A)} \quad (2.4)$$

Recall merupakan rasio dari item relevan yang dipilih terhadap total jumlah item relevan yang tersedia. *Recall* dapat dihitung dengan rumus sebagai berikut:

$$Recall = \frac{A}{(A + D)} \quad (2.5)$$

Presisi dan *Recall* dapat diberi nilai dengan menggunakan perhitungan persentase (1-100%) atau dengan menggunakan bilangan antara 0-1.

F1 Score merupakan perbandingan rata-rata presisi dan recall (Powers & Ailab, 2011).

$$F_1 \text{ Score} = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.6)$$

Akurasi memiliki tingkat nilai diagnosa yaitu:

- a. Akurasi dengan nilai 0.90-1.00 dapat dikategorikan sebagai *excellent classification*
- b. Akurasi dengan nilai 0.80-0.90 dapat dikategorikan sebagai *good classification*
- c. Akurasi dengan nilai 0.70-0.80 dapat dikategorikan sebagai *fair classification*
- d. Akurasi dengan nilai 0.60-0.70 dapat dikategorikan sebagai *poor classification*
- e. Akurasi dengan nilai 0.50-0.0 dapat dikategorikan sebagai *failure*

2.5 Nadam Optimization

Nadam merupakan algoritma optimasi yang dikembangkan dari Adam Kingma & Ba (2014). Adam merupakan sebuah metode yang diciptakan untuk optimasi stokastik yang efisien dan hanya membutuhkan gradien urutan pertama dengan kebutuhan memori yang sedikit. Metode ini menghitung laju pembelajaran adaptif individu untuk parameter yang berbeda dari perkiraan momen pertama dan kedua dari gradien; nama Adam berasal dari estimasi momentum adaptif. Metode ini mudah diterapkan, efisien secara komputasi, memiliki sedikit persyaratan memori, tidak berubah-ubah untuk skala gradien secara diagonal, dan sangat cocok untuk masalah yang besar dalam hal data dan / atau parameter.

Berbeda dengan Adam yang momentum reguler, Nadam menggunakan *Nesterov's accelerated gradient* (NAG). Nadam memodifikasi komponen momentum Adam untuk memanfaatkan kelebihan dari NAG, dan kemudian menghasilkan substitusi yang meningkatkan kecepatan konvergensi dan kualitas dari model (Dozat, 2016). Nadam memiliki kinerja lebih baik dibandingkan algoritma sejenis lainnya seperti yang ditunjukkan pada Gambar 2.10.

Nadam merupakan bagian dari stochastic gradient descent, dalam deep learning algoritma ini berfungsi untuk memperbarui nilai bobot dan bias supaya loss yang dihasilkan semakin kecil. Algoritma Nadam digunakan dalam proses *backpropagation* untuk mengupdate bobot (*weight*) dan bias pada parameter selanjutnya w_{t+1} dimana t adalah *timestep* atau posisi bobot dan w_t adalah bobot saat ini. Nilai L merupakan nilai dari *loss function*, sedangkan nilai parameter *learning rate* α dan *decay* β dapat diatur sesuai kebutuhan. Rata-rata gerak eksponensial

gradien V dan rata-rata gerak eksponensial dari gradien kuadrat S diinisiasi dengan nilai 0.

$$w_{t+1} = w_t - \frac{a}{\sqrt{\hat{S}_t + \epsilon}} \left(\beta_1 \hat{V}_t + \frac{1 - \beta_1}{1 - \beta_1^t} \cdot \frac{\delta L}{\delta w_t} \right) \quad (2.6)$$

Dengan \hat{V}_t dan \hat{S}_t didapat dari:

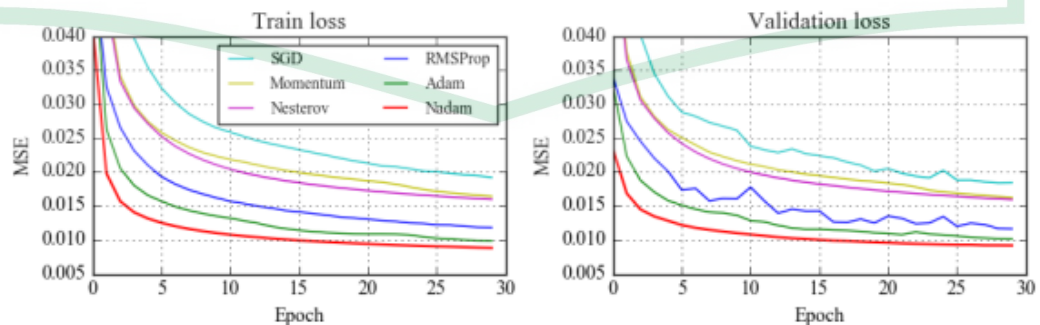
$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t} \quad (2.7)$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t} \quad (2.8)$$

Dan V_t dan S_t didapat dari:

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\delta L}{\delta w_t} \quad (2.9)$$

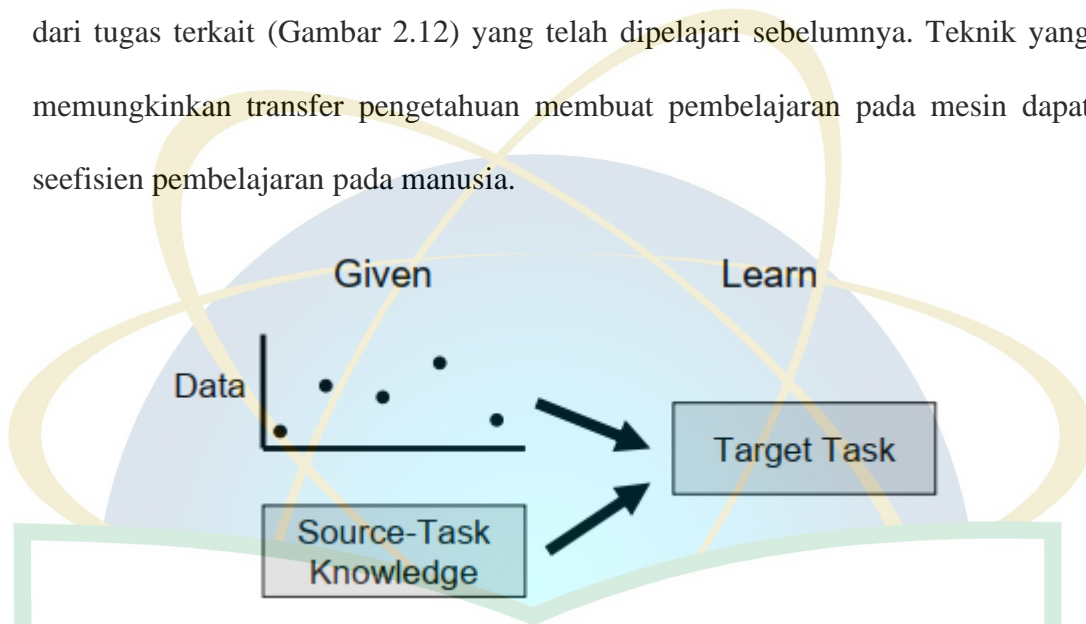
$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[\frac{\delta L}{\delta w_t} \right]^2 \quad (2.10)$$



Gambar 2.10 Perbandingan algoritma optimasi untuk *machine learning* (Dozat, 2016)

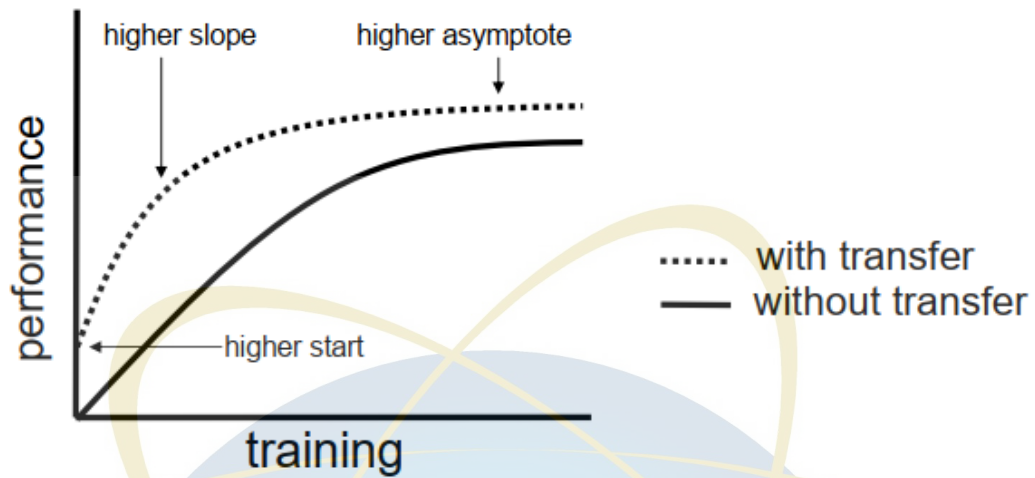
2.6 Transfer Learning

Menurut Torrey & Shavlik (2010) *transfer learning* dalam *machine learning* adalah peningkatan proses pembelajaran yang baru melalui transfer pengetahuan dari tugas terkait (Gambar 2.12) yang telah dipelajari sebelumnya. Teknik yang memungkinkan transfer pengetahuan membuat pembelajaran pada mesin dapat seefisien pembelajaran pada manusia.



Gambar 2.11 Proses pada *transfer learning* (Torrey & Shavlik, 2010)

Tujuan dari *transfer learning* adalah untuk meningkatkan pembelajaran dalam pelatihan dengan memanfaatkan pengetahuan dari pelatihan lainnya. Ada tiga keuntungan utama jika menggunakan *transfer learning*. Pertama adalah kinerja awal yang dapat dicapai dalam menggunakan pengetahuan yang ditransfer lebih baik dibandingkan dengan kinerja awal yang tidak menggunakan *transfer learning*. Kedua adalah jumlah waktu yang diperlukan untuk proses pembelajaran dengan *transfer learning* dapat lebih cepat dibandingkan yang tidak menggunakan *transfer learning*. Ketiga adalah tingkat kinerja akhir yang dapat dicapai lebih baik jika menggunakan *transfer learning* (Gambar 2.13).



Gambar 2.12 Perbandingan kinerja dengan dan tanpa *transfer learning* (Torrey & Shavlik, 2010)

2.7 Deep Residual Networks (ResNets)

Deep Residual Networks (ResNets) menyajikan kerangka belajar residual untuk memudahkan pelatihan *neural network* yang lebih dalam dari yang digunakan sebelumnya. ResNets secara eksplisit merumuskan ulang layer sebagai fungsi residual dengan mengacu pada input layer, alih-alih mempelajari fungsi yang tidak direferensikan. ResNets memberikan bukti yang komprehensif yang menunjukkan bahwa jaringan residual ini lebih mudah untuk dioptimalkan, dan dapat memperoleh akurasi yang meningkat secara signifikan. Pada *dataset* ImageNet ResNets berhasil mengevaluasi jaring residual dengan kedalaman hingga 152 *layer*, 8× lebih dalam dari VGG tetapi masih memiliki kompleksitas yang lebih rendah (He et al., 2016).

2.8 Bunga

Bunga, bagian reproduksi tanaman apa pun di divisi *Magnoliophyta*, kelompok yang biasa disebut tanaman berbunga atau angiospermae. Angiospermae,

yang terdiri atas sebagian besar spesies tanaman, dicirikan oleh keanekaragaman yang sangat besar dalam bunga yang berkisar dari bunga-bunga kecil *duckweed* (*Lemna minor*) panjang milimeter hingga bunga bangkai raksasa selebar 2 meter (*Rafflesia arnoldi*) (Rusman, Lucas-Barbosa, Poelman, & Dicke, 2019).

Angiospermae adalah tanaman benih vaskular di mana sel telur dibuahi dan berkembang menjadi biji dalam ovarium berongga tertutup. Ovarium itu sendiri biasanya tertutup oleh bunga, bagian dari tanaman angiospermae yang mengandung organ reproduksi jantan atau betina atau keduanya. Pada dasarnya, setiap bunga terdiri atas sumbu bunga yang menjadi dasar organ reproduksi penting (benang sari dan putik) dan biasanya organ aksesori (kelopak), kelopak dapat berfungsi untuk menarik serangga penyerbuk dan melindungi organ-organ penting seperti yang ditunjukkan pada Gambar 2.14.

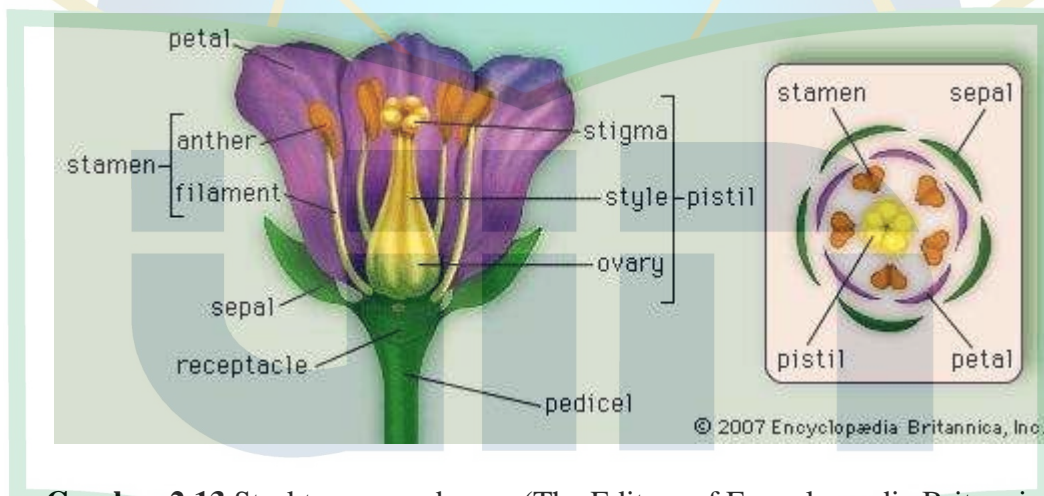
وَهُوَ الَّذِي أَنْزَلَ مِنَ السَّمَاءِ مَاءً فَأَخْرَجْنَا بِهِ نَبَاتَ كُلِّ شَيْءٍ فَأَخْرَجْنَا مِنْهُ خَضِرًا مِّنْهُ خُجْرًا مِّنْهُ حَبًّا مُّتَرَاكِبًا وَمِنَ النَّخْلِ مِن طَلْعِهَا قِنْوَانٌ دَانِيَةٌ وَجَنَّاتٍ مِّنْ أَعْنَابٍ وَالزَّيْتُونَ وَالرُّمَّانَ مُشْتَبِهًا وَغَيْرَ مُتَشَابِهٍ انظُرُوا إِلَى ثَمَرِهِ إِذَا أَثْمَرَ وَيَنْعِهِ إِنَّ فِي ذَلِكُمْ لَآيَاتٍ لِّقَوْمٍ يُؤْمِنُونَ

“Dan Dialah yang menurunkan air hujan dari langit, lalu Kami menumbuhkan dengan air itu segala macam tumbuh-tumbuhan, maka Kami keluarkan dari tumbuh-tumbuhan itu tanaman yang menghijau. Kami keluarkan dari tanaman yang menghijau itu butir yang banyak, dan dari mayang korma mengurai tangkai-tangkai yang menjulai, dan kebun-kebun anggur, dan (kami keluarkan pula) zaitun dan delima yang serupa dan yang tidak serupa. Perhatikanlah buahnya di waktu pohonnya berbuah dan (perhatikan pulalah) kematangannya. Sesungguhnya pada

yang demikian itu ada tanda-tanda (kekuasaan Allah) bagi orang-orang yang beriman.” (Q.S. Al-An’am: 99).

وَفِي الْأَرْضِ قِطْعٌ مُتَجَاوِرَةٌ وَجَنَّاتٌ مِّنْ أَعْنَابٍ وَزُرْعٌ وَنَخِيلٌ صِنَوَانٌ وَغَيْرُ صِنَوَانٍ يُسْقَى بِمَاءٍ وَاحِدٍ وَنُفِضَتِلْ بِغَضِّهَا عَلَىٰ بَعْضٍ فِي الْأُكُلِ ۚ إِنَّ فِي ذَٰلِكَ لَآيَاتٍ لِّقَوْمٍ يَعْقِلُونَ

“Dan di bumi ini terdapat bagian-bagian yang berdampingan, dan kebun-kebun anggur, tanaman-tanaman dan pohon korma yang bercabang dan yang tidak bercabang, disirami dengan air yang sama. Kami melebihkan sebahagian tanam-tanaman itu atas sebahagian yang lain tentang rasanya. Sesungguhnya pada yang demikian itu terdapat tanda-tanda (kebesaran Allah) bagi kaum yang berpikir.” (Q.S. Ar-Rad:4).



Gambar 2.13 Struktur umum bunga (The Editors of Encyclopaedia Britannica, 2019)

2.9 Citra Digital

Menurut McAndrew (2016) citra (*image*) adalah contoh dari sinyal dua dimensi, dengan koordinat horizontal dan vertikal dari gambar yang mewakili dua dimensi. Sinyal 2D adalah fungsi dari dua variabel independen $f(x, y)$, di mana nilai x dan y merupakan titik koordinat bidang datar, dan harga dari fungsi f dari setiap

pasangan titik koordinat (x,y) yang disebut dengan intensitas atau *grey level* dari suatu gambar. Ketika nilai titik x,y dan nilai intensitas f terbatas dengan nilai diskrit, maka gambar tersebut akan dapat dikatakan sebagai sebuah citra digital.

Tipe-tipe dari citra digital menurut McAndrew (2016):

- a. Biner (*Binary*). Dalam setiap piksel hanya hitam atau putih. Karena hanya ada dua nilai yang mungkin untuk setiap piksel, hanya perlu satu bit per piksel. Karena itu gambar seperti itu bisa sangat efisien dalam hal penyimpanan. Gambar yang mungkin cocok dengan representasi biner mencakup teks (cetak atau tulisan tangan), sidik jari, atau rancangan arsitektur.
- b. Skala abu-abu (*Grayscale*). Setiap piksel berwarna abu-abu, biasanya dari 0 (hitam) hingga 255 (putih). Rentang ini berarti bahwa setiap piksel dapat diwakili oleh delapan bit, atau tepat satu *byte*.
- c. *True color*, atau RGB. Di sini setiap piksel memiliki warna tertentu; warna itu digambarkan dengan jumlah merah, hijau, dan biru di dalamnya. Jika masing-masing komponen memiliki rentang 0-255, sehingga memberikan total $255^3 = 16.777.216$ kemungkinan warna berbeda dalam gambar.
- d. *Indexed*. Sebagian besar gambar berwarna hanya memiliki sub set kecil dari lebih dari enam belas juta warna yang mungkin dalam gambar. Untuk kemudahan penyimpanan dan penanganan *file*, gambar memiliki peta warna yang terkait, atau palet warna, yang ada hanyalah daftar semua warna yang digunakan dalam gambar itu. Setiap piksel memiliki nilai yang tidak memberikan warna (seperti pada gambar RGB), melainkan indeks untuk warna di peta warna.

2.10 Computer Vision

Computer vision berkaitan dengan proses memperoleh pemahaman tingkat tinggi dari gambar digital, video, kamera, atau data multidimensi. *Computer vision* berupaya mengotomatiskan tugas-tugas yang dapat dilakukan oleh sistem visual manusia dan melibatkan pengembangan dasar secara teoretis dan *algoritmik* untuk mencapai pemahaman visual secara otomatis. Dengan kata lain, tujuan akhir dari *computer vision* adalah pemahaman gambar, kemampuan tidak hanya untuk memulihkan struktur gambar tetapi juga untuk mengetahui apa data yang diwakilinya. Sebagai disiplin teknologi dan teknik, *computer vision* berupaya menerapkan teori dan modelnya untuk pembangunan sistem dan aplikasi *computer vision* (Borji, 2018).

2.10.1 Pattern Recognition

Pattern Recognition atau pengenalan pola adalah sub bidang *machine learning* yang berfokus pada pengenalan pola dalam data. Suatu pola dapat didefinisikan secara samar-samar sebagai suatu entitas atau konsep yang dapat diberi nama seperti pola sidik jari, pola tulisan tangan, pola fitur wajah manusia, pola sinyal vokal atau pola urutan DNA. Pengenalan pola adalah tekniknya; sebagian besar dikategorikan sebagai teknik pembelajaran mesin, yang dapat membuat mesin memahami lingkungan dan berbagai pola yang muncul di lingkungan itu. Teknik pengenalan pola banyak digunakan pada *computer vision* (Ali, Shahzad, & Shahzad, 2017).

2.10.2 Object Recognition

Object Recognition adalah metode mengidentifikasi objek dalam suatu gambar. Mendeteksi objek dalam gambar adalah kemampuan penting dari aplikasi *computer vision*. Deteksi / pengenalan objek digunakan dalam navigasi robot, pengenalan bagian wajah dan tubuh, deteksi penyakit dan kanker, objek dalam citra satelit, pengenalan tulisan tangan, dan banyak lagi (Gollapudi, 2019).

2.11 Tools

2.11.1 Python

Python merupakan bahasa pemrograman tingkat tinggi yang bertipe *interpreted language* yang banyak digunakan oleh non-programmer dan ilmuwan. Secara desain, kode Python dapat bekerja pada sebagian besar platform modern (Bingol & Krishnamurthy, 2019). Contoh kode bahasa pemrograman python sebagai berikut:

```
num1 = 1.5
num2 = 6.3
# Add two numbers
sum = float(num1) + float(num2)
# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

Kode tersebut berfungsi untuk menambahkan dua buah bilangan desimal.

2.11.2 Tensorflow

TensorFlow adalah sistem *machine learning* yang beroperasi pada skala besar dan dalam lingkungan yang heterogen. Tensorflow memetakan node grafik aliran data di banyak mesin dalam sebuah *cluster*, dan di dalam mesin di beberapa perangkat komputasi, termasuk CPU, GPU, dan ASIC yang dirancang khusus yang

dikenal sebagai *Tensor Processing Units* (TPUs). Arsitektur ini memberikan fleksibilitas kepada pengembang aplikasi: sedangkan dalam "server parameter" desain yang dibangun ke dalam sistem, TensorFlow memungkinkan pengembang untuk bereksperimen dengan optimasi baru dan algoritma untuk proses pelatihan. TensorFlow mendukung berbagai aplikasi, dengan fokus pada pelatihan dan inferensi pada *deep neural networks* (Abadi et al., 2016).

2.11.3 Jupyter Notebooks

Jupyter Notebook dirancang untuk mendukung alur kerja komputasi ilmiah, mulai dari eksplorasi secara interaktif hingga penerbitan catatan. Kode dalam Jupyter Notebook disusun menjadi *cells*, potongan yang dapat dimodifikasi dan dijalankan secara individual. Jupyter bertujuan untuk membawa notebook ke khalayak yang lebih luas. Jupyter Notebook merupakan proyek *open source*, yang dapat bekerja dengan kode dalam berbagai bahasa pemrograman. *Backend* bahasa yang berbeda, yang disebut kernel, berkomunikasi dengan Jupyter menggunakan protokol umum. Lebih dari 50 *backend* tersebut telah tersedia, untuk bahasa mulai dari C++ hingga Bash. Jupyter Notebook tumbuh dari proyek IPython, yang awalnya menyediakan antarmuka ini hanya untuk bahasa Python. Jupyter Notebook dapat diakses melalui browser web. File di simpan dalam format JSON, dengan ekstensi *.ipynb* (Kluyver et al., 2016).

2.12 Penelitian Sejenis

Penelitian sejenis merupakan penelitian yang telah dilakukan sebelumnya dan terkait dengan topik yang dibahas. Pada Tabel 2.2 menampilkan penelitian-

penelitian terdahulu yang sesuai dengan penelitian ini. Penelitian sejenis digunakan oleh penulis sebagai perbandingan dan acuan dalam melakukan penelitian ini.



Tabel 2.2 Penelitian sejenis

No	Peneliti	Metode	Tools	Hasil	Variabel	Kinerja	Kontribusi	Kelebihan/Kekurangan
1	Albadarneh & Ahmad (2017)	<i>Support Vector Machine (SVM)</i>	MATLAB 2014	Menghasilkan dataset dan model untuk klasifikasi beberapa jenis bunga di Yordania.	19 spesies bunga dengan total 513 data.	Akurasi: 83.52%.	Mengusulkan sistem pengenalan bunga, yang memiliki keunggulan dengan pengklasifikasi yang cepat.	-Hanya mengekstrasi fitur penting untuk meningkatkan akurasi. - Pengenalan jenis bunga yang terbatas pada daerah timur tengah.
2	Almogdady et al. (2018)	<i>Back-Propagation Artificial Neural Networks (ANNs)</i>	MATLAB 2013	Menghasilkan model untuk mengklasifikasikan 102 jenis bunga.	102 spesies bunga dengan total 8189 data.	Akurasi: 81.19%.	Menghasilkan teknik pemrosesan gambar berdasarkan sistem CAFR dan ANNs.	Menggabungkan ANNs dengan teknik pemrosesan gambar.
3	Tiay et al (2014)	<i>K-nearest neighbor</i>		Menghasilkan sistem yang mampu mendeteksi dan	10 spesies bunga, dengan masing-	Akurasi: 80%.	Memberikan pendekatan baru untuk mengenali dan mengidentifikasi	Menggunakan algoritma Hu's seven moment untuk mendeteksi tepi dan

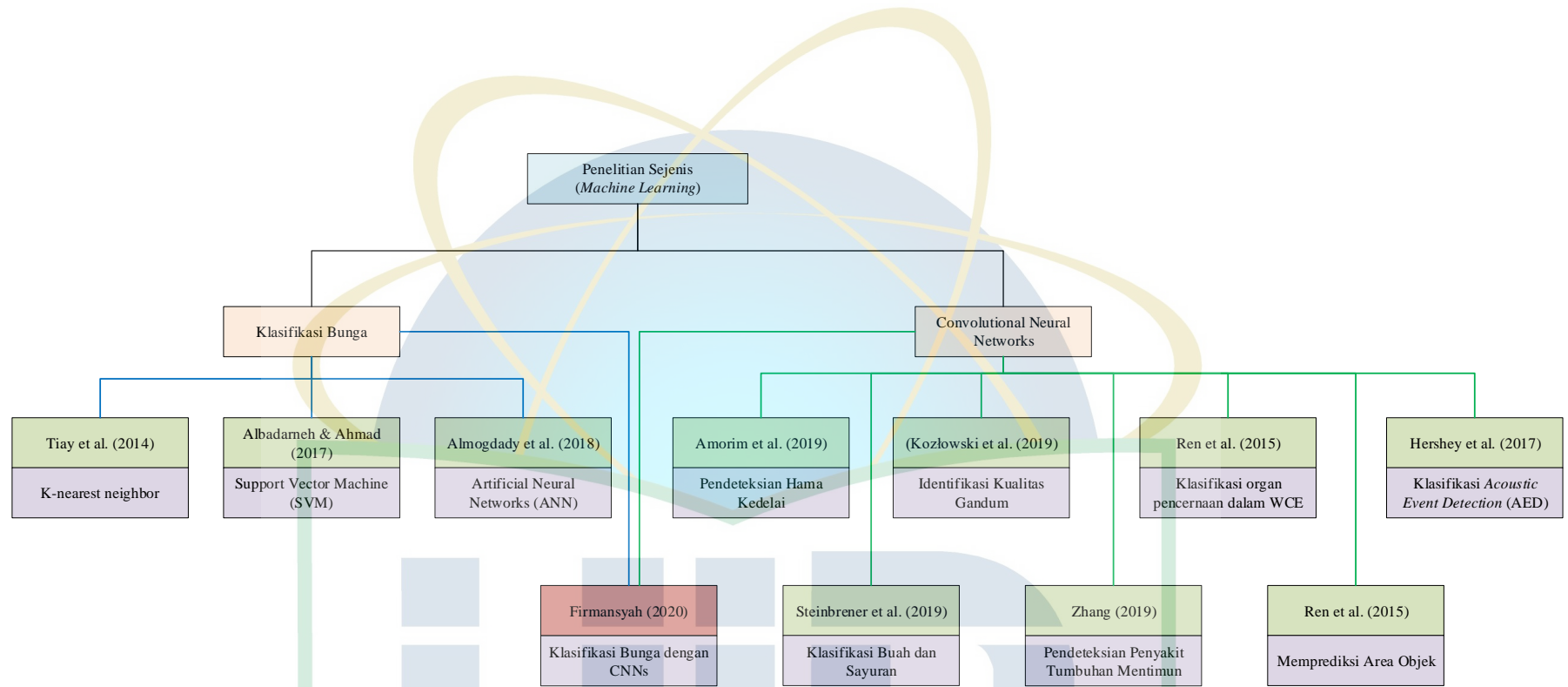
				mengklasifikasikan bunga.	masing 50 data <i>test</i> dan 100 data <i>training</i> .		si beberapa jenis bunga dan tanaman.	karakteristik warna untuk mengklasifikasikan bunga.
4	Amorim et al. (2019)	<i>Convolutional Neural Network</i> (CNN)		Menghasilkan model untuk mendeteksi penyakit pada daun tanaman kedelai dan hama pada tanaman kedelai.	3000 gambar untuk 6 kelas untuk penyakit pada daun dan 5000 gambar untuk 13 label pada hama kedelai.	Akurasi: 98%.	Melakukan penilaian dampak pada penggunaan teknik <i>semi-supervised learning</i> untuk propagasi label dan pelatihan menggunakan arsitektur CNN.	- Menggabungkan teknik <i>supervised learning</i> dan <i>unsupervised learning</i> .
5	Kozłowski et al. (2019)	<i>Convolutional Neural Network</i> (CNN)		Menghasilkan nilai kinerja berbagai arsitektur CNN dalam mengidentifikasi kualitas varietas gandum.	6 jenis gandum, dengan masing-masing gambar gandum	Akurasi: 93%.	Menguji beberapa arsitektur CNN untuk mengidentifikasi kualitas dari	- Dataset yang berjumlah besar. - Menggunakan metode <i>transfer learning</i> untuk meningkatkan akurasi klasifikasi.

					berjumlah sekitar 10.000 gambar.		berbagai jenis gandum.	
6	Zou et al. (2015)	<i>Convolutional Neural Network</i> (CNN)		Klasifikasi organ pencernaan dalam <i>Wireless Capsule Endoscopy</i> (WCE).	60.000 citra untuk <i>training</i> dan 15.000 citra untuk validasi.	Akurasi: 95%	Mengusulkan metode untuk memperoleh mengklasifikasi organ pencernaan dalam gambar WCE	Menggunakan metode hasil pengembangan CNN yaitu DCNN (<i>Deep Convolutional Neural Networks</i>)
7	Steinbrener et al. (2019)	<i>Convolutional Neural Networks</i> (CNN)	Python	Menghasilkan dataset dan model dari arsitektur CNN GoogleNet dengan <i>transfer learning</i> untuk klasifikasi buah dan sayuran.	13 buah dan sayuran dengan total 2700 gambar.	Akurasi: 92.23%.	Pendekatan baru yang memanfaatkan informasi tambahan yang disediakan oleh gambar hyperspectral untuk mencapai akurasi yang lebih baik bahkan untuk arsitektur CNN dengan data	Menggunakan pendekatan pseudo-RGB untuk meningkatkan akurasi.

							pelatihan (dataset) yang terbatas.	
8	Zhang et al. (2019)	<i>Convolutional Neural Network (CNN)</i>	Berkeley Vision and Learning Center (BVLC), Tensorflow dan Python 3.4	Menghasilkan model untuk mendeteksi 6 jenis penyakit pada tumbuhan mentimun.	600 gambar mentimun dari 6 penyakit pada tumbuhan mentimun dan 100 gambar daun mentimun sehat.	Akurasi: 94.65%	Penelitian ini mengusulkan arsitektur menggabungkan CNN dan GPDCNN yang digunakan untuk mengenali penyakit pada tumbuhan mentimun.	Menggunakan <i>a global pooling dilated convolutional neural network (GPDCNN)</i> untuk meningkatkan akurasi.
9	Ren et al. (2015)	<i>Convolutional Neural Network (CNN)</i>	PASCAL VOC, Caffe	Menghasilkan metode baru yaitu <i>Region Proposal Networks (RPNs)</i> untuk meningkatkan efisiensi dan akurasi dalam memprediksi kawasan objek.	10.000 gambar untuk 20 kategori.	Akurasi: 73.2%	Memperkenalkan <i>Region Proposal Network (RPN)</i> yang berbagi fitur konvolusional gambar secara penuh dengan jaringan deteksi, sehingga	<ul style="list-style-type: none"> - Memiliki kinerja pemerosesan yang lebih tinggi dibandingkan metode sejenis, yaitu 5-17 fps. - Sumber daya yang rendah untuk melakukan training data.

							memungkinkan pengajuan kawasan yang hampir <i>cost-free</i> .	
10	Hershey et al. (2017)	<i>Convolutional Neural Network (CNN)</i>	TensorFlow dan Python	Menghasilkan model yang menggunakan <i>embeddings</i> untuk pengklasifikasi <i>Acoustic Event Detection (AED)</i>	70 juta <i>soundtrack</i> video (5.24 juta jam) dengan 30,871 label video.	Akurasi: 95.9%	Membandingkan penggunaan arsitektur CNN AlexNet, VGG, Inception, dan ResNet	Menggunakan dataset yang besar (70juta)

Penelitian ini bertujuan untuk mengklasifikasikan bunga, terdapat penelitian-penelitian terdahulu yang membahas proses pengklasifikasian bunga dengan berbagai algoritma atau metode, Tiay et al. (2014) dengan K-nearest neighbor, Albadarneh & Ahmad (2017) dengan SVM, dan Almogdady et al. (2018) dengan ANN. Namun belum ada penelitian klasifikasi bunga yang menggunakan arsitektur atau metode CNN dalam melakukan klasifikasi terhadap bunga. Terdapat beberapa penelitian-penelitian untuk berbagai kasus yang menggunakan CNN dalam berbagai bidang, baik untuk kesehatan dalam penelitian Ren et al. (2015), maupun kualitas tanaman pada penelitian Kozłowski et al. (2019), Amorim et al. (2019) dan Zhang et al. (2019), serta berbagai bidang lainnya. Penelitian ini sendiri mendekati penelitian Kozłowski et al. (2019) yang menggunakan lebih dari satu model untuk mendapatkan kinerja terbaik dari CNN, pada penelitian ini metode *transfer learning* digunakan pada model untuk melakukan klasifikasi. Kelebihan pada penelitian ini menggunakan dua pendekatan yaitu pelatihan arsitektur CNN *from the scratch* dan CNN dengan *transfer learning*, serta dengan menggunakan algoritma optimasi Nadam yang merupakan algoritma optimasi terbaru yang memiliki kinerja lebih baik dari algoritma optimasi untuk *neural networks* lainnya (Dogo et al., 2018). Arsitektur pada CNN pada penelitian ini menggunakan arsitektur yang dikembangkan dari Steinbrener et al. (2019) dengan menambahkan *layer dropout* untuk menghindari *overfitting*. Sedangkan untuk pemilihan arsitektur model untuk *transfer learning* merujuk pada penelitian Hershey et al. (2017).



Gambar 2.14 Penelitian Sejenis

BAB 3

METODE PENELITIAN

3.1 Populasi dan sampel penelitian

Data dalam penelitian ini menggunakan dua *dataset*, pemilihan kedua dataset ditujukan untuk membandingkan CNN dengan model lainya, *dataset* oxford17 untuk membandingkan dengan *Support Vector Machine* (SVM) dari penelitian Almogdady et al. (2018) dengan hasil akurasi 83.52% dan *Artificial neural networks* (ANN) pada penelitian Albadarneh & Ahmad (2017) dengan hasil akurasi 81.19%.

Data pertama adalah oxford17 dari penelitain Nilsback & Zisserman (2006) yang terdiri atas 17 jenis tanaman bunga dan masing-masing jenis berjumlah sekitar 80 gambar, sehingga total terdapat 1,360 data gambar. Serta data yang terakhir adalah oxford102 dari penelitian Nilsback & Zisserman (2008) yang memiliki 102 jenis bunga dan setiap jenis terdiri atas 40-258 gambar. *Dataset* ini memiliki total data 8,189 gambar. Dari jumlah tersebut, diambil data sebesar 80% dari jumlah data digunakan untuk proses pembelajaran (*training*) model dan 20% atau digunakan sebagai data untuk melakukan tes atau validasi (*validation*) model.

3.2 Kerangka Penelitian

Kerangka penelitian terdapat pada Gambar 3.1. Studi pendahuluan untuk menentukan objek penelitian dan sesuai dengan tema penelitian yang menjadi fokus peneliti. Dengan melakukan studi pendahuluan diharapkan dapat memperoleh

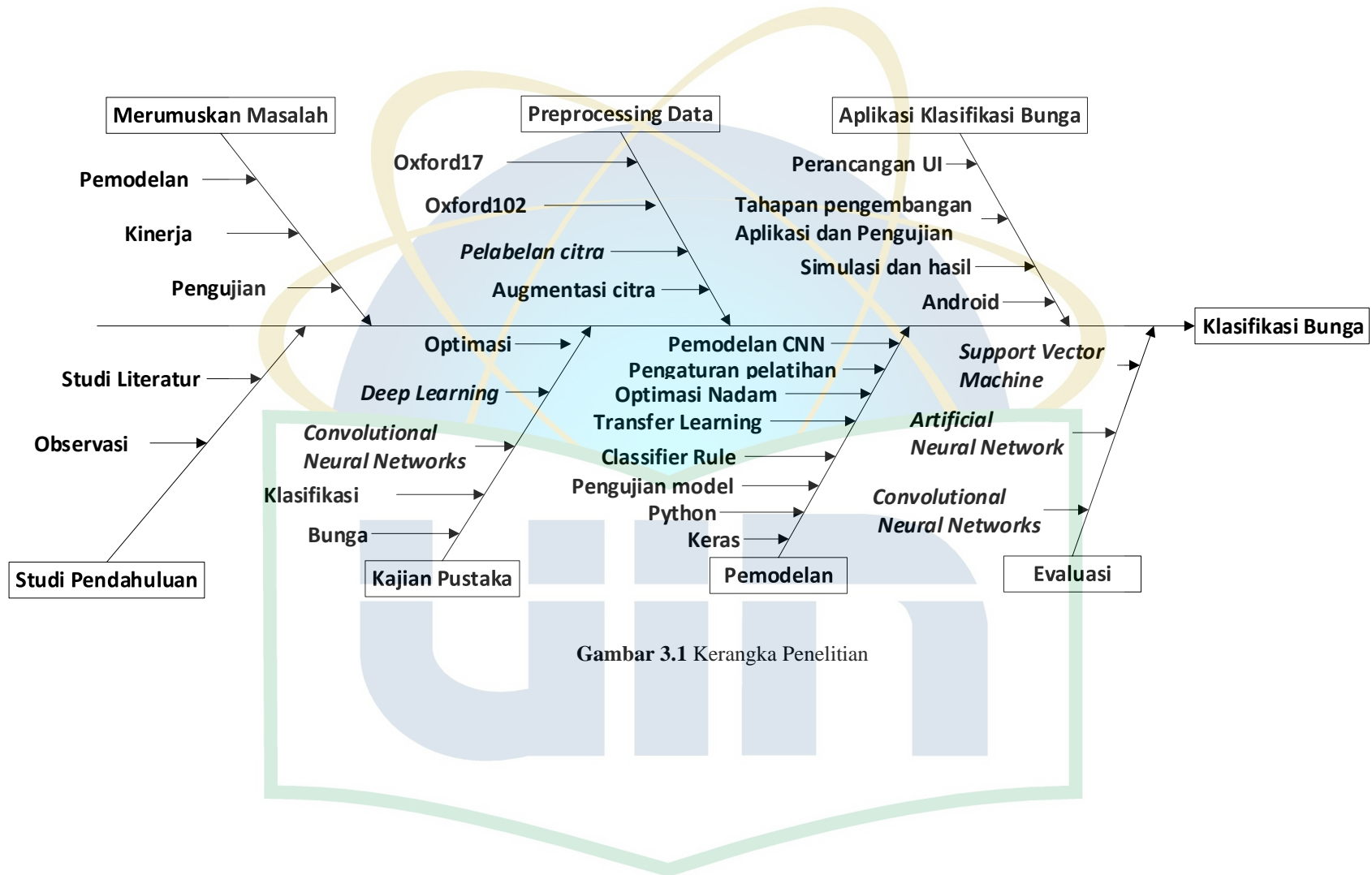
informasi-informasi tentang permasalahan yang ada. Dalam proses ini peneliti melakukan observasi dan studi literatur mengenai permasalahan yang berkaitan dengan *deep learning*. Merumuskan masalah dilakukan untuk menjadi alasan mengapa penelitian dilakukan dan menjadi pedoman yang dilakukan oleh peneliti dalam menyelesaikan karya tulis.

Kajian pustaka mengenai *deep learning*, bunga, *convolutional neural networks*, klasifikasi dan beberapa hal lainnya yang terkait dengan penelitian ini. Pemodelan dilakukan untuk membentuk model dari *convolutional neural networks*, tujuan model pada penelitian ini adalah untuk melakukan klasifikasi terhadap bunga. Sebelum tahap pemodelan, peneliti melakukan pemerosesan data, data penelitian adalah oxford17 dan oxford102, augmentasi citra untuk menghindari terjadinya *overfitting* pada model dan pelabelan citra bunga untuk mengetahui jenis bunga pada citra gambar. Perancangan model CNN ini menggunakan model yang dikembangkan dari (Kozłowski et al., 2019) dan (Steinbrener et al., 2019), peneliti menambahkan layer *dropout* untuk mencegah terjadinya *overfitting* pada model. Penelitian ini juga menggunakan metode *transfer learning* dengan ResNet50 untuk mengetahui perbandingan model yang dilatih dengan *transfer learning* dan yang tidak. Pada proses optimasi *hyperparameter* peneliti menggunakan algoritma Nadam. Proses pembuatan model menggunakan bahasa pemrograman Python dengan *library* keras.

Tahap selanjutnya untuk mengetahui kinerja dari model yang telah dibuat, maka perlu dibuat aplikasi untuk pengujian. Pengujian dilakukan untuk mengetahui sejauh mana efektifitas dan efisiensi dari model yang telah dibuat, pada tahapan

aplikasi klasifikasi bunga terdapat beberapa langkah, yaitu perancangan *user interface*, pengembangan aplikasi pada *platform* Android dan pengujian aplikasi, serta simulasi dan hasil. Interpretasi dan evaluasi dilakukan untuk melakukan penggabungan terhadap sebuah hasil dari analisis dengan berbagai macam pertanyaan, kriteria, maupun pada sebuah standar tertentu guna untuk dapat menciptakan sebuah makna dari adanya sebuah data yang telah dikumpulkan untuk mencari sebuah jawaban terhadap permasalahan yang terdapat di dalam sebuah penelitian. Evaluasi juga untuk membandingkan metode CNN dengan metode ANN dan SVM serta untuk mengukur hasil atau dampak dari penelitian ini.





Gambar 3.1 Kerangka Penelitian

BAB 4

HASIL DAN PEMBAHASAN

4.1 *Preprocessing Data*

Tahapan *preprocessing* terhadap data citra dilakukan dengan melakukan pengumpulan data (*gathering*), pelabelan citra, pengubahan ukuran *pixel* pada citra (*resize*), dan melakukan augmentasi pada citra dari keseluruhan citra *training*.

4.1.1 Pengumpulan data citra

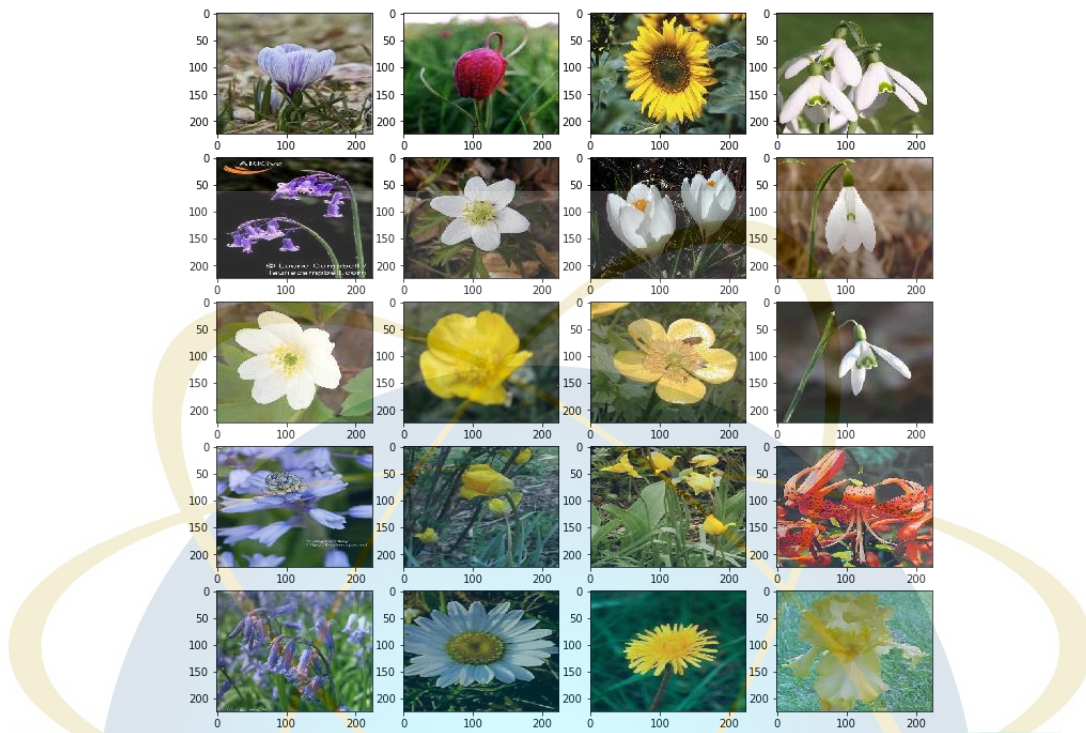
a. Oxford17

Data pertama adalah oxford17 yang terdiri atas 17 jenis tanaman bunga (*Daffodil, Snowdrop, Lily Valley, Bluebell, Crocus, Tigerlily, Tulip, Fritillary, Sunflower, Daisy, Colts Foot, Dandelion, Cowslip, Buttercup, Windflower, Pansy*) dan masing-masing jenis berjumlah sekitar 80 gambar, sehingga total terdapat 1.360 data gambar. Contoh citra gambar bunga dapat dilihat pada Gambar 4.1.

b. Oxford102

Serta data yang kedua adalah oxford102 yang memiliki 102 jenis bunga dan setiap jenis terdiri atas 40 sampai 258 gambar. Dataset ini memiliki total data 8.189 gambar. Contoh citra gambar bunga dapat dilihat pada Gambar 4.2. Terdiri atas: *pink primrose, hard-leaved pocket orchid, canterbury bells, sweet pea, english marigold, tiger lily, moon orchid, bird of paradise, monkshood, globe thistle, snapdragon, colts foot, king protea, spear thistle, yellow iris, globe-flower, purple*

coneflower, peruvian lily, balloon flower, giant white arum lily, fire lily, pincushion flower, fritillary, red ginger, grape hyacinth, corn poppy, prince of wales feathers, stemless gentian, artichoke, sweet william, carnation, garden phlox, love in the mist, mexican aster, alpine sea holly, ruby-lipped cattleya, cape flower, great masterwort, siam tulip, lenten rose, barbeton daisy, daffodil, sword lily, poinsettia, bolero deep blue, wallflower, marigold, buttercup, oxeye daisy, common dandelion, petunia, wild pansy, primula, sunflower, pelargonium, bishop of llandaff, gaura, geranium, orange dahlia, pink-yellow dahlia, cautleya spicata, japanese anemone, black-eyed susan, silverbush, californian poppy, osteospermum, spring crocus, bearded iris, windflower, tree poppy, gazania, azalea, water lily, rose, thorn apple, morning glory, passion flower, lotus, toad lily, anthurium, frangipani, clematis, hibiscus, columbine, desert-rose, tree mallow, magnolia, cyclamen, watercress, canna lily, hippeastrum, bee balm, ball moss, foxglove, bougainvillea, camellia, mallow, mexican petunia, bromelia, blanket flower, trumpet creeper, blackberry lily).



Gambar 4.1 Oxford17 (Nilsback & Zisserman, 2006)



Gambar 4.2 Oxford102 (Nilsback & Zisserman, 2008)

4.1.2 Pelabelan Data

Proses pemberian label pada data berfungsi untuk memberikan nama terhadap data untuk dapat dikenali. Peneliti membuat dua folder utama yaitu *folder train* dan *folder test / validation*. *Folder train* berfungsi untuk menaruh data untuk diproses pada proses pembelajaran, sedangkan *folder test / validation* berfungsi untuk memvalidasi data pada proses *training*. Pada setiap *subfolder* diberi nama bunga dan diisi dengan data sesuai namanya seperti pada Gambar 4.3. Perbandingan jumlah data pada *folder train* berjumlah 80% dan *folder test* atau *validation* sebesar 20% dari jumlah data.



Gambar 4.3 Data yang telah diberi label

4.1.3 Augmentasi Data

Proses agumentasi terhadap data citra bunga untuk proses *training* dilakukan untuk mencegah terjadinya *overfitting* (memiliki kinerja baik selama pelatihan, tetapi buruk pada data baru). Pengaturan augmentasi data citra secara otomatis menggunakan kode sebagai berikut:

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=10,
    width_shift_range=0.2,
```

```

height_shift_range=0.2,

shear_range=0.2,

zoom_range=0.2,

horizontal_flip=True,

vertical_flip=False
)

```

- *Rescale* = 1./255. Berfungsi berfungsi untuk mengubah ukuran data piksel RGB gambar (0-255) menjadi rentang angka (0-1) untuk memudahkan proses training data.
- *Rotation_range* = 10. Berfungsi untuk mengubah rotasi gambar secara acak, angka 10 menunjukkan besaran derajat rotasi terhadap citra gambar.
- *width_shift_range* = 0.2. Berfungsi untuk mengatur posisi gambar pada lebar gambar, angka 0.2 menunjukkan bahwa citra dapat secara acak berada maksimal 20% dari samping atau lebar awal citra gambar.
- *height_shift_range* = 0.2. Berfungsi untuk mengatur posisi gambar pada tinggi gambar, angka 0.2 menunjukkan bahwa citra dapat secara acak berada maksimal 20% dari atas bawah atau tinggi awal citra gambar.
- *shear_range* = 0.2. Merupakan sudut geser dalam arah berlawanan arah jarum jam dalam derajat.
- *zoom_range* = 0.2. Berfungsi untuk membesarkan citra gambar secara acak, 0.2 menunjukkan intensitas pembesaran pada citra gambar.
- *horizontal_flip* = *True*. Berfungsi untuk membalik secara horizontal citra gambar dengan acak, di atur dengan nilai benar.

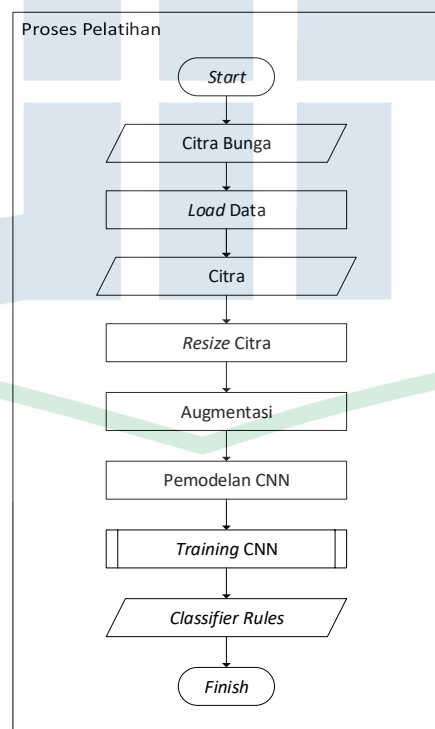
- *vertical_flip = False*. Berfungsi untuk membalik secara vertikal citra gambar dengan acak, di atur dengan nilai benar. Sehingga akan menghasilkan citra gambar yang lebih beragam seperti yang ditunjukan pada Gambar 4.4.



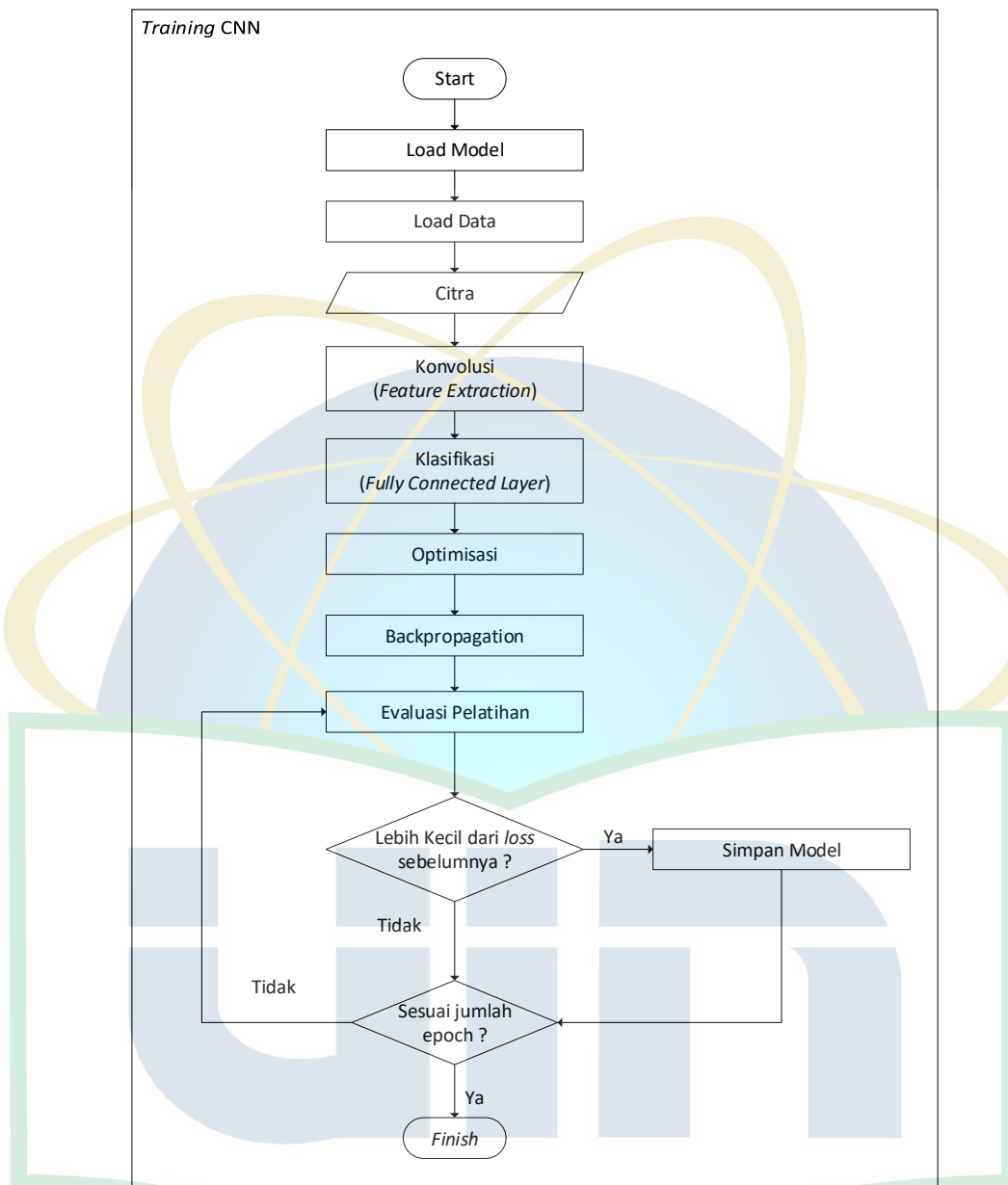
Gambar 4.4 Hasil gambar yang telah diaugmentasi

4.2 Pemodelan dan Pelatihan

Dalam pembuatan model atau pemodelan terdiri atas beberapa proses, untuk model klasifikasi bunga membutuhkan input citra bunga, sebagai sumber data pembelajaran. Data citra yang dimuat ke dalam dua variabel, yaitu citra *training* dan citra validasi, dan pada masing-masing citra dilakukan pengubahan ukuran citra menjadi 224×224 pixel. Pada data citra terjadi proses augmentasi data untuk menghasilkan model dengan kinerja yang baik, menghindari *overfitting*, dan memperkaya data citra *training*. Tahapan selanjutnya adalah perancangan model, pada penelitian ini menggunakan dua model CNN *from the scratch* dan CNN dengan *transfer learning* ResNet. Setelah tahap perancangan model CNN, kemudian dilakukan *training* untuk menghasilkan model (*classifier rules*) yang dapat mengklasifikasikan bunga.



Gambar 4.5 Flowchart pemodelan CNN



Gambar 4.6 Flowchart training CNN

4.2.1 Pemodelan *Convolutional Neural Network*

a. Pemodelan CNN

Model untuk pelatihan model *deep learning* dengan *convolutional neural networks*, menggunakan empat *convolutional layer* dan tiga *fully connected layer* (Gambar 4.7).


```

model = Sequential()

model.add(Conv2D(
    filters = 16, kernel_size = (5,5),
    padding = 'valid', activation = 'relu',
    input_shape = (224,224,3)))

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

```

Pada lapisan pertama menerima input gambar dengan ukuran panjang 224 piksel, lebar 224 piksel, dan 3 *channel* RGB dan kemudian dilakukan proses *convolution* pertama (Gambar 4.7 poin 1) mengekstraksi fitur dari citra dengan 16 *filter*, *kernel* 5×5 , *padding valid* dan menggunakan fungsi aktivasi ReLU. Selanjutnya proses *pooling* untuk memperkecil ukuran citra dengan ukuran 2×2 , *stride* 2×2 dan menggunakan *max pooling* untuk menghasilkan citra dengan ukuran 50% lebih kecil. Lapisan ini menghasilkan citra dengan ukuran 64×64 *pixel*.

```

model.add(Conv2D(
    filters = 32, kernel_size = (5,5),
    padding = 'valid', activation = 'relu'))

model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Dropout(0.25))

```

Selanjutnya pada lapisan kedua (Gambar 4.7 poin 2) menerima input gambar 64×64 *pixel* dan kemudian dilakukan proses *convolution* mengekstraksi fitur dari citra dengan 32 *filter*, *kernel* 5×5 , *padding valid* dan menggunakan fungsi aktivasi

ReLU. Sama seperti proses *pooling* sebelumnya, pada lapisan ini menggunakan *pooling* dengan ukuran 2×2 , *stride* 2×2 dan menggunakan *max pooling*. Layer *Dropout* juga digunakan untuk menghindari terjadinya *overfitting*.

```
model.add(Conv2D(
    filters = 64, kernel_size = (5,5),
    padding = 'valid', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
```

Lapisan ketiga (Gambar 4.7 poin 3) menerima input gambar 32×32 *pixel* dan kemudian dilakukan proses *convolution* mengekstraksi fitur dari citra dengan 64 *filter*, *kernel* 5×5 , *padding* *valid* dan menggunakan fungsi aktivasi ReLU. Sama seperti proses *pooling* sebelumnya, pada lapisan ini menggunakan *pooling* dengan ukuran 2×2 , *stride* 2×2 dan menggunakan *max pooling*.

```
model.add(Conv2D(
    filters = 128, kernel_size = (5,5),
    padding = 'valid', activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))
```

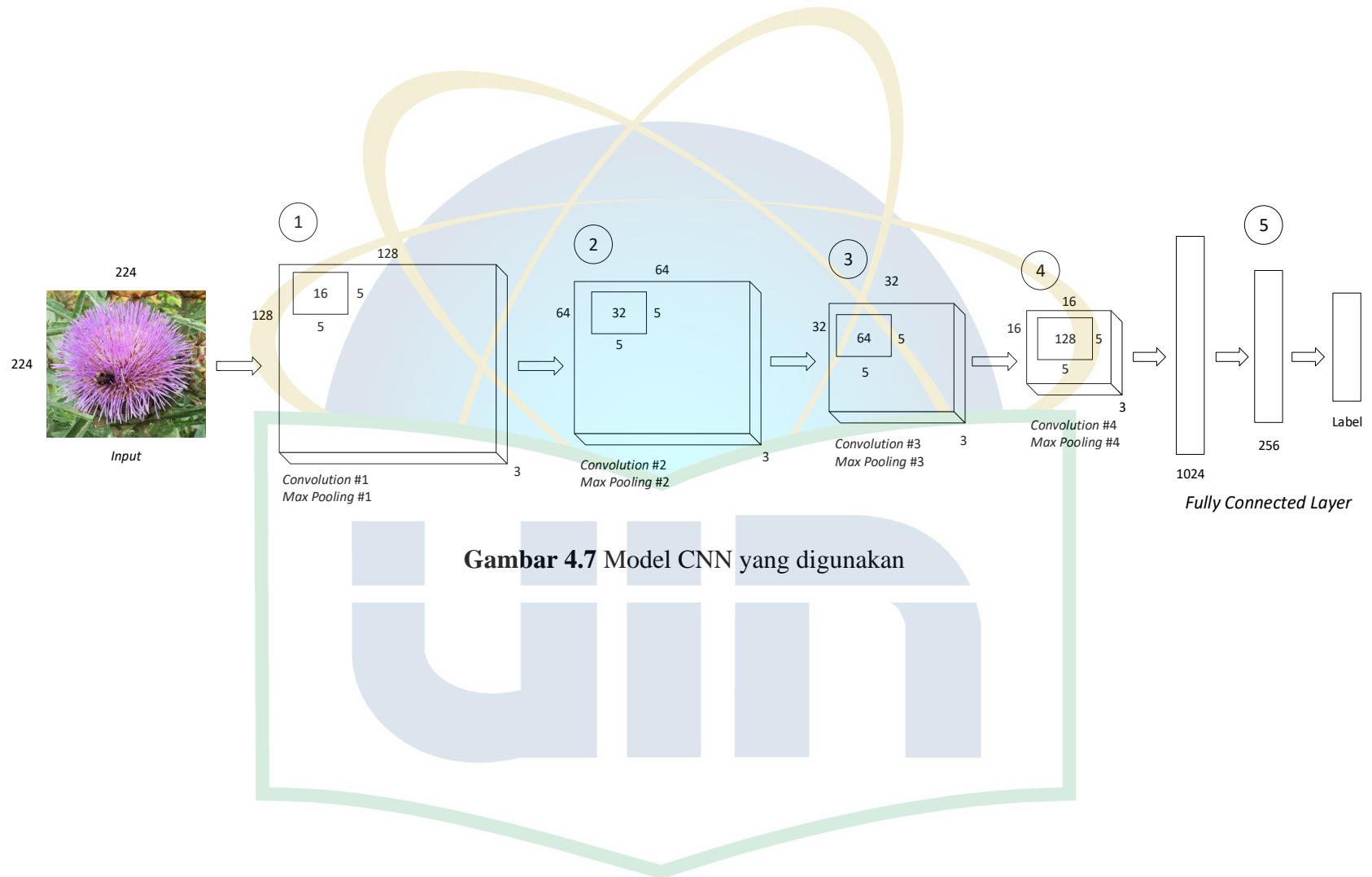
Lapisan keempat (Gambar 4.7 poin 4) menerima input gambar 16×16 *pixel* dan kemudian dilakukan proses *convolution* mengekstraksi fitur dari citra dengan 128 *filter*, *kernel* 5×5 , *padding* *same* dan menggunakan fungsi aktivasi ReLU. Sama seperti proses *pooling* sebelumnya, pada lapisan ini menggunakan *pooling*

dengan ukuran 2×2 , *stride* 2×2 dan menggunakan *max pooling*, serta ditambahkan *dropout layer*.

```
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation = "softmax"))
```

Selanjutnya semua parameter dihubungkan menjadi sebuah vektor (*flatten*) pada Flatten untuk masuk ke *fully connected layer*. Masuk ke *Dense layer* (Gambar 4.7 poin 5), hasil tersebut diperkecil menjadi 1024 *output* dan kemudian diperkecil kembali menjadi 256 *output*, lalu masuk *layer* terakhir proses klasifikasi dilakukan dengan menggunakan *softmax function* sehingga menghasilkan jumlah kelas yang sesuai dengan kategori atau kelas pada data.



Gambar 4.7 Model CNN yang digunakan

b. Pemodelan CNN ResNet *transfer learning*

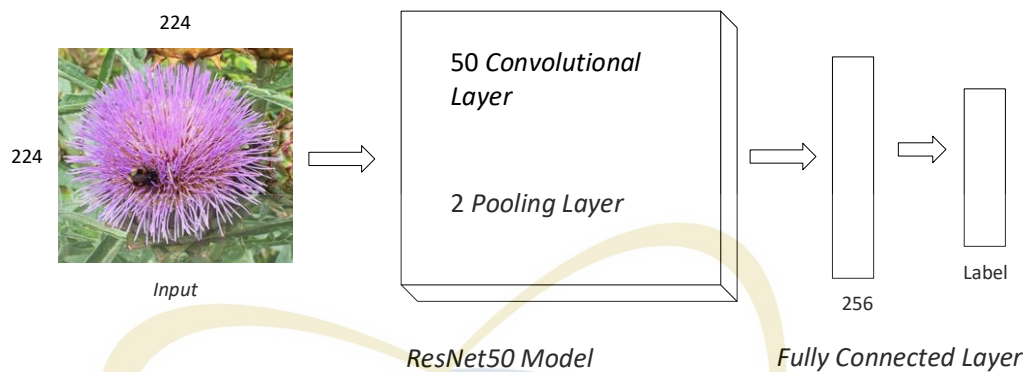
Dalam pendekatan dengan metode *transfer learning*, *layer* untuk *feature extraction* menggunakan *layer* dari ResNet50 dan untuk *fully connected layer* terdiri atas dua *layer*.

```
WEIGHTS_PATH_NO_TOP =
"/kaggle/input/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5"
base_model = ResNet50(include_top=False, pooling='avg')
base_model.load_weights(WEIGHTS_PATH_NO_TOP)
```

Proses pertama adalah memuat bobot (*weight*) dari model yang telah dilatih sebelumnya dan hanya mengambil bagian *feature extraction layer*, tanpa mengambil *fully connected layer*. Selanjutnya untuk *pooling* menggunakan *average pooling* sesuai dengan *pooling* yang digunakan pada ResNet50.

```
model = Sequential()
model.add(base_model)
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
base_model.trainable=False
```

Pada *fully connected layer* menggunakan *dense layer* dimana hasil tersebut diperkecil menjadi 256 *output* lalu masuk layer terakhir dengan *output* sesuai jumlah kategori/kelas dengan *softmax* sebagai fungsi pengklasifikasi.



Gambar 4.8 Model CNN *transfer learning* yang digunakan

4.2.2 Optimasi

```

model.compile(loss='categorical_crossentropy',
              optimizer=keras.optimizers.Nadam(),
              metrics = ['accuracy'])
  
```

Proses pelatihan memerlukan beberapa pengaturan, penggunaan *loss function categorical_crossentropy* karena data pelatihan berbentuk kategori, sehingga pengevaluasian *loss* dilakukan berdasarkan kategori data. Penelitian ini menggunakan optimasi Nadam, penggunaan optimasi Nadam dipilih karena optimasi ini lebih efisien dan efektif serta tidak menggunakan banyak sumber daya. Metrik yang digunakan dalam penelitian ini adalah akurasi.

4.2.3 Pelatihan

a. Pengaturan Pelatihan

Untuk mengatur jalannya proses pelatihan diperlukan pengaturan pelatihan. Pengaturan pelatihan bertujuan untuk mendapatkan model terbaik secara efisien dan efektif.

```

nb_train_samples = train.shape[0]
nb_validation_samples = val.shape[0]
  
```



```

epochs = 50

keras_model = 'flowers_cnn_model.h5'

checkpoint = ModelCheckpoint(keras_model,

                             monitor='val_loss',
                             mode='auto',
                             save_best_only=True)

```

Variabel `keras_model` bertujuan untuk menyimpan model ke dalam direktori yang telah disiapkan. Tujuan `nb_train_samples` untuk menentukan jumlah data *training* dan `nb_validation_samples` untuk menentukan jumlah data untuk validasi. Pada proses pelatihan terhadap *dataset* terjadi pengulangan (*epoch*) beberapa kali untuk mendapatkan kinerja model yang paling maksimal, pada setiap kali pengulangan nilai akurasi dapat naik dan turun, maka untuk mendapatkan nilai terbaik dibutuhkan fungsi *checkpoint*. *Checkpoint* di set untuk hanya menyimpan model dengan nilai *loss error* terendah, nilai *loss function* yang rendah berbanding lurus dengan tingkat akurasi yang dimiliki model.

```

history = model.fit_generator(train_generator,

                              steps_per_epoch=nb_train_samples // batch_size,

                              validation_steps=nb_validation_samples // batch_size,

                              epochs=epochs,

                              callbacks=[checkpoint, earlystop],

                              validation_data=validation_generator)

```

Proses terakhir adalah melakukan pengaturan proses jalannya pelatihan. *Fit generator* dipilih untuk dapat melakukan pelatihan secara paralel, sehingga proses

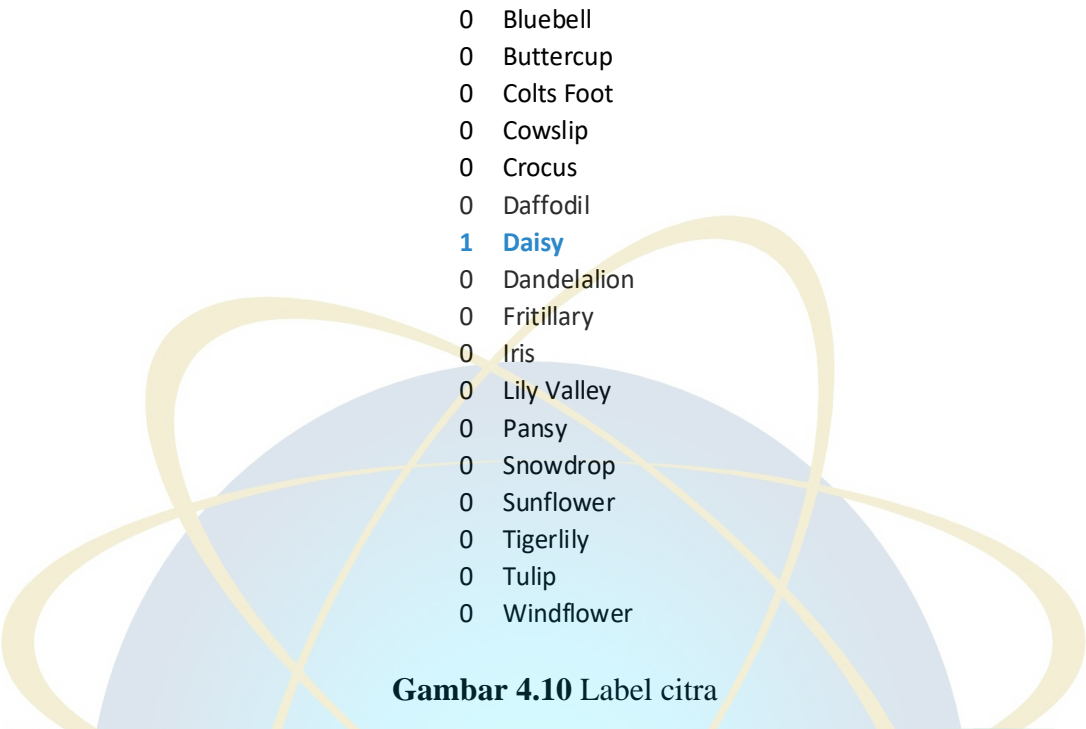
pelatihan menjadi lebih efisien. Penggunaan *fit generator* memungkinkan augmentasi data pada CPU dan *training* pada GPU dapat dilaksanakan paralel secara *real-time*.

b. Proses Pelatihan

Layer pertama dalam CNN adalah input *layer*, pada input *layer* menerima masukan citra gambar dengan tiga atribut, yaitu panjang citra (*pixel*), lebar citra (*pixel*) dan *channel* warna (RGB atau *grayscale*). Karena penelitian ini adalah *supervised learning*, maka pada input citra pelatihan juga menggunakan label. Sebagai contoh diambil salah satu data citra untuk pelatihan (Gambar 4.9) dan labelnya yaitu bunga Daisy (Gambar 4.10)



Gambar 4.9 Salah satu data citra pada input *layer*



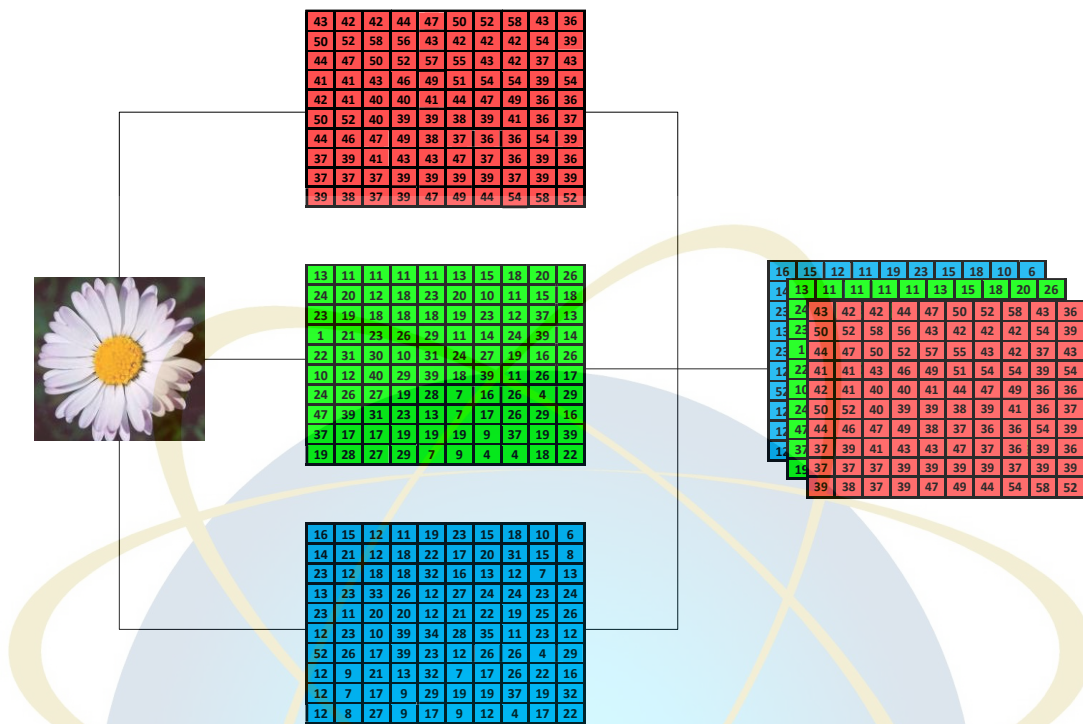
0 Bluebell
 0 Buttercup
 0 Colts Foot
 0 Cowslip
 0 Crocus
 0 Daffodil
 1 Daisy
 0 Dandelion
 0 Fritillary
 0 Iris
 0 Lily Valley
 0 Pansy
 0 Snowdrop
 0 Sunflower
 0 Tigerlily
 0 Tulip
 0 Windflower

Gambar 4.10 Label citra

Proses pelatihan terdiri sebagai berikut:

1. Input Layer

Tahapan pertama dalam melakukan pelatihan terhadap model adalah memasukkan data citra bunga ke input *layer*, pada *layer* ini citra gambar dikonversi kedalam matriks tiga dimensi, dengan ukuran panjang×lebar×3 *channel* RGB (*Red, Green, Blue*). Pada penelitian ini nilai RGB pada setiap piksel dinormalisasi menjadi rentang 0-1 untuk mempermudah proses komputasi dengan cara membagi setiap nilai RGB setiap piksel dengan 255.



Gambar 4.11 Input layer



Gambar 4.12 Channel RGB pada citra input

2. Convolutional Layer

Data besaran nilai RGB *channel* pada setiap piksel kemudian diproses di lapisan konvusi (*convolution layer*). Fungsi lapisan ini adalah untuk mengekstrasi fitur (*feature map*) yang ada pada citra dengan menggunakan *filter*. Sebagai contoh, pada Gambar 4.13 adalah input citra pada *channel* warna merah (*red*). Angka-angka

yang tersebut mewakili besaran nilai intensitas warna merah pada citra gambar tersebut (0-255).

43	42	42	44	47	50	52	58	43	36
50	52	58	56	43	42	42	42	54	39
44	47	50	52	57	55	43	42	37	43
41	41	43	46	49	51	54	54	39	54
42	41	40	40	41	44	47	49	36	36
50	52	40	39	39	38	39	41	36	37
44	46	47	49	38	37	36	36	54	39
37	39	41	43	43	47	37	36	39	36
37	37	37	39	39	39	39	37	39	39
39	38	37	39	47	49	44	54	58	52

Gambar 4.13 Input *channel* merah

Untuk mendapatkan fitur dari citra maka citra tersebut maka diperlukan proses konvolusi dengan *filter*, *filter* sendiri merupakan kumpulan nilai (*weight*) yang diinisiasi secara acak. Contoh nilai filter terdapat pada Gambar 4.14.

2	2	2	2	2
1	1	1	1	1
0	0	0	0	0
-1	-1	-1	-1	-1
-2	-2	-2	-2	-2

Gambar 4.14 Contoh *filter*

$$y_{m,n} = g \left(\sum_j \sum_k x[j,k] W[m-n, n-k] + b \right)$$

$$\begin{aligned} (W * x)_{m,n} &= 43 \times 2 + 42 \times 2 + 42 \times 2 + 44 \times 2 + 47 \times 2 + 50 \times 1 + 52 \times \\ &1 + 58 \times 1 + 56 \times 1 + 43 \times 1 + 44 \times 0 + 47 \times 0 + 50 \times 0 + 52 \times 0 + 57 \times \\ &0 + 41 \times (-1) + 41 \times (-1) + 43 \times (-1) + 46 \times (-1) + 49 \times (-1) + 42 \times \\ &(-2) + 41 \times (-2) + 40 \times (-2) + 40 \times (-2) + 41 \times (-2) = 67 \end{aligned}$$

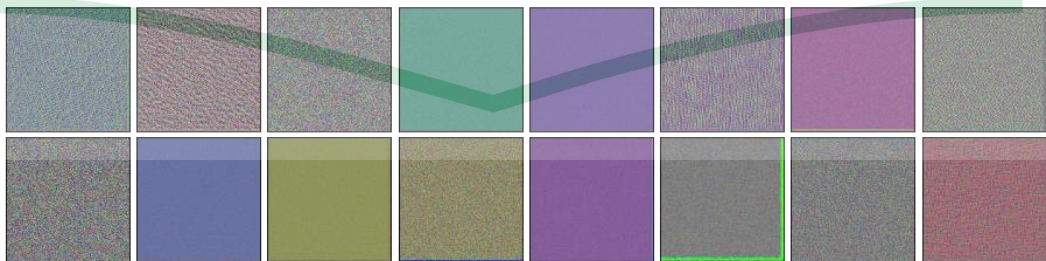
$$y_{m,n} = g(\sum_j W_j x_j + b)$$

$$y_{0,1} = \max(67 + 1, 0) = 68$$

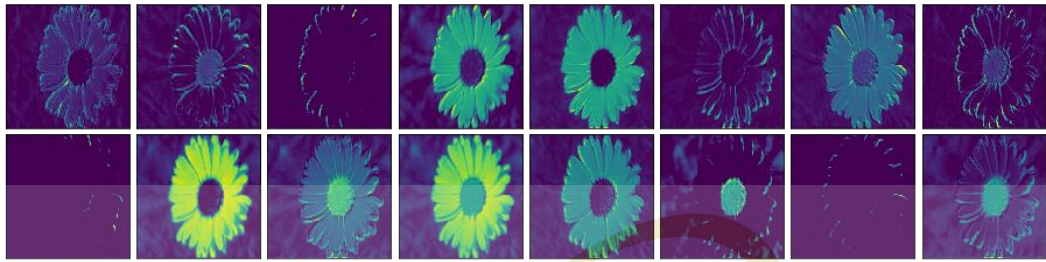
68
...
...
...
...
...

Gambar 4.15 Output proses konvolusi

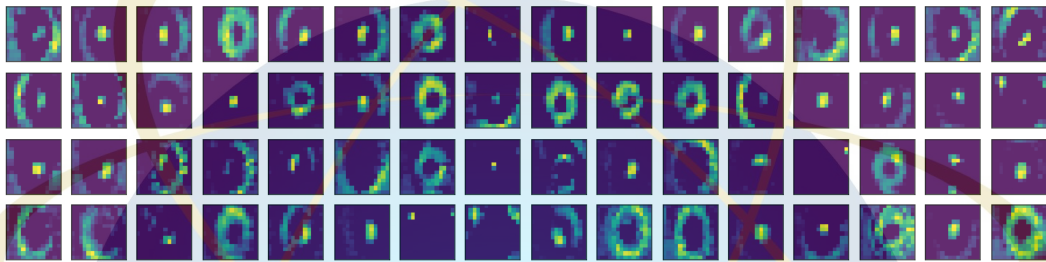
Pada lapisan konvolusional terjadi proses ekstraksi fitur untuk menghasilkan peta fitur (*feature map*) pada data citra, untuk mendapatkan mengenal pola pada setiap citra. Proses ekstraksi fitur dilakukan dengan *filter* pada lapisan konvolusional dengan besaran sesuai dengan besaran *kernel*. Visualisasi terhadap nilai *filter* pada lapisan konvolusi pertama yang digunakan pada data oxford17 terdapat pada Gambar 4.15. *Filter* tersebut kemudian dikonvolusi dengan data input sehingga menghasilkan peta fitur yang dapat mendeteksi tepian (*edge*) yang membentuk pola (Gambar 4.17), proses konvolusi dan *pooling* yang dilakukan terus menerus akan membentuk pola yang lebih detail (Gambar 4.18).



Gambar 4.16 Visualisasi filter pada CNN



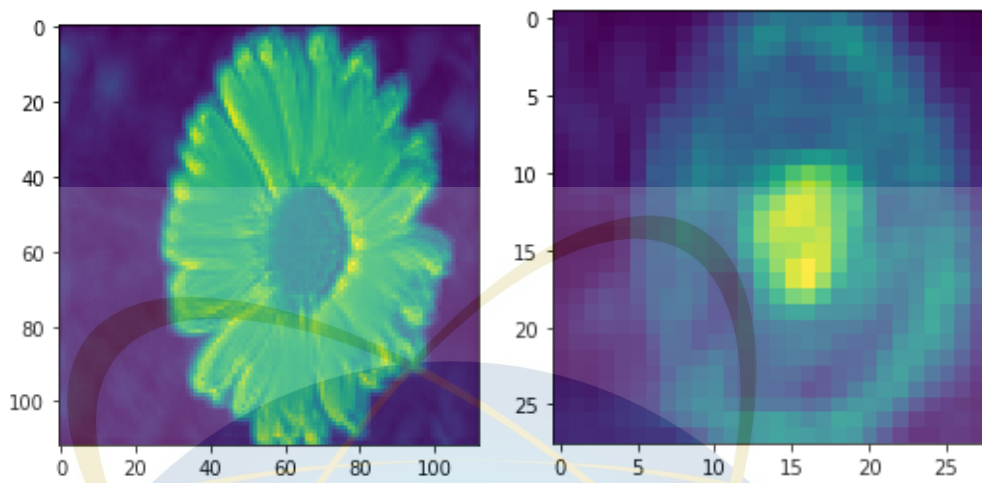
Gambar 4.17 *Feature map* pada lapisan konvolusi



Gambar 4.18 Pola pada *feature map*

3. *Activation Function*

Untuk dapat mengenal objek dalam citra diperlukan pemisahan objek dengan latar belakang pada objek. Pada penelitian ini *activation function* ReLU digunakan untuk menentukan aktif tidaknya neuron pada *neural networks* (Gambar 4.19), sehingga hanya neuron yang berhubungan dengan objek bunga saja yang dipilih (Gambar 4.20). Selain itu tujuan dari fungsi aktivasi adalah untuk menambahkan properti non-linear ke fungsi, yang merupakan *neural network*.



Gambar 4.19 Hasil dari *activation function* pada *layer 1* dan *2*



Gambar 4.20 Pemilihan area objek pada citra bunga

4. *Pooling Layer*

Pooling layer berfungsi untuk mengurangi ukuran spasial dari citra dan mengurangi jumlah parameter dan perhitungan dalam *neural network*. *Pooling layer* beroperasi pada setiap fitur secara independen. Pada lapisan ini menerima input dari hasil peta fitur (*feature map*) dari hasil proses konvolusi pada *convolution layer*. Penelitian ini menggunakan *max pooling* ukuran *pooling* 2×2 dengan *stride* 2 sehingga ukuran

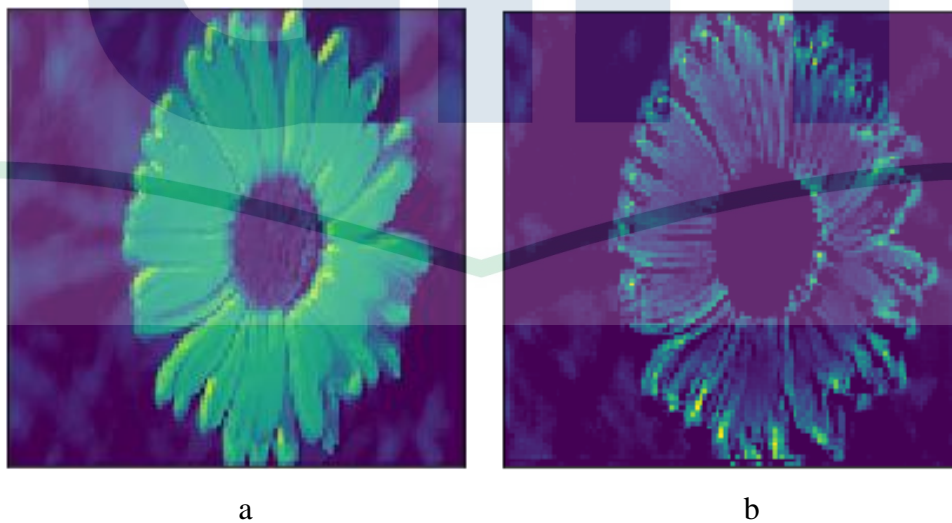
citra yang dihasilkan 50% lebih kecil. Sebagai contoh pada Gambar 4.21 merupakan input max pooling, proses ini menghasilkan *output* dengan nilai yang terbesar dari semua nilai input, pada kasus ini 67 adalah nilai terbesar sehingga diambil nilai tersebut sehingga menghasilkan nilai pada Gambar 4.22. Pada data oxford17 proses *pooling* dilakukan setelah proses konvolusi dan menggunakan *max pooling*, hasil dari proses *pooling* dapat dilihat pada Gambar 4.23.

67	42	42	44	38	40
42	41	41	38	44	43
41	42	43	41	67	38
23	12	76	21	32	12
23	23	42	75	23	45
86	23	42	12	43	23

Gambar 4.21 Input pada *pooling layer*

67	44	44
42	76	67
86	75	45

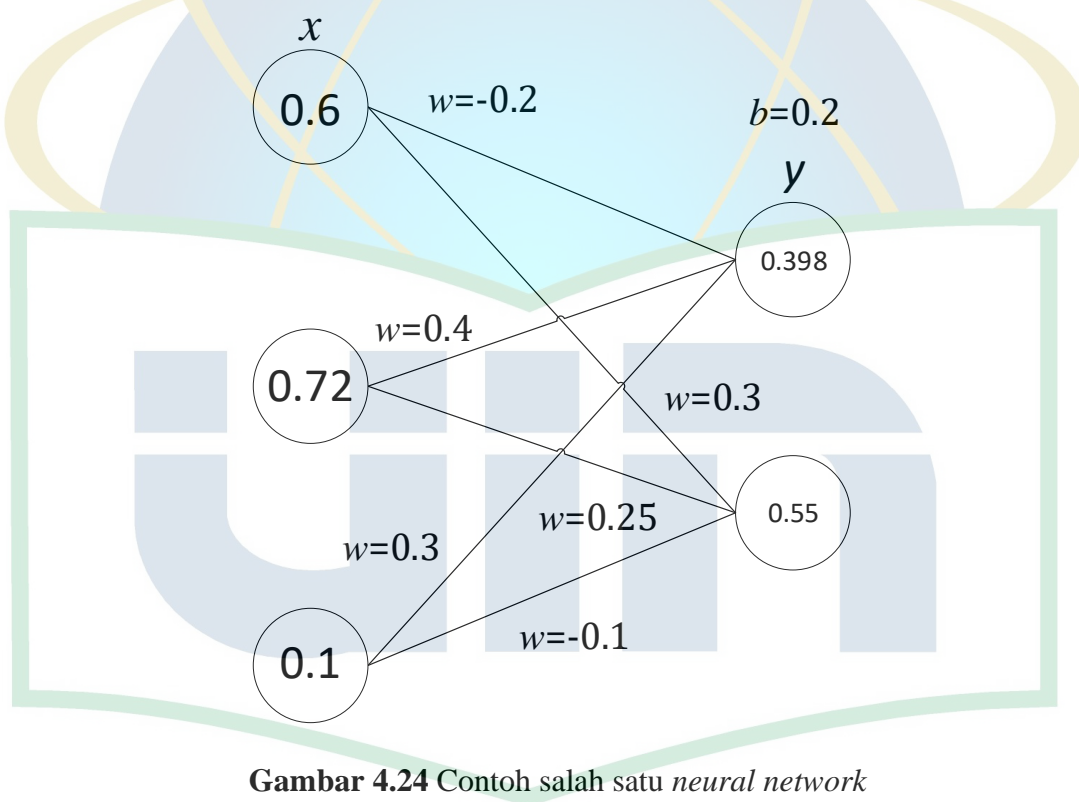
Gambar 4.22 Output pada *pooling layer*



Gambar 4.23 Input *pooling layer* (a), output *pooling layer* (b)

5. Fully Connected Layer

Tahap selanjutnya dari CNN adalah *fully connected layer*, tahapan awal pada *fully connected layer* adalah mengubah data matriks 3 dimensi pada tahap konvolusi menjadi satu dimensi vektor (*flatten*). Nilai yang ada pada neuron (x) dikalkulasi dengan bobot (w) dan ditambahkan dengan bias (b), yang akan menentukan neuron selanjutnya (y). Gambaran keseluruhan neural network pada model oxford 17 dapat dilihat pada Gambar 4.24.



Gambar 4.24 Contoh salah satu *neural network*

Berikut adalah proses untuk mendapatkan nilai pada *neuron* tujuan (y) dengan menggunakan persamaan 2.1.

$$y = g(0.6 \times (-0.2) + 0.72 \times 0.4 + 0.1 \times 0.3 + 0.2) = 0.398$$

$$y = g(0.6 \times 0.3 + 0.72 \times 0.25 + 0.1 \times (-0.1) + 0.2) = 0.55$$

Tahapan terakhir dari *fully connected* layer adalah *softmax* yang berfungsi untuk menghasilkan probabilitas dari prediksi klasifikasi atau disebut tahapan klasifikasi.

Tahapan ini menggunakan nilai dari *neuron* sebelumnya lalu mengplikasikan fungsi aktivasi *softmax* dengan menggunakan Persamaan 2.2.

Table 4.1 Neuron terakhir pada pelatihan

e	s	label		
6.2711093e-05	0.05345823	0	0	Bluebell
2.3693376e-05	0.05345614	1	0	Buttercup
1.5455488e-06	0.05345496	2	0	Colts Foot
1.1844796e-05	0.05345551	3	0	Cowslip
1.7310450e-03	0.05354749	4	0	Crocus
2.6060341e-04	0.05346881	5	0	Daffodil
9.9361295e-01	0.1443803	6	1	Daisy
1.4375035e-05	0.05345564	7	0	Dandelalion
7.3801125e-06	0.05345564	8	0	Fritillary
5.2712415e-04	0.05345564	9	0	Iris
1.4992844e-05	0.05345564	10	0	Lily Valley
1.6319246e-03	0.05345564	11	0	Pansy
4.1258110e-05	0.05345564	12	0	Snowdrop
4.4960607e-06	0.05345564	13	0	Sunflower
1.4411211e-06	0.05345564	14	0	Tigerlily
5.2668565e-05	0.05345564	15	0	Tulip
1.9999850e-03	0.05345564	16	0	Windflower

6. Loss function

Loss untuk mengukur kinerja model diperlukan *loss function*, *loss function* mengukur perbedaan antara prediksi dan aktual. Nilai prediksi didapat dari hasil *softmax* dan aktual didapat dari urutan label atau indeks label. Berikut ini adalah proses menghitung *loss* pada satu citra yang diinput di atas. Bunga Daisy berada pada indeks ke-7, maka hanya nilai ke-7 adalah 1, dan kelas/jenis lain bernilai 0.

Aktual = [0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]

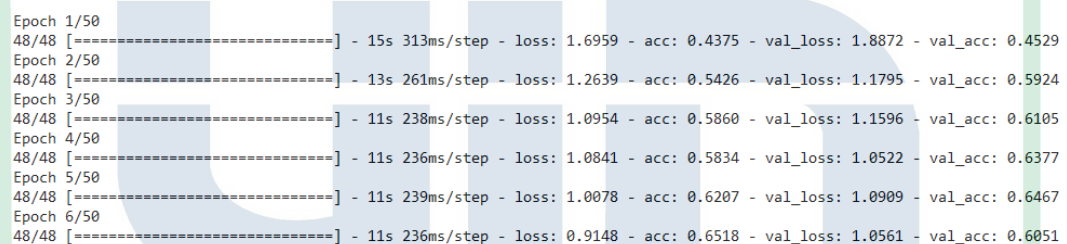
Prediksi = [0.05345823, 0.05345614, 0.05345496, 0.05345551, 0.05354749, 0.05346881, 0.1443803 , 0.05345564, 0.05345527, 0.05348306, 0.05345568, 0.05354218, 0.05345708, 0.05345512, 0.05345495, 0.05345769, 0.05356189]

Loss = 0.1655677436840361

7. Backpropagation

Tahap terakhir dari proses pelatihan *neural network* dalam satu iterasi (*epoch*) adalah *backpropagation*, yaitu proses untuk memperbaharui bobot dan bias untuk mengurangi *loss* secara keseluruhan pelatihan dengan metode *stochastic gradient descent* yang mana dalam penelitian ini menggunakan Nadam. Semakin kecil *loss* maka akan semakin baik model dan semakin baik tingkat akurasinya, seperti pada

Gambar 4.25 dimana *loss* akan semakin kecil setiap *epoch*nya, dan akurasi memiliki nilai yang tinggi jika *loss* bernilai rendah.



Epoch	Progress	Time	Step	loss	acc	val_loss	val_acc
Epoch 1/50	48/48	15s	313ms/step	1.6959	0.4375	1.8872	0.4529
Epoch 2/50	48/48	13s	261ms/step	1.2639	0.5426	1.1795	0.5924
Epoch 3/50	48/48	11s	238ms/step	1.0954	0.5860	1.1596	0.6105
Epoch 4/50	48/48	11s	236ms/step	1.0841	0.5834	1.0522	0.6377
Epoch 5/50	48/48	11s	239ms/step	1.0078	0.6207	1.0909	0.6467
Epoch 6/50	48/48	11s	236ms/step	0.9148	0.6518	1.0561	0.6051

Gambar 4.25 Loss dan akurasi pada pelatihan

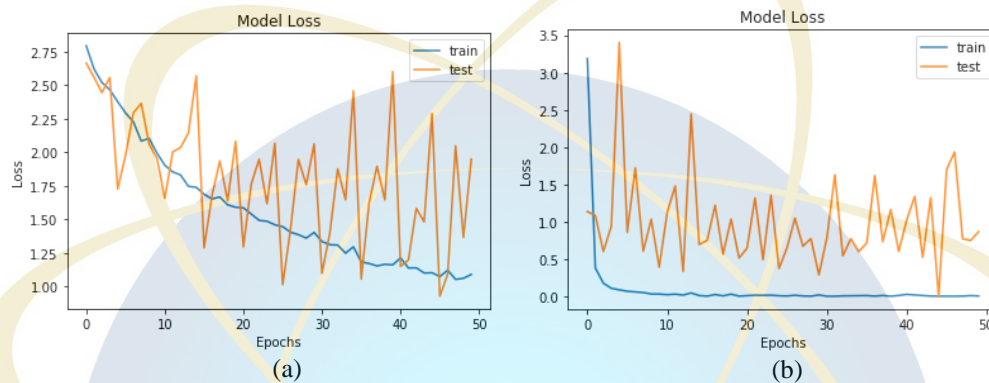
4.2.4 Pengujian Model

a. Model Loss

1. Oxford17

Hasil yang didapatkan untuk proses pelatihan terhadap *dataset* oxford17 menghasilkan nilai *loss* pada validasi yang fluktuatif, dengan nilai *loss* tertinggi 1.1 pada nilai *loss training* hingga 0.7 pada nilai *loss* validasi. Sedangkan dengan

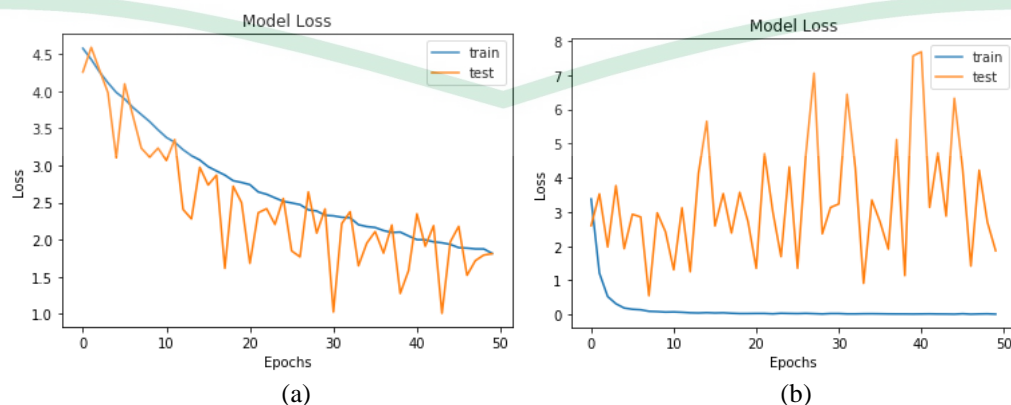
menggunakan metode *transfer learning* menghasilkan nilai 3.2 pada *loss* tertinggi dan 0.09 pada nilai *loss* terendah pada data validasi. Pada data *training* nilai *loss* terbesar berada pada 3.2 dan terkecil 0.05. Semakin kecil nilai *loss* maka semakin baik kinerja model yang didapatkan.



Gambar 4.26 Model *loss* untuk dataset oxford17 dengan CNN (a) dan CNN *transfer learning* (b)

2. Oxford102

Pada dataset oxford102 mendapatkan kinerja lebih buruk daripada data pada oxford17, nilai *loss* terendah yang didapat pada *training* data berada pada 1.8 dan nilai *loss* pada data validasi/test sebesar 1.1. Dengan metode *transfer learning* nilai *loss* yang berhasil didapatkan 0.02 pada data *training* dan 0.4 pada data validasi.

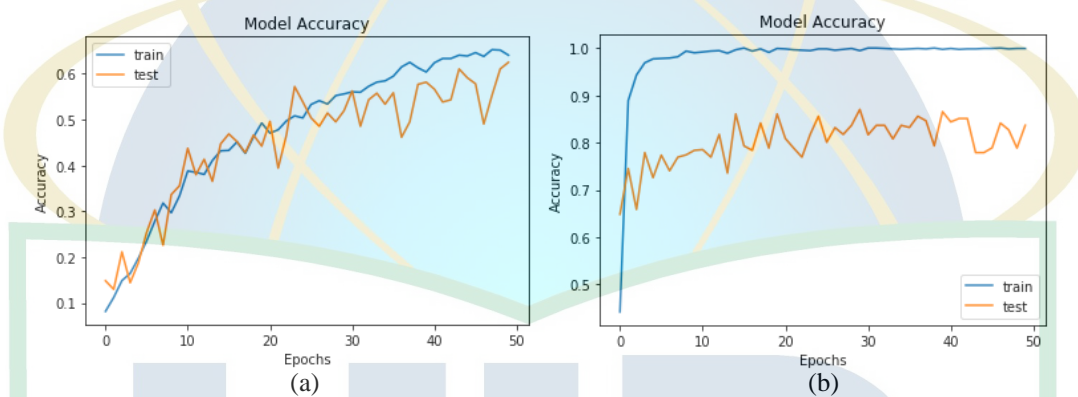


Gambar 4.27 Model *loss* untuk dataset oxford102 dengan CNN (a) dan CNN *transfer learning* (b)

b. Model Akurasi

1. Oxford17

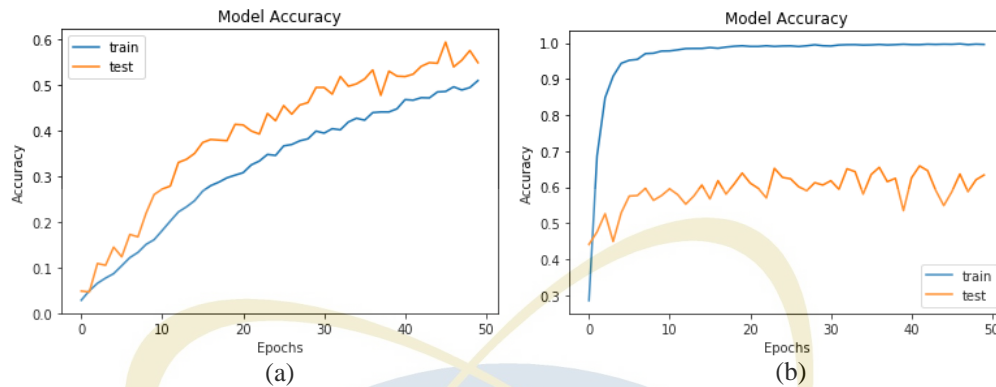
Untuk nilai akurasi untuk *training* terhadap *dataset* oxford17 mendapatkan hasil rata-rata 0.54 untuk hasil pada validasi dan 0.65 pada hasil *training*. Sedangkan untuk pelatihan *dataset* oxford17 menggunakan *transfer learning* dari ResNet50 dapat menghasilkan 0.98 akurasi pada *training* dan 0.83 akurasi pada validasi.



Gambar 4.28 Model akurasi untuk *dataset* oxford17 dengan CNN (a) dan CNN *transfer learning* (b)

2. Oxford102

Akurasi pada *dataset* oxford102 mendapatkan hasil akurasi tertinggi 0.58 untuk hasil pada validasi dan 0.47 pada hasil *training*. Pelatihan dengan *transfer learning* ResNet50 dapat menghasilkan akurasi tertinggi 0.98 pada *training* dan 0.6 akurasi pada validasi.



Gambar 4.29 Model akurasi untuk *dataset* oxford102 dengan CNN (a) dan CNN *transfer learning* (b)

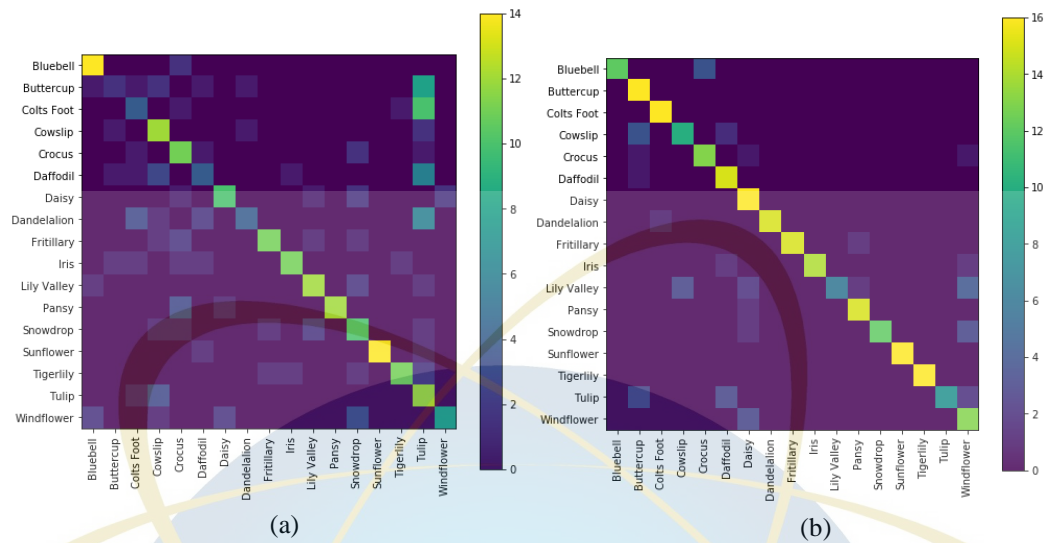
c. *Confusion matrix*

1. Oxford17

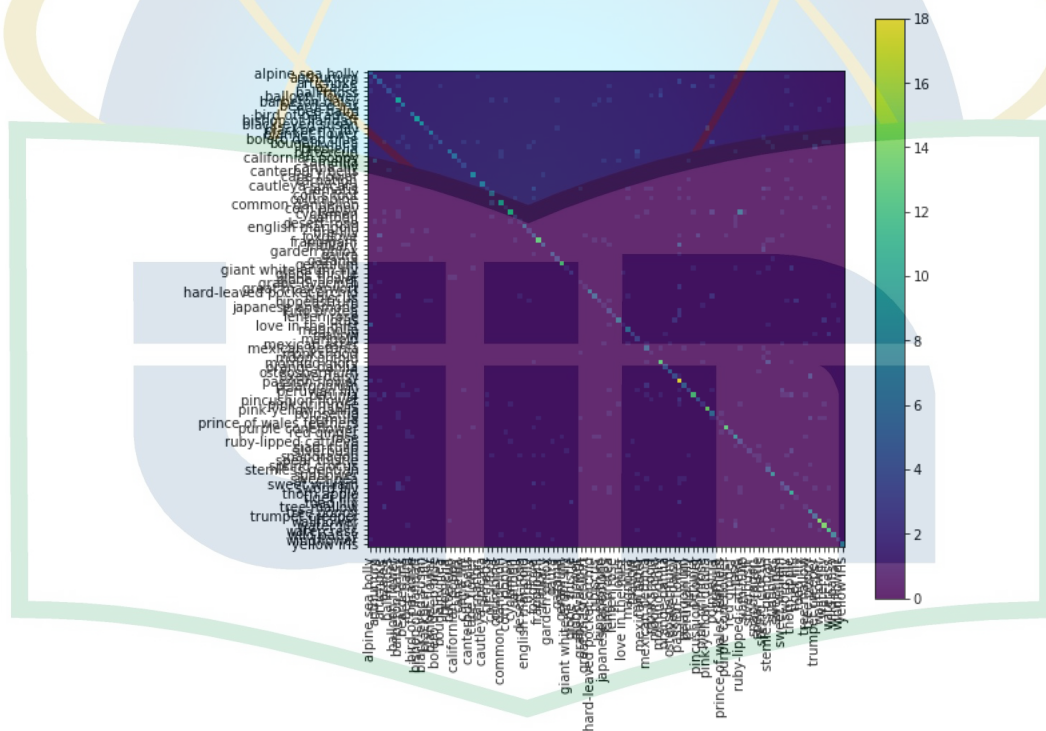
Hasil menggunakan dengan CNN (Gambar 4.30) tanpa *transfer learning* menunjukkan terdapat kekeliruan pada klasifikasi bunga tulip sehingga sering dikenali sebagai bunga *buttercup* dan bunga *coltsfoot*. Sedangkan *confusion matrix* pada pelatihan dengan *transfer learning* menunjukkan hasil yang baik.

2. Oxford102

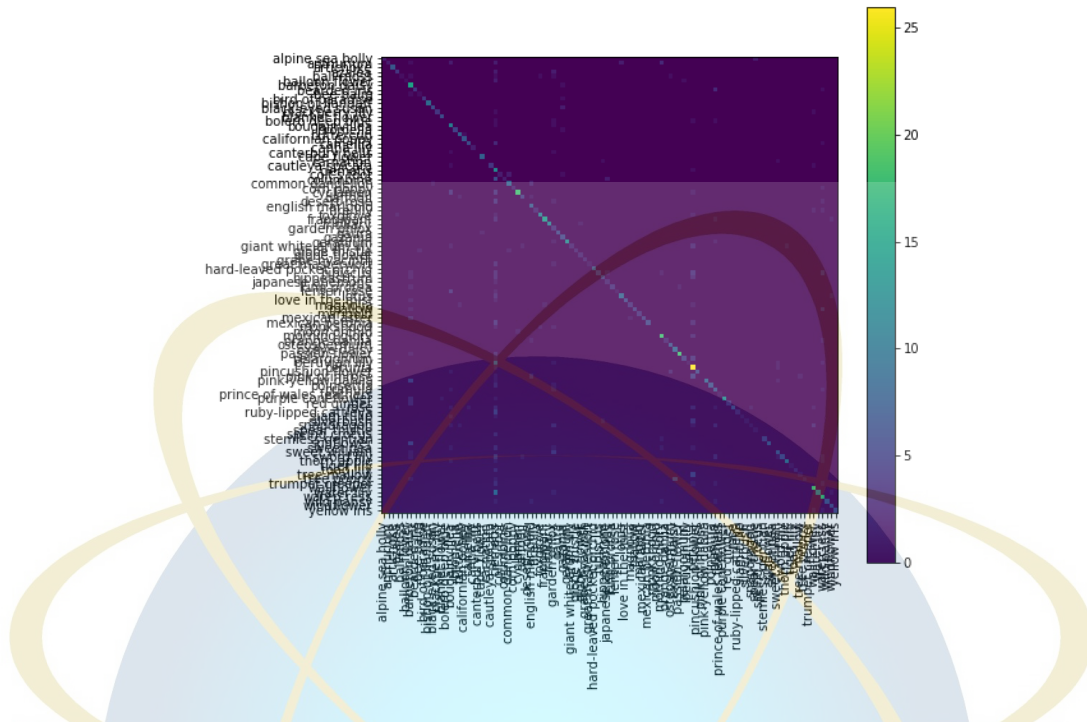
Confusion matrix pada dataset oxford 102 untuk dengan dan tanpa *transfer learning* menunjukkan kinerja yang cukup baik (terdapat garis diagonal) (Gambar 4.31 dan Gambar 4.32). Meskipun pada pelatihan dengan *transfer learning* terdapat kekeliruan pada klasifikasi terhadap bunga *thorn apple*.



Gambar 4.30 *Confusion matrix* oxford17 (a) dan dengan *transfer learning* (b)



Gambar 4.31 *Confusion matrix* oxford102



Gambar 4.32 *Confusion matrix oxford102 transfer learning*

c. Skor hasil klasifikasi

Dalam mengukur kinerja kinerja model dalam penelitian ini menggunakan *Precision*, *Recall* & *f1-Score* secara rata-rata. Dari hasil penilaian, pelatihan dengan metode *transfer learning* memiliki kinerja model yang lebih baik dalam melakukan klasifikasi.

1. Oxford17

Tabel 4.1 Perbandingan skor klasifikasi CNN

	<i>Precision</i>	<i>Recall</i>	<i>f1-score</i>
CNN	0.66	0.6	0.60
CNN dengan <i>transfer learning</i>	0.88	0.84	0.83

2. Oxford102

Tabel 4.2 Perbandingan skor klasifikasi CNN

	<i>Precision</i>	<i>Recall</i>	<i>f1-score</i>
CNN	0.58	0.54	0.52
CNN dengan <i>transfer learning</i>	0.79	0.64	0.64

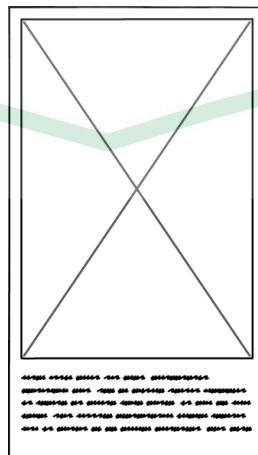
4.3 Aplikasi Klasifikasi Bunga

4.3.1 Perancangan *User Interface*

Pengujian terhadap model CNN *deep learning* dilakukan pada aplikasi android. Platform Android dipilih untuk memudahkan dalam mendapatkan citra gambar, karena pada android dapat langsung menggunakan kamera untuk mendapatkan citra bunga.

a. Tampilan klasifikasi

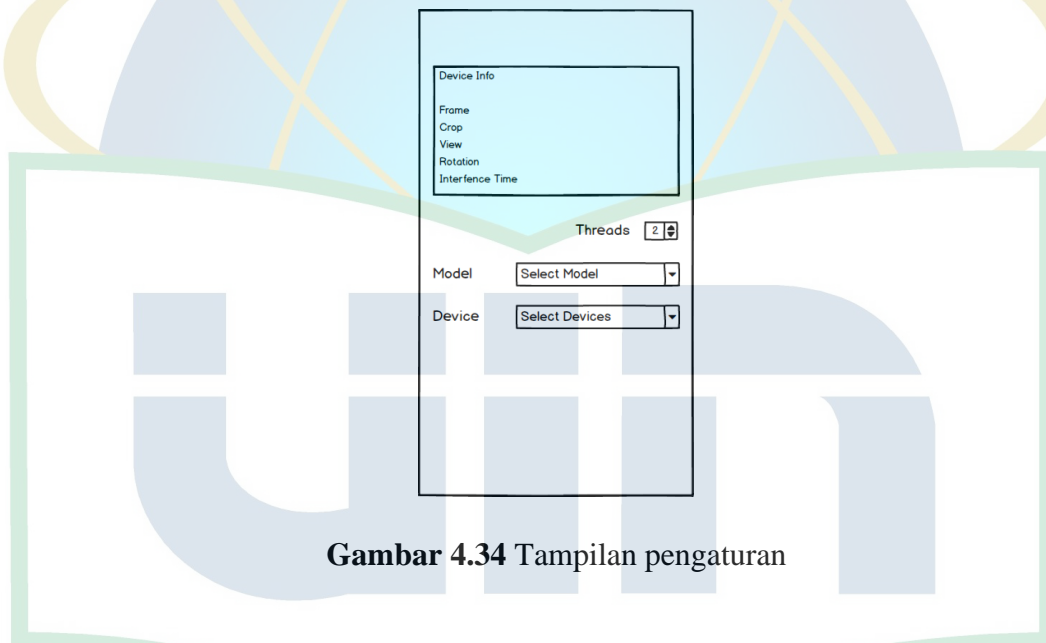
Pada tampilan klasifikasi untuk citra yang berasal dari kamera dilakukan proses pengklasifikasian dari model, hasil probabilitas klasifikasi dipilih tiga yang terbesar dan ditampilkan pada bagian bawah aplikasi.



Gambar 4.33 Tampilan klasifikasi

b. Tampilan Pengaturan

Pengaturan berfungsi untuk memilih model klasifikasi yang akan digunakan, dalam penelitian ini empat model yang dapat digunakan untuk melakukan klasifikasi, yaitu oxford17, oxford17 resnet, oxford102, dan oxford102 resnet. Pada pengaturan ini juga dapat memilih dimana pemrosesan dilakukan, terdapat dua pilihan CPU dan GPU dan *threads* yang digunakan. Halaman pengaturan ini juga terdapat info mengenai media input klasifikasi.



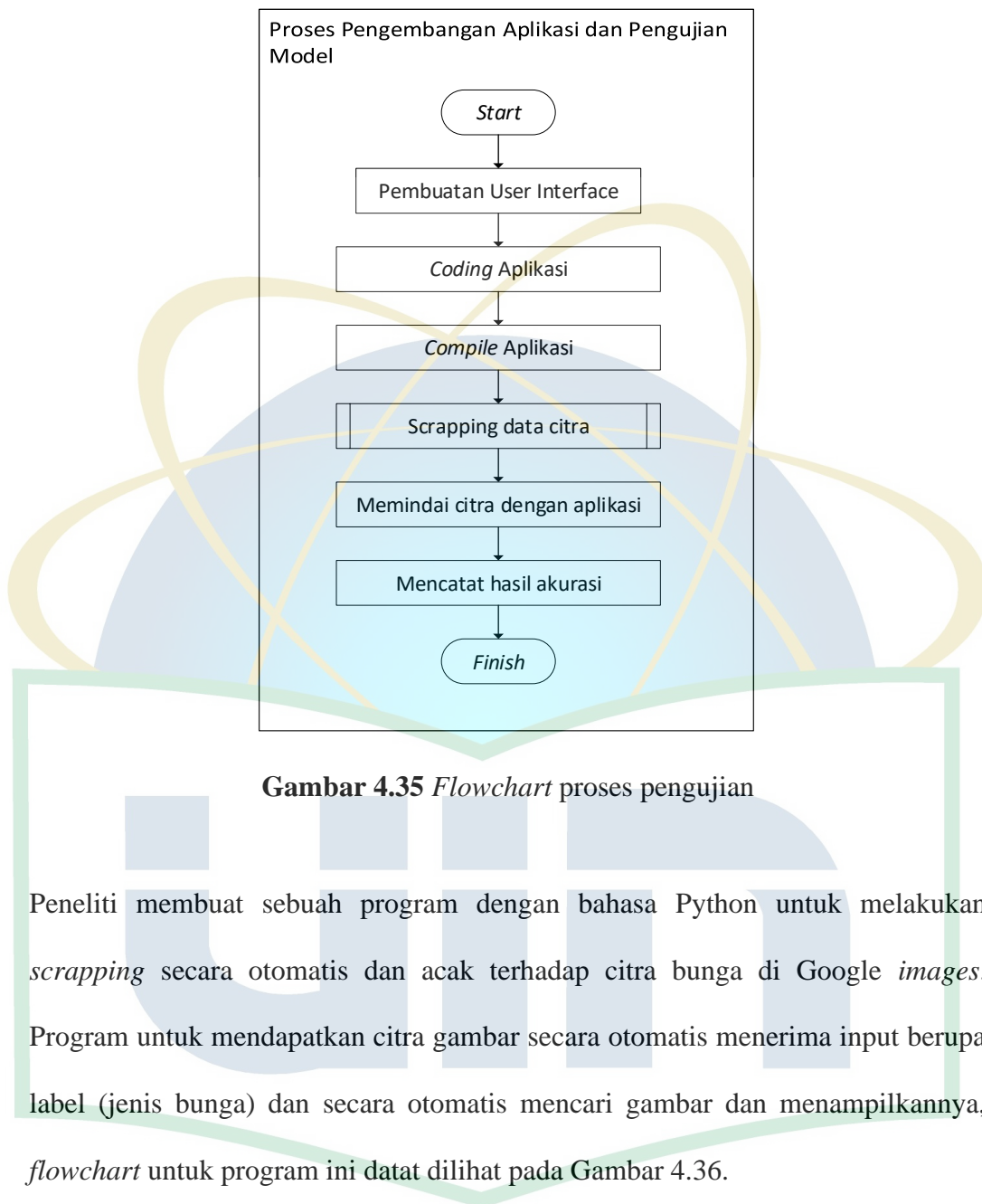
Gambar 4.34 Tampilan pengaturan

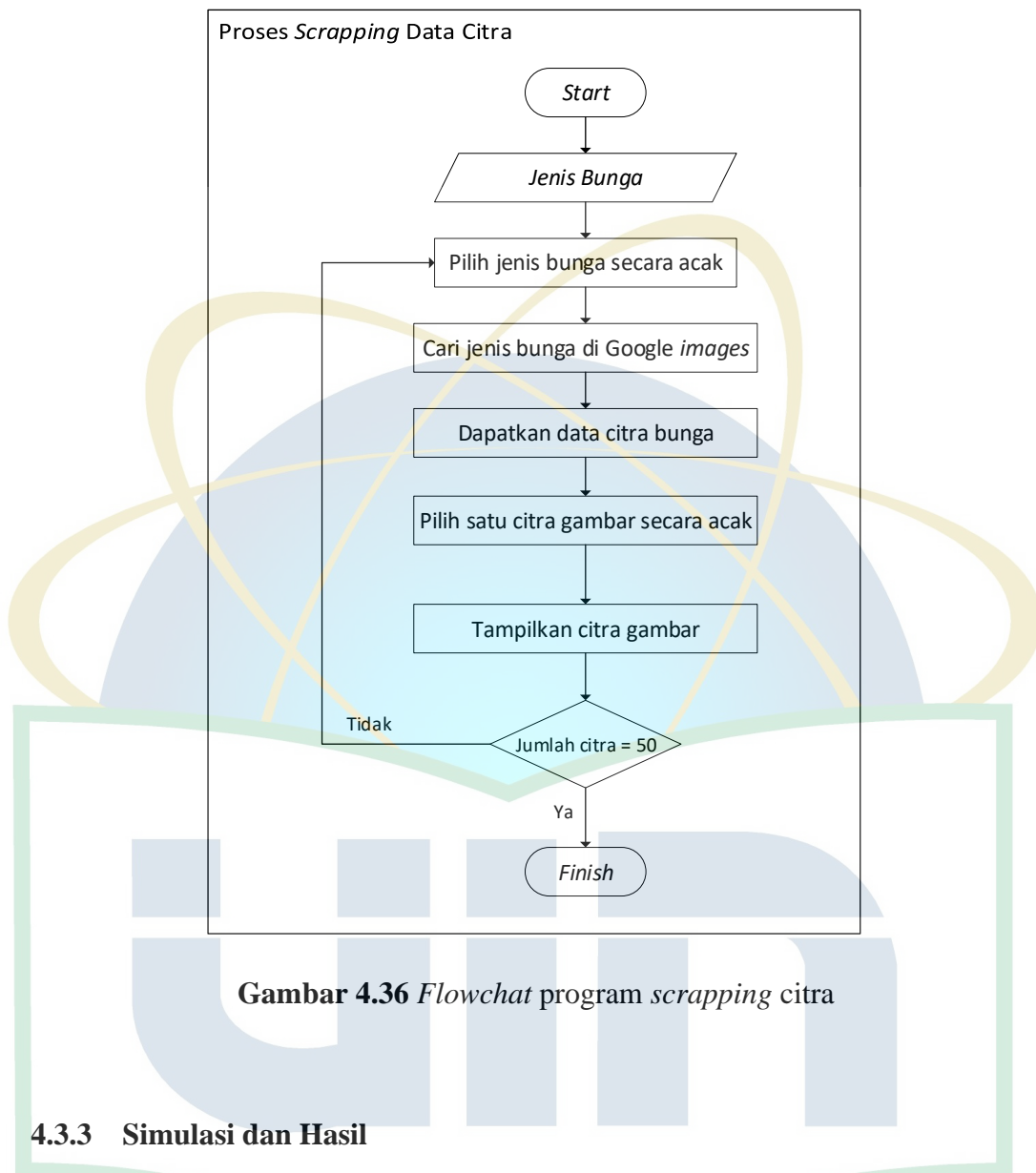
4.3.2 Tahapan Pengembangan Aplikasi dan Pengujian

Peneliti mengembangkan aplikasi untuk menguji kinerja model CNN. Proses pengembangan aplikasi dapat dilihat pada Gambar 4.35. Proses pengembangan aplikasi serta pengujian terhadap model terdiri sebagai berikut:

a. Peneliti membuat *user interface* aplikasi

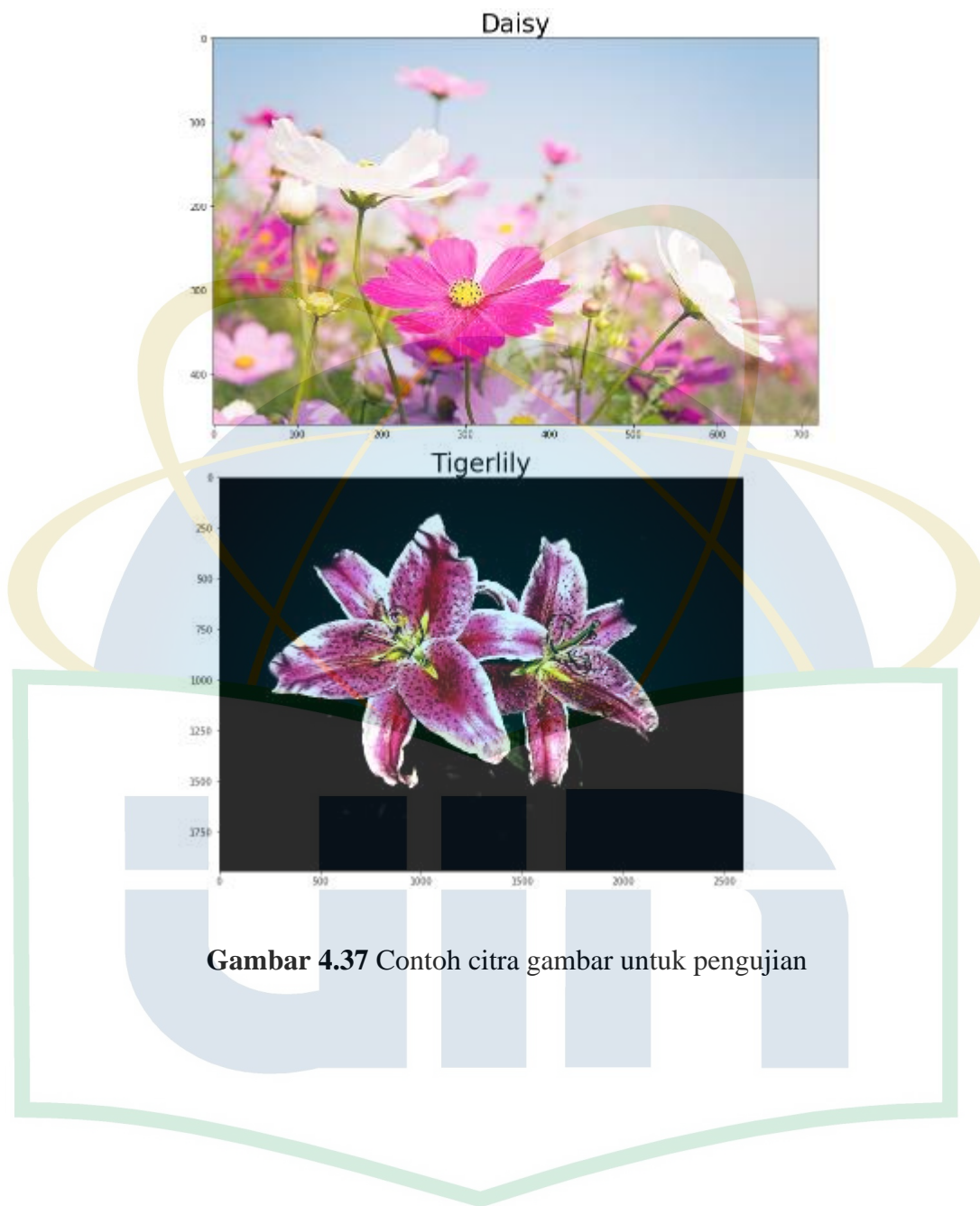
- b. Peneliti membuat aplikasi di Android dengan Android Studio yang dapat memuat model klasifikasi dan menggunakan model tersebut untuk melakukan tugas klasifikasi terhadap citra bunga.
- c. Selanjutnya peneliti meng-*compile* aplikasi dan memasangnya pada *smartphone*.
- d. Peneliti membuat program *scrapping* dengan Python untuk mendapatkan data citra bunga secara acak.
- e. Setelah citra bunga didapatkan, peneliti melakukan pengambilan citra gambar dengan kamera *smartphone* dengan aplikasi klasifikasi.
- f. Proses pengujian dilakukan terhadap 50 citra bunga untuk masing-masing model dengan total 4 model, sehingga menjadi total 200 citra.
- g. Peneliti menghitung akurasi dari hasil pengujian untuk mengukur kinerja dari model klasifikasi.

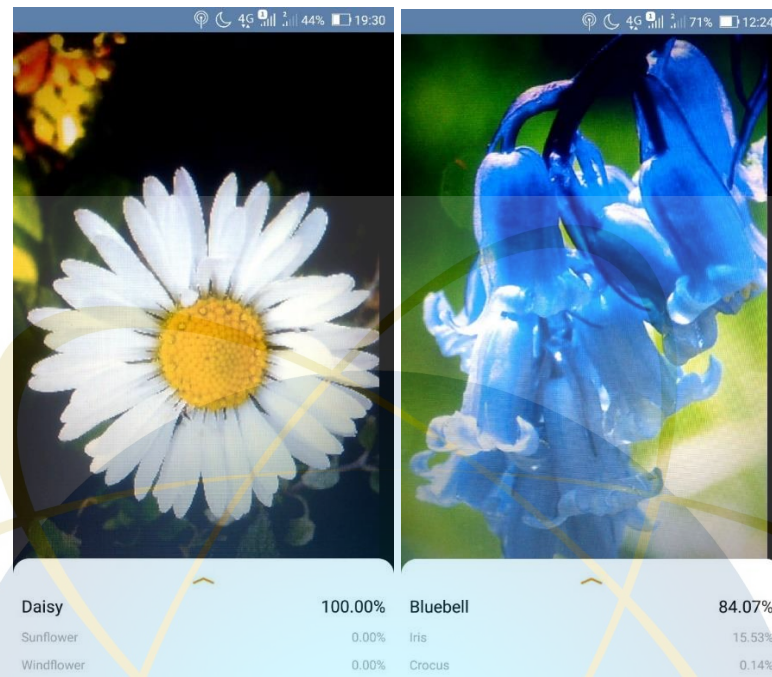




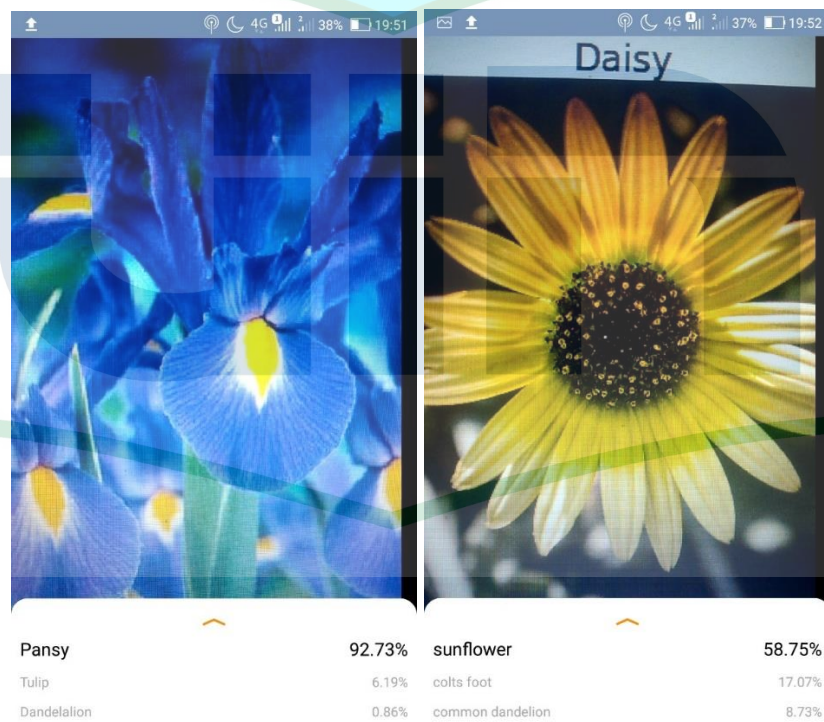
4.3.3 Simulasi dan Hasil

Peneliti menjalankan program pada laptop untuk mendapatkan data uji sebanyak 50 citra untuk tiap model (Gambar 4.37). Setelah mendapatkan data citra peneliti melakukan pengklasifikasian dengan *scanning* menggunakan kamera *smartphone* pada citra gambar yang ada pada layar *laptop*. Peneliti mencatat setiap prediksi yang benar (Gambar 4.38) dan prediksi yang salah (Gambar 4.39), kemudian hasil tersebut dikumpulkan dan digunakan untuk menghitung nilai akurasi pada setiap model.





Gambar 4.38 Pengujian dengan prediksi benar



Gambar 4.39 Pengujian dengan prediksi salah

a. Oxford17

Tabel 4.3 Hasil pengujian model untuk oxford17

Prediksi	
Benar	30
Salah	20

$$\text{Akurasi} = \frac{30}{50} = 0.6$$

Akurasi model pada oxford17 sebesar 60%.

b. Oxford17 *Transfer Learning*

Tabel 4.4 Hasil pengujian model untuk oxford17 transfer learning

Prediksi	
Benar	42
Salah	8

$$\text{Akurasi} = \frac{42}{50} = 0.84$$

Akurasi model pada oxford17 *transfer learning* sebesar 84%.

c. Oxford102

Tabel 4.5 Hasil pengujian model untuk oxford102

Prediksi	
Benar	21
Salah	29

$$\text{Akurasi} = \frac{21}{50} = 0.42$$

Akurasi model pada oxford102 sebesar 42%.

d. Oxford102 Transfer Learning

Tabel 4.6 Hasil pengujian model untuk oxford102 *transfer learning*

Prediksi	
Benar	32
Salah	18

$$\text{Akurasi} = \frac{32}{50} = 0.64$$

Akurasi model pada oxford102 *transfer learning* sebesar 64%.

4.4 Interpretasi dan Evaluasi

Berdasarkan dari proses pengujian *Convolutional Neural Network* untuk klasifikasi jenis bunga dengan total 119 jenis bunga didapatkan hasil akurasi sebagai nilai untuk perbandingan dengan metode *Support Vector Machine* (SVM) dan *Artificial Neural Network* (ANN) pada Tabel 4.7. *Convolutional Neural Network* dengan menggunakan metode *transfer learning* lebih unggul dengan nilai akurasi 84% jika dibandingkan dengan model SVM pada penelitian Albadarneh & Ahmad (2017) yang memiliki nilai akurasi 83.52% dan model ANN mendapatkan nilai akurasi 72% yang dibuat Firmansyah (2020), akan tetapi model CNN *from the scratch* tidak dapat mengungguli ANN dan SVM karena hanya memiliki nilai akurasi 60%. Data *preprocessing* terhadap citra bunga pada metode SVM menggunakan teknik *region growing segmentation*, yaitu memisahkan objek bunga dari latar belakangnya.

Sedangkan bila dibandingkan dengan ANN dan SVM, model CNN yang menggunakan metode *from the scratch*, memiliki nilai akurasi jauh di bawah model

ANN dalam penelitian Almogdady et al. (2018) dengan nilai akurasi 81.19%, tetapi unggul dari model SVM yang pengujian hanya mendapatkan nilai akurasi 32.4%, untuk *dataset* Oxford102 model CNN yang menggunakan *transfer learning* mendapatkan nilai akurasi sebesar 64% sedangkan model CNN yang tidak menggunakan metode *transfer learning* mendapatkan nilai akurasi 42%, sangat jauh di bawah nilai akurasi ANN dan SVM yang mencapai 81.19% dan 80%. ANN pada penelitian tersebut menggunakan HSV *color descriptor*, Gray Level Co-occurrence Matrix (GLCM) dan Invariant Moments pada tahapan *data preprocessing* untuk memisahkan objek bunga dengan latar belakangnya.

Model CNN yang dilatih dengan metode *transfer learning* memiliki keunggulan dalam hal akurasi untuk model yang dilatih secara singkat. *Transfer learning* dapat menghasilkan model dengan akurasi yang tinggi dengan waktu yang singkat (hanya dengan beberapa *epoch*) dibandingkan dengan model CNN yang dilatih dari dasar dengan jumlah *epoch* yang sama. Selain itu, semakin banyak jumlah jenis atau kelas maka hasil klasifikasi yang diperoleh semakin tidak baik atau buruk, hal ini dibuktikan dengan kasus klasifikasi oxford17 yang memiliki 17 jenis bunga memiliki nilai akurasi yang lebih baik (60%) dibandingkan dengan klasifikasi oxford102 yang memiliki jenis bunga sebanyak 102 jenis (42%), keduanya menggunakan model yang sama.

Tabel 4.7 Hasil perbandingan metode CNN dengan SVM dan ANN

Dataset	Metode			
	Firmansyah (2020)		SVM	ANN
	CNN	CNN <i>Transfer Learning</i>		
Oxford17	60%	84%	83.52% Albadarneh & Ahmad (2017)	72% Firmansyah (2020)
Oxford102	42%	64%	32.4% Firmansyah (2020)	81.19% Almogdady et al. (2018)

BAB 5

PENUTUP

5.1 Kesimpulan

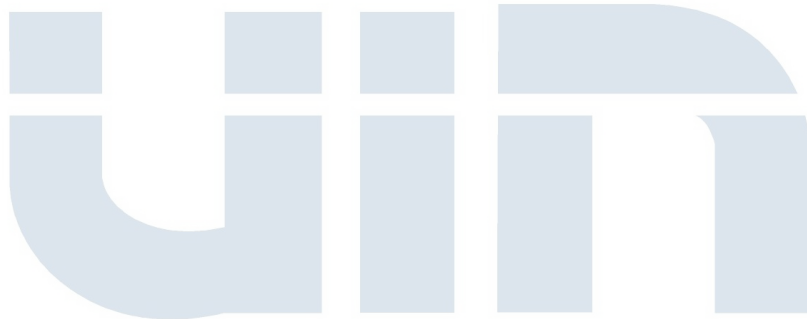
Berdasarkan hasil-hasil pembuatan dan pengujian model *deep learning* dengan *convolutional neural network* pada penelitian ini, maka dapat ditarik kesimpulan sebagai berikut:

1. Penelitian ini menghasilkan empat model *deep learning* dengan *convolutional neural network* untuk mengklasikasikan jenis model. Model ini untuk menghasilkan prediksi klasifikasi jenis bunga dari dua dataset yang berbeda yaitu oxford17 dan oxford102 dengan dua pendekatan yang berbeda, yaitu *from the strach* dan *transfer learning*.
2. Penelitian ini menghasilkan aplikasi pengujian untuk mengklasifikasikan bunga berdasarkan model. Pengujian dilakukan dengan menscan 50 data bunga dengan aplikasi dan mencatat hasil akurasi. Dari hasil pengujian didapatkan bahwa kinerja model CNN ini untuk mengklasifikasikan bunga pada oxford17 mendapatkan akurasi 60% dan 84% dengan menggunakan pendekatan *transfer learning*. Sedangkan untuk bunga pada oxford102 mendapatkan hasil akurasi sebesar 42% dan 64% untuk akurasi dengan *transfer learning*. Berdasarkan hasil penelitian terhadap *dataset* oxford17, CNN dapat mengungguli SVM, dimana model

CNN dengan *transfer learning* mendapatkan hasil akurasi 84% sedangkan SVM mendapatkan akurasi 83.52% dan ANN dengan nilai akurasi 72%. Sedangkan untuk *dataset* oxford 102, akurasi CNN jauh di bawahnya, yakni CNN dengan *transfer learning* dengan akurasi 64% sedangkan ANN berada diatasnya dengan akurasi 81.19%, namun akurasi CNN dapat mengungguli akurasi SVM dengan nilai 32.4%.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, maka peneliti memberikan saran untuk penelitian selanjutnya, agar dapat membandingkan metode *Convolutional Neural Network* dengan metode lainnya seperti *Deep Reinforcement Learning*, *Generative Adversarial Network (GAN)* ataupun *Random Forest*.



DAFTAR PUSTAKA

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: A system for large-scale machine learning. *Symposium on Operating Systems Design and Implementation* (pp. 265–283).
- Abdelhamid, N., & Thabtah, F. (2014). Associative Classification Approaches: Review and Comparison. *Journal of Information & Knowledge Management*, 13(03), 1–30.
- Albadarneh, A., & Ahmad, A. (2017). Automated Flower Species Detection and Recognition from Digital Images. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(4), 144.
- Ali, Z., Shahzad, S. K., & Shahzad, W. (2017). Performance Analysis of Statistical Pattern Recognition Methods in KEEL. *Procedia Computer Science*, 112, 2022–2030.
- Almogdady, H., Manaseer, S., & Hiary, H. (2018). A Flower Recognition System Based On Image Processing And Neural Networks. *International Journal of Scientific and Technology Research*, 7, 166–173.
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., et al. (2018). The history began from AlexNet: a comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 1–39.
- Amorim, W. P., Tetila, E. C., Pistori, H., & Papa, J. P. (2019). Semi-supervised

learning with convolutional neural networks for UAV images automatic recognition. *Computers and Electronics in Agriculture*, 164, 1–9.

Bingol, O. R., & Krishnamurthy, A. (2019). NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9, 85–94.

Borji, A. (2018). Negative results in computer vision: A perspective. *Image and Vision Computing*, 69, 1–8.

Brownlee, J. (2016). What is Deep Learning? *Deep Learning*. Retrieved March 28, 2019, from <https://machinelearningmastery.com/what-is-deep-learning/>

Bunker, R. P., & Thabtah, F. (2019). A machine learning framework for sport result prediction. *Applied Computing and Informatics*, 15(1), 27–33.

Chai, Y., Lempitsky, V., & Zisserman, A. (2011). BiCoS: A Bi-level Co-Segmentation Method for Image Classification. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2579–2586).

Christenhusz, M. J. M., & James, B. (2016). The number of known plants species in the world and its annual increase. *Phytotaxa*, 261(3), 201–217.

Cilimkovic, M. (2015). Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15, 1–12.

Danukusumo, K. P. (2017). *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Citra Candi Berbasis GPU*. Fakultas Teknologi Industri Universitas Atma Jaya.

Dogo, E., Afolabi, O., Nwulu, N., Twala, B., & Aigbavboa, C. (2018). A

Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)* (pp. 92–99).

Dozat, T. (2016). Incorporating nesterov momentum into adam. *ICLR*, 1–4.

Gollapudi, S. (2019). Object Detection and Recognition. *Learn Computer Vision Using OpenCV*. Apress. Retrieved from https://doi.org/10.1007/978-1-4842-4261-2_5

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Grewal, P. (2014). A Critical Conceptual Analysis of Definitions of Artificial Intelligence as Applicable to Computer Engineering. *IOSR Journal of Computer Engineering*, 16, 9–13.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, Y.-L., Zhang, X.-L., Ao, W., & Huang, J. Z. (2018). Determining the optimal temperature parameter for Softmax function in reinforcement learning. *Applied Soft Computing*, 70, 80–85.

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., et al. (2017). CNN architectures for large-scale audio classification. *IEEE international conference on acoustics, speech and signal*

processing (icassp) (pp. 131–135).

James, S. C., Zhang, Y., & O'Donncha, F. (2018). A machine learning framework to forecast wave conditions. *Coastal Engineering*, 137, 1–10.

Karim, M. R. (2018). *Practical Convolutional Neural Networks : Implement advanced deep learning models using Python*. Birmingham: Packt Publishing.

Kim, P. (2017). *MATLAB deep learning : with machine learning, neural networks and artificial intelligence*. New York, NY: Apress.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., et al. (2016). Jupyter Notebooks-a publishing format for reproducible computational workflows. *ELPUB* (pp. 87–90).

Kozłowski, M., Górecki, P., & Szczypiński, P. M. (2019). Varietal classification of barley by convolutional neural networks. *Biosystems Engineering*, 184, 155–165.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436.

Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.

Mayadewi, P., & Rosely, E. (2015). Prediksi Nilai Proyek Akhir Mahasiswa Menggunakan Algoritma Klasifikasi Data Mining. *SESINDO 2015*, 2015.

McAndrew, A. (2016). *A computational introduction to digital image processing*. Boca Raton: Taylor & Francis Group, CRC Press.

- McCulloch, W., & Pitts, W. (1990). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52(1–2), 99–115.
- Miller, T. H., Gallidabino, M. D., MacRae, J. I., Owen, S. F., Bury, N. R., & Barron, L. P. (2019). Prediction of bioconcentration factors in fish and invertebrates using machine learning. *Science of The Total Environment*, 648, 80–89.
- Moolayil, J. (2019). *Learn Keras for deep neural networks : a fast-track approach to modern deep learning with Python*. New York, NY: Apress.
- Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination press San Francisco, CA, USA.
- Nilsback, M.-E., & Zisserman, A. (2006). A visual vocabulary for flower classification. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)* (Vol. 2, pp. 1447–1454).
- Nilsback, M.-E., & Zisserman, A. (2008). Automated flower classification over a large number of classes. *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing* (pp. 722–729).
- Powers, D., & Ailab. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *J. Mach. Learn. Technol*, 2, 2229–3981.
- Ptucha, R., Such, F. P., Pillai, S., Brockler, F., Singh, V., & Hutkowski, P. (2019). Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition*, 88, 604–613.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time

object detection with region proposal networks. *Advances in neural information processing systems* (pp. 91–99).

Rozenblit, O., Haddad, Y., Mirsky, Y., & Azoulay, R. (2018). Machine learning methods for SIR prediction in cellular networks. *Physical Communication*, 31, 239–253.

Rusman, Q., Lucas-Barbosa, D., Poelman, E. H., & Dicke, M. (2019). Ecology of Plastic Flowers. *Trends in Plant Science*, 24(8), 725–740.

Saikia, A. R., Bora, K., Mahanta, L. B., & Das, A. K. (2019). Comparative assessment of CNN architectures for classification of breast FNAC images. *Tissue and Cell*, 57, 8–14. Retrieved from

<http://www.sciencedirect.com/science/article/pii/S0040816618304397>

Saitoh, T., Aoki, K., & Kaneko, T. (2004). Automatic Recognition of Blooming Flowers. *Proceedings - International Conference on Pattern Recognition* (Vol. 1, pp. 27-30 Vol.1).

Sarigül, M., Ozyildirim, B. M., & Avci, M. (2019). Differential convolutional neural network. *Neural Networks*, 116, 279–287.

Shukla, N. (2018). *Machine learning with TensorFlow*. Shelter Island, NY: Manning Publications.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.

(2014). Dropout: a simple way to prevent neural networks from overfitting.

The journal of machine learning research, 15(1), 1929–1958.

Steinbrener, J., Posch, K., & Leitner, R. (2019). Hyperspectral fruit and vegetable classification using convolutional neural networks. *Computers and Electronics in Agriculture*, 162, 364–372.

Suyanto. (2018). *Machine Learning Tingkat Dasar dan Lanjut*. Bandung: Informatika Bandung.

The Editors of Encyclopaedia Britannica. (2019). Flower. *Encyclopædia Britannica, inc.* Retrieved December 9, 2019, from <https://www.britannica.com/science/flower>

Tiay, T., Benyaphaichit, P., & Riyamongkol, P. (2014). Flower recognition system based on image processing. *2014 Third ICT International Student Project Conference (ICT-ISPC)* (pp. 99–102).

Torrey, L., & Shavlik, J. (2010). Transfer learning. *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242–264). IGI Global.

Vasilev, Ivan; Slater, Danie; Spacagna, Gianmario; Roelants, Peter; Zocca, V. (2019). *Python Deep Learning Second Edition*. Birmingham: Packt Publishing Ltd.

Xiao, D., Fang, F., Zheng, J., Pain, C. C., & Navon, I. M. (2019). Machine learning-based rapid response tools for regional air pollution modelling. *Atmospheric Environment*, 199, 463–473.

You, W., Shen, C., Guo, X., Jiang, X., Shi, J., & Zhu, Z. (2017). A hybrid

technique based on convolutional neural network and support vector regression for intelligent diagnosis of rotating machinery. *Advances in Mechanical Engineering*, 9(6), 1687814017704146. SAGE Publications Sage UK: London, England.

Yu, R., & Shi, L. (2018). A user-based taxonomy for deep learning visualization. *Visual Informatics*, 2(3), 147–154.

Zaharchuk, G., Gong, E., Wintermark, M., Rubin, D., & Langlotz, C. P. (2018). Deep Learning in Neuroradiology. *American Journal of Neuroradiology*, 39(10), 1776–1784.

Zhang, S., Zhang, S., Zhang, C., Wang, X., & Shi, Y. (2019). Cucumber leaf disease identification with global pooling dilated convolutional neural network. *Computers and Electronics in Agriculture*, 162, 422–430. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0168169918317976>

Zou, Y., Li, L., Wang, Y., Yu, J., Li, Y., & Deng, W. J. (2015). Classifying digestive organs in wireless capsule endoscopy images based on deep convolutional neural network. *2015 IEEE International Conference on Digital Signal Processing (DSP)* (pp. 1274–1278).

LAMPIRAN

Lampiran Coding

Training

```
# Import system libraries
import re
import os
from os import listdir
from os.path import isfile, join
import PIL
import pickle
from PIL import Image
import pandas as pd
import numpy as np
import keras
from keras.models import Sequential, Model
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Dropout, Flatten, Activation
from keras.layers import Conv2D, MaxPooling2D,
GlobalAveragePooling2D
from keras.callbacks import ModelCheckpoint
from keras.preprocessing import image
from keras.models import load_model
from keras import regularizers
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns

# Set image size and number of classes
img_cols, img_rows = 224, 224
batch_size = 64
train_dir = '/17/training'
validation_dir = '/17/validation'
num_classes = 17

# Generate data for training
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    #
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
```

```

        fill_mode='nearest'
    )

    # Generate data for testing
    validation_datagen = ImageDataGenerator(
        preprocessing_function=preprocess_input,
        # rescale=1./255
    )

    # Load training data
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(img_cols, img_rows),
        batch_size=batch_size,
        class_mode='categorical')

    # Load validation data
    validation_generator = validation_datagen.flow_from_directory(
        validation_dir,
        target_size=(img_cols, img_rows),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False,
    )

    # Set input shape
    input_shape=(img_cols, img_rows, 3)

    # CNN from the strach
    model = Sequential()
    model.add(Conv2D(filters = 16, kernel_size = (5,5),padding =
'same',activation = 'relu', input_shape = input_shape))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Conv2D(filters = 32, kernel_size = (5,5),padding =
'same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Conv2D(filters = 64, kernel_size = (5,5),padding =
'same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Conv2D(filters = 128, kernel_size = (5,5),padding =
'same',activation = 'relu'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(1024))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation = "softmax"))

```

```

# CNN transfer learning
WEIGHTS_PATH_NO_TOP =
"/kaggle/input/resnet50/resnet50_weights_tf_dim_ordering_tf_kerne
ls_notop.h5"
base_model = ResNet50(include_top=False, pooling='avg')
base_model.load_weights(WEIGHTS_PATH_NO_TOP)
model = Sequential()
model.add(base_model)
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(17, activation='softmax'))
base_model.trainable=False
print(model.summary())

# Get the best model
keras_model = '/kaggle/working/flower_cnn_model.h5'
checkpoint = ModelCheckpoint(keras_model,
                             monitor='val_loss',
                             mode='auto',
                             save_best_only=True)
earlystop = EarlyStopping(monitor='val_loss',
                           patience=5)
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                               factor=0.2,
                               patience=3,
                               verbose=1,
                               min_delta=0.0001)
callbacks = [checkpoint]
model.compile(loss='categorical_crossentropy',
              optimizer=keras.optimizers.Nadam(),
              metrics = ['accuracy'])
nb_train_samples = 5283
nb_validation_samples = 1269
epochs = 50
history = model.fit_generator(train_generator,
                             steps_per_epoch=nb_train_samples //
batch_size,
validation_steps=nb_validation_samples // batch_size,
epochs=epochs,
callbacks=callbacks,
validation_data=validation_generator)

```

Scrapping Image

```

from bs4 import BeautifulSoup
import requests
import argparse
import sys
import json
import random
import shutil
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

%matplotlib inline

data_test = '/Labels/oxford17.txt'

def get_random():
    classes = open(data_test).read().splitlines()
    return random.choice(classes)

def scrap(num):
    images_link = []
    for i in range(num):
        query = get_random()
        url="https://www.google.co.in/search?q="+query+" flower petals"+"&source=lnms&tbm=isch"
        header={'User-Agent':"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safari/537.36"}
        soup = BeautifulSoup(requests.get(url, headers=header).text, 'html.parser')
        imageslink=[]
        for i in soup.find_all("div",{"class":"rg_meta"}):
            link , Type =json.loads(i.text)["ou"],json.loads(i.text)["ity"]
            imageslink.append((link,Type))
        if not imageslink:
            return
        imageslink = random.sample(set(imageslink), 1)
        print(imageslink)
        for i , (img , Type) in enumerate( imageslink[0:1]):
            try:
                req = requests.get(img, headers=header, stream=True)
                with open('image.jpg', 'wb') as f:
                    req.raw.decode_content = True
                    shutil.copyfileobj(req.raw, f)
                plt.figure(figsize = (12,8))

```



```

plt.title(query, fontdict={'fontsize': 28})
imgs = mpimg.imread('image.jpg')
imgplot = plt.imshow(imgs)
del req
except Exception as e:
    print("could not load : "+img)
    print(e)
scrap(50)

# Android

package com.rezkyfm.flowclass.tflite;
import android.app.Activity;
import java.io.IOException;
import org.tensorflow.lite.support.common.TensorOperator;
import org.tensorflow.lite.support.common.ops.NormalizeOp;

public class Oxford17 extends Classifier {
    private static final float IMAGE_MEAN = 0.f;
    private static final float IMAGE_STD = 255.0f;

    private static final float PROBABILITY_MEAN = 0.0f;
    private static final float PROBABILITY_STD = 1.0f;

    public Oxford17(Activity activity, Device device, int
numThreads)
        throws IOException {
        super(activity, device, numThreads);
    }

    @Override
    protected String getModelPath() {
        return "oxford17base.tflite";
    }

    @Override
    protected String getLabelPath() {
        return "oxford17.txt";
    }

    @Override
    protected TensorOperator getPreprocessNormalizeOp() {
        return new NormalizeOp(IMAGE_MEAN, IMAGE_STD);
    }

    @Override
    protected TensorOperator getPostprocessNormalizeOp() {

```

```

        return new NormalizeOp(PROBABILITY_MEAN,
PROBABILITY_STD);
    }
}

package com.rezkyfm.flowclass;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Typeface;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.SystemClock;
import android.util.Size;
import android.util.TypedValue;
import android.widget.Toast;
import com.rezkyfm.flowclass.env.BorderedText;
import com.rezkyfm.flowclass.env.Logger;
import com.rezkyfm.flowclass.tflite.Classifier;
import com.rezkyfm.flowclass.tflite.Classifier.Device;
import com.rezkyfm.flowclass.tflite.Classifier.Model;
import java.io.IOException;
import java.util.List;

public class ClassifierActivity extends CameraActivity implements
OnImageAvailableListener {
    private static final Logger LOGGER = new Logger();
    private static final Size DESIRED_PREVIEW_SIZE = new Size(640,
480);
    private static final float TEXT_SIZE_DIP = 10;
    private Bitmap rgbFrameBitmap = null;
    private long lastProcessingTimeMs;
    private Integer sensorOrientation;
    private Classifier classifier;
    private BorderedText borderedText;
    private int imageSizeX;
    private int imageSizeY;

    @Override
    protected int getLayoutId() {
        return R.layout.camera_connection_fragment;
    }
    @Override
    protected Size getDesiredPreviewFrameSize() {
        return DESIRED_PREVIEW_SIZE;
    }
    @Override
    public void onPreviewSizeChosen(final Size size, final int
rotation) {
        final float textSizePx =
            TypedValue.applyDimension(

```

```

TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
borderedText = new BorderedText(textSizePx);
borderedText.setTypeface(Typeface.MONOSPACE);

recreateClassifier(getModel(), getDevice(), getNumThreads());
if (classifier == null) {
    LOGGER.e("No classifier on preview!");
    return;
}
previewWidth = size.getWidth();
previewHeight = size.getHeight();
sensorOrientation = rotation - getScreenOrientation();
LOGGER.i("Camera orientation relative to screen canvas: %d",
sensorOrientation);
LOGGER.i("Initializing at size %dx%d", previewWidth,
previewHeight);
rgbFrameBitmap = Bitmap.createBitmap(previewWidth,
previewHeight, Config.ARGB_8888);
}

@Override
protected void processImage() {
    rgbFrameBitmap.setPixels(getRgbBytes(), 0, previewWidth, 0,
0, previewWidth, previewHeight);
    final int cropSize = Math.min(previewWidth, previewHeight);

    runInBackground(
        new Runnable() {
            @Override
            public void run() {
                if (classifier != null) {
                    final long startTime = SystemClock.uptimeMillis();
                    final List<Classifier.Recognition> results =
                        classifier.recognizeImage(rgbFrameBitmap,
sensorOrientation);
                    lastProcessingTimeMs = SystemClock.uptimeMillis() -
startTime;
                    LOGGER.v("Detect: %s", results);

                    runOnUiThread(
                        new Runnable() {
                            @Override
                            public void run() {
                                showResultsInBottomSheet(results);
                                showFrameInfo(previewWidth + "x" +
previewHeight);
                                showCropInfo(imageSizeX + "x" +
imageSizeY);

```

```

        showCameraResolution(cropSize + "x" +
cropSize);

showRotationInfo(String.valueOf(sensorOrientation));
        showInference(lastProcessingTimeMs + "ms");
    }
    });
}
readyForNextImage();
}
});
}
@Override
protected void onInferenceConfigurationChanged() {
    if (rgbFrameBitmap == null) {
        return;
    }
    final Device device = getDevice();
    final Model model = getModel();
    final int numThreads = getNumThreads();
    runInBackground(() -> recreateClassifier(model, device,
numThreads));
}
private void recreateClassifier(Model model, Device device, int
numThreads) {
    if (classifier != null) {
        LOGGER.d("Closing classifier.");
        classifier.close();
        classifier = null;
    }
    if (device == Device.GPU && model == Model.QUANTIZED) {
        LOGGER.d("Not creating classifier: GPU doesn't support
quantized models.");
        runOnUiThread(
            () -> {
                Toast.makeText(this, "GPU does not yet supported
quantized models.", Toast.LENGTH_LONG)
                    .show();
            });
    }
    return;
}
try {
    LOGGER.d(
        "Creating classifier (model=%s, device=%s,
numThreads=%d)", model, device, numThreads);
    classifier = Classifier.create(this, model, device,
numThreads);
} catch (IOException e) {
    LOGGER.e(e, "Failed to create classifier.");
}
}

```

```
    }  
    imageSizeX = classifier.getImageSizeX();  
    imageSizeY = classifier.getImageSizeY();  
  }  
}
```

