

**KLASIFIKASI SPESIES BUNGA EDELWEIS MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK DAN PENGOPTIMAL
*ADAPTIVE MOMENT ESTIMATION (ADAM)***

SKRIPSI

Diajukan untuk menempuh Ujian Sarjana
pada Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

**WIBI ANTO
NPM 140110200025**



**UNIVERSITAS PADJADJARAN
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
PROGRAM STUDI S-1 MATEMATIKA
JATINANGOR
2024**

LEMBAR PENGESAHAN

**JUDUL : KLASIFIKASI SPESIES BUNGA EDELWEIS
MENGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*
DAN PENGOPTIMAL *ADAPTIVE MOMENT ESTIMATION*
(ADAM)**

**PENULIS : WIBI ANTO
NPM : 140110200025**

Jatinangor, Juli 2024

	Menyetujui,	
Pembimbing Utama		Pembimbing Pendamping

Herlina Napitupulu M.Sc., Ph.D.
NIP 198804082019032015

Nurul Gusriani S.Si., M.Si.
NIP 197008181998032001

Mengetahui,
Ketua Program Studi S-1 Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Padjadjaran

Edi Kurniadi, S.Si., M.Si., Ph.D
NIP 198104182008121003

ABSTRAK

Wibi Anto
NPM 140110200025

Indonesia adalah negara tropis yang memiliki banyak kekayaan hayati. Pemerintah mengatur perlakuan tumbuhan yang dilindungi untuk menjaga kelestariannya. Salah satu karakteristik hewan dan tumbuhan dilindungi adalah endemik, yaitu suatu organisme hanya dapat ditemukan di wilayah tertentu. *Anaphalis javanica* merupakan salah satu spesies edelweis endemik yang ada di Indonesia. Pemerintah telah melakukan berbagai upaya untuk melestarikan bunga edelweis dengan membuat peraturan perundang-undangan dan menetapkan kawasan konservasi dan pelestarian. Pelestarian bunga edelweis secara khusus dilakukan di Desa Wonokitri dengan membudidayakan berbagai bunga edelweis termasuk beberapa jenis yang tidak dilindungi. Spesies bunga edelweis sangat beragam dan memiliki kemiripan warna dan bentuk yang serupa. Perbedaan perlakuan bunga liar dan hasil budidaya juga menambah keberagamannya. Identifikasi spesies bunga edelweis secara akurat menjadi tantangan tersendiri karena perlunya ketelitian yang tinggi. Permasalahan tersebut dapat diatasi dengan melakukan penelitian klasifikasi edelweis menggunakan *Convolutional Neural Network* dengan pengoptimal Adam. Penelitian dilakukan menggunakan sebanyak 3498 data gambar *train* yang kemudian *dibagi ke dalam data train* dan *validation* serta 1050 data gambar test yang terbagi ke dalam tiga kelas. Kelas data dalam penelitian ini merupakan nama-nama spesies edelweis yang ada pada dataset yaitu *Anaphalis javanica*, *Leontopodium alpinum*, dan *Leucogenes grandiceps*. Penelitian ini bertujuan untuk membangun algoritma CNN dengan pengoptimal Adam, memperoleh hasil klasifikasi dan performa akurasi model, dan memperoleh nilai *learning rate* terbaik yang menghasilkan nilai akurasi tertinggi.

Kata kunci: Klasifikasi Gambar; Bunga Edelweis; CNN; Pengoptimal Adam.

ABSTRACT

Wibi Anto
NPM 140110200025

Indonesia is a tropical country that has a lot of biological wealth. The government regulates the treatment of protected plants to maintain their sustainability. One of the characteristics of protected animals and plants is endemic, which means that an organism can only be found in certain areas. Anaphalis javanica is one of the endemic edelweiss species in Indonesia. The government has made various efforts to conserve edelweiss flowers by making laws and regulations and establishing conservation and preservation areas. edelweiss conservation is specifically carried out in Wonokitri Village by cultivating various edelweiss flowers including several species that are not protected. edelweiss flower species are very diverse and have similarities in color and shape. Differences in the treatment of wild and cultivated flowers also add to the diversity. Accurately identifying edelweiss flower species is a challenge because of the need for high accuracy. To try to overcome this problem, edelweiss classification research will be conducted using Convolutional Neural Network with Adam optimizer. The research is conducted using 3498 train image data which will then be divided into train and validation data and 1050 test image data which are divided into three classes. The data classes in this study are the names of edelweiss species in the dataset, namely Anaphalis javanica, Leontopodium alpinum, and Leucogenes grandiceps. This research aims to build a CNN algorithm with Adam optimizer, obtain classification results and model accuracy performance, and obtain the best learning rate value that produces the highest accuracy value.

Keywords: Image Classification; Edelweiss flower; CNN; Adam optimizer.

KATA PENGANTAR

Segala puji dan syukur kepada Allah SWT, karena berkat rahmat, petunjuk, dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul “Klasifikasi Spesies Bunga Edelweis Menggunakan *Convolutional Neural Network* dan Pengoptimal *Adaptive Moment Estimation* (Adam)” Skripsi ini disusun untuk memenuhi salah satu syarat ujian sarjana pada Program Studi S-1 Matematika di Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Skripsi ini tidak dapat selesai tanpa adanya dukungan dan bantuan dari berbagai pihak. Oleh karena itu pada kesempatan ini, penulis mengucapkan terima kasih kepada Ibu Herlina Napitupulu M.Sc., Ph.D., selaku Dosen Pembimbing Utama dan Ibu Nurul Gusriani S.Si., M.Si., selaku Dosen Pembimbing Pendamping yang telah memberikan bimbingan, bantuan, dan dorongan yang sangat berharga kepada penulis dalam proses penyusunan skripsi ini. Selain itu, penulis juga ingin mengucapkan terima kasih kepada:

1. Prof. Dr. Iman Rahayu, S.Si., M.Si., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran.
2. Dr. Ema Carnia, M.Si., selaku Kepala Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran.
3. Edi Kurniadi, S.Si., M.Si., Ph.D., selaku Ketua Program Studi S-1 Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran.
4. Dr. Alit Kartiwa S.Si., M.Si., selaku Dosen Wali penulis.

5. Seluruh Civitas Akademika Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran.
6. Kedua orang tua dan kakak-kakak penulis yang selalu memberikan dukungan penuh, motivasi serta doa yang tidak pernah terputus kepada penulis.
7. Seluruh pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan banyak dukungan dan doa kepada penulis.

Semoga skripsi ini dapat bermanfaat bagi diri penulis sendiri, orang-orang yang membacanya, dan bahkan masyarakat secara luas lewat segala ilmu dan gagasannya.

Jatinangor, Juli 2024

Penulis

DAFTAR ISI

ABSTRAK.....	iii
<i>ABSTRACT</i>	iv
KATA PENGANTAR.....	v
DAFTAR ISI.....	vii
DAFTAR TABEL.....	ix
DAFTAR GAMBAR.....	x
DAFTAR LAMPIRAN.....	xi
DAFTAR NOTASI.....	xii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah.....	5
1.3 Batasan Masalah	5
1.4 Tujuan Penelitian	6
1.5 Kegunaan Penelitian	6
1.6 Metodologi Penelitian.....	6
1.7 Sistematika Penulisan	7
BAB II LANDASAN TEORI.....	8
2.1 Klasifikasi Gambar	8
2.2 Tensor	8
2.3 Prapemrosesan Gambar	9
2.4 <i>Convolutional Neural Network</i>	11
2.5 <i>Convolution Layer</i> Dua Dimensi	12
2.6 <i>Pooling Layer</i> Dua Dimensi	15
2.7 <i>Flatten Layer</i>	16
2.8 <i>Dense Layer</i>	17

2.9	Pengoptimal	19
2.10	Metrik <i>Accuracy</i>	23
2.11	Python	23
BAB III OBJEK DAN METODE PENELITIAN		26
3.1	Objek Penelitian.....	26
3.2	Metode Penelitian	26
3.3	Diagram Alir Penelitian	38
BAB IV HASIL DAN PEMBAHASAN		42
4.1	Prapemrosesan Gambar	42
4.2	<i>Training</i> Model	45
4.3	<i>Testing</i> Model	54
4.4	Performa Model	57
BAB V SIMPULAN DAN SARAN.....		60
5.1	Simpulan	60
5.2	Saran	61
DAFTAR PUSTAKA		62
LAMPIRAN.....		66
RIWAYAT HIDUP PENULIS		76

DAFTAR TABEL

Tabel 1. Performa model pada tahap <i>training</i>	59
--	----

DAFTAR GAMBAR

Gambar 2.1 Arsitektur CNN secara umum.....	11
Gambar 2.2 Tipe arsitektur CNN dengan dua tahap ekstraksi fitur.....	11
Gambar 2.3 Ilustrasi perpindahan piksel dengan $strides = 1$	12
Gambar 2.4 Ilustrasi <i>convolution layer</i> dua dimensi	13
Gambar 2.5 Ilustrasi <i>max pooling</i> 2D dengan input (4, 4) dan <i>pool size</i> (2, 2) ..	14
Gambar 2.6 Ilustrasi <i>flatten layer</i> dengan dimensi input (2, 2, 1)	15
Gambar 2.7 Ilustrasi <i>neuron</i> di <i>dense layer</i>	17
Gambar 2.8 Perbandingan Adam dengan pengoptimal lainnya.....	19
Gambar 2.9 Perbandingan pemilihan <i>learning rate</i>	21
Gambar 3.1 Diagram alir penelitian.....	37
Gambar 3.2 Diagram alir pada tahap <i>training</i> model	38
Gambar 3.3 Diagram alir pada tahap <i>forward propagation</i>	39
Gambar 3.4 Diagram alir pada tahap <i>backpropagation</i>	40
Gambar 4.1 Perubahan ukuran dan kualitas gambar.....	42
Gambar 4.2 Perbedaan gambar setelah dan sebelum dilakukan <i>resize</i>	43
Gambar 4.3 Contoh Hasil klasifikasi pada tahap <i>testing</i> satu <i>mini batch</i>	56
Gambar 4.4 Perbandingan grafik akurasi dan <i>cost</i> pada proses <i>training</i>	57
Gambar 4.5 Perbandingan grafik akurasi dan <i>cost</i> pada proses <i>validation</i>	58

DAFTAR LAMPIRAN

Lampiran 1. Tautan dataset.....	65
Lampiran 2. Tabel hasil performa <i>training</i> dengan <i>learning rate</i> = 10^{-1}	65
Lampiran 3. Tabel hasil performa <i>training</i> dengan <i>learning rate</i> = 10^{-2}	66
Lampiran 4. Tabel hasil performa <i>training</i> dengan <i>learning rate</i> = 10^{-3}	66
Lampiran 5. Tabel hasil performa <i>training</i> dengan <i>learning rate</i> = 10^{-4}	67
Lampiran 6. Tautan hasil testing per <i>mini batch</i>	68
Lampiran 7. Output dari perhitungan lengkap tahap <i>forward propagation</i>	68
Lampiran 8. <i>Syntax</i> Python	69
Lampiran 9. Spesifikasi <i>software</i> dan <i>hardware</i> yang digunakan Dalam penelitian	69

DAFTAR NOTASI

a	: indeks baris pada suatu filter
b	: indeks kolom pada suatu filter
c	: indeks saluran warna pada suatu filter
$\vec{d}^{(L)}$: vektor output <i>dense layer</i> pada <i>layer</i> ke- L
d_j	: elemen output <i>dense layer</i> di <i>layer</i> ke- j
i	: indeks <i>learning rate</i>
j	: indeks <i>neuron</i> pada <i>layer</i> saat ini
k	: indeks <i>neuron</i> pada <i>layer</i> sebelumnya
m	: banyak data pada suatu dataset
n	: banyak kelas pada dataset train
p	: indeks data pada dataset
q	: indeks baris pada output <i>feature map</i>
r	: indeks kolom pada output <i>feature map</i>
s	: indeks saluran warna pada output <i>feature map</i>
t	: indeks iterasi atau indeks <i>mini batch</i>
$w_{jk}^{(L)}$: bobot penghubung antara <i>neuron</i> ke- j pada <i>layer</i> ke- L dan <i>neuron</i> ke- k pada <i>layer</i> sebelumnya
$w_{a,b,c}$: elemen tensor filter baris ke- a , kolom ke- b , dan saluran warna ke- c
$x_k^{(L-1)}$: nilai input untuk <i>dense layer</i> dari <i>neuron</i> pada <i>layer</i> sebelumnya ($L - 1$)
$x_{q,r,s}$: nilai input berupa elemen tensor dari gambar dengan baris ke- q , kolom ke- r , dan saluran warna ke- s
\hat{y}	: hasil prediksi gambar
y	: label gambar sebenarnya pada data <i>train</i>
bs	: ukuran <i>batch</i> (<i>batch size</i>)
lr	: <i>learning rate</i>

lr_i	: <i>learning rate</i> ke- i
$zc^{(L)}$: hasil perhitungan operasi konvolusi pada <i>layer</i> ke- L
\vec{zf}	: output proses <i>flatten</i>
$zd_j^{(L)}$: output proses perhitungan <i>neuron</i> ke- j <i>dense layer</i> pada <i>layer</i> ke- L
$A^{(L)}$: output dari <i>convolution layer</i> pada <i>layer</i> ke- L
$A_{q,r,s}$: elemen output <i>convolution layer</i> dengan indeks baris ke- q , kolom ke- r , dan saluran warna ke- s
L	: indeks <i>layer</i>
N	: banyak <i>neuron</i> pada <i>dense layer</i>
$P^{(L)}$: hasil <i>pooling layer</i> pada <i>layer</i> ke- L
Q	: banyak baris pada <i>feature map</i>
R	: banyak kolom pada <i>feature map</i>
S	: banyak saluran warna pada <i>feature map</i>
T	: banyak iterasi
$W^{(L)}$: filter pada operasi konvolusi di <i>layer</i> ke- L
X	: dataset gambar
X_p	: data ke- p dari dataset X yang berupa tensor
$X^{\{t\}}$: <i>mini batch</i> ke- t dari dataset X
$X_p^{\{t\}}$: data ke- p dari <i>mini batch</i> ke- t pada dataset X
$Xtn^{\{t\}}$: <i>mini batch</i> ke- t dari <i>dataset train</i>
$Xts^{\{t\}}$: <i>mini batch</i> ke- t dari <i>dataset test</i>
$Xval^{\{t\}}$: <i>mini batch</i> ke- t dari <i>dataset validation</i>
$Y_{q,r,s}$: hasil operasi konvolusi baris ke- q kolom ke- r dan saluran warna ke- s
$\alpha_{ReLU}(z)$: fungsi aktivasi ReLU
$\alpha_{sm}(z)$: fungsi aktivasi Softmax
$\vec{\mu}$: vektor kumpulan parameter bobot dan bias
$\beta^{(L)}$: bias filter dari <i>convolution layer</i> pada <i>layer</i> ke- L
$\beta_j^{(L)}$: bias <i>neuron</i> ke- j <i>dense layer</i> pada <i>layer</i> ke- L
γ_1	: konstanta untuk mengontrol seberapa cepat momentum turun

γ_2	: konstanta untuk mengontrol rata-rata eksponensial dari kuadrat gradien
ϵ	: konstanta untuk mencegah pembagian dengan nol pada proses update parameter
g_t	: gradien saat iterasi ke- t
m_t	: pembaruan momentum pertama
\hat{m}_t	: koreksi bias momentum pertama
v_t	: pembaruan momentum kedua
\hat{v}_t	: koreksi bias momentum kedua
θ_t	: pembaruan parameter model pada iterasi ke- t

BAB I

PENDAHULUAN

1.1 Latar Belakang

Indonesia adalah negara tropis yang memiliki banyak kekayaan alam termasuk kekayaan hayati. Sebagai upaya pelestarian dari beragamnya kekayaan alam di Indonesia, pemerintah membagi kawasan pelestarian menjadi taman nasional, taman hutan raya, dan taman wisata alam. Taman nasional adalah kawasan pelestarian alam yang mempunyai ekosistem asli, dikelola dengan sistem zonasi yang dimanfaatkan untuk tujuan penelitian, ilmu pengetahuan, pendidikan, menunjang budidaya, pariwisata, dan rekreasi (Presiden RI, 1990).

Setiap taman nasional memiliki karakteristik yang berbeda bergantung pada lokasi geografis. Perbedaan karakteristik tersebut diantaranya adalah ragam keanekaragaman hayati yang juga didukung dengan perbedaan topografi dari taman nasional. Taman nasional dengan topografi pegunungan cenderung memiliki keragaman hayati yang mampu beradaptasi dengan ketinggian dan suhu yang rendah sedangkan Taman nasional dengan topografi pesisir cenderung memiliki karakteristik hayati yang mampu beradaptasi dan berinteraksi dengan ekosistem laut.

Perbedaan karakteristik taman nasional membuat adanya beberapa hayati yang hanya dapat tumbuh pada lokasi tertentu, salah satunya adalah bunga Edelweis. Menurut Direktorat Jenderal Konservasi Sumber Daya Alam dan Ekosistem, bunga edelweis hanya dapat tumbuh di daerah pegunungan dan memerlukan sinar matahari penuh. Pemerintah melalui Kementerian LHK menyatakan salah satu spesies endemik bunga edelweis yaitu *Anaphalis javanica* sebagai jenis tumbuhan yang dilindungi (Menteri LHK, 2018). Bunga edelweis telah memenuhi kriteria sebagai tumbuhan yang dilindungi. Kriteria tumbuhan dan satwa yang dilindungi adalah: mempunyai populasi yang kecil, adanya penurunan yang tajam pada jumlah individu di alam, dan daerah penyebaran yang terbatas atau disebut juga dengan endemik (Presiden RI, 1999).

Upaya pelestarian bunga edelweis secara khusus dilakukan dengan membudidayakannya di Desa Wonokitri. Desa Wonokitri menjadi tempat budidaya bunga edelweis termasuk beberapa jenis yang tidak dilindungi (Anam, 2021; Kartika, 2023). Pengunjung secara khusus dapat membeli bibit bunga edelweis hasil budidaya untuk dapat ditanam maupun buket bunga sebagai hiasan di desa Wonokitri (Riani, 2024). Namun demikian, bunga edelweis liar terutama dengan jenis *Anaphalis javanica*, tetap menjadi tumbuhan yang dilindungi undang-undang.

Keberagaman spesies bunga edelweis menjadi bertambah dengan adanya perbedaan perlakuan antara bunga liar dan hasil budidaya dengan bentuk dan warna yang serupa (Malasari, 2022). Ketelitian sangat diperlukan dalam mengetahui jenis bunga edelweis agar tidak melanggar undang-undang dan mendapatkan sanksi yang

berat. Hal ini menghadirkan tantangan dalam identifikasi spesies bunga edelweis secara akurat.

Salah satu bidang *Machine Learning* yaitu *Computer Vision* mampu menjawab tantangan tersebut. Salah satu cabang *Computer Vision* yaitu *Image Classification* atau klasifikasi gambar dengan menggunakan *Convolutional Neural Network* (CNN) mampu melakukan pengenalan pola dan fitur visual yang rumit, sehingga memungkinkan untuk membedakan spesies-spesies bunga edelweis dengan lebih tepat.

Klasifikasi gambar menggunakan CNN secara umum memiliki dua buah tahapan pelatihan model yaitu proses ekstraksi fitur dan proses klasifikasi (Géron, 2019). Proses ekstraksi fitur bertujuan untuk memperoleh fitur yang ada pada gambar dan proses tersebut dilakukan dengan menggunakan operasi konvolusi dua dimensi dan operasi *max pooling* dua dimensi. Setelah mendapatkan hasil ekstraksi, selanjutnya dilakukan proses *flatten* untuk mengubah dimensi dari hasil ekstraksi fitur menjadi satu dimensi. Kemudian dilanjutkan dengan proses klasifikasi pada lapisan *fully-connected layer* berupa *dense layer*.

Penelitian mengenai klasifikasi gambar menggunakan CNN telah banyak dilakukan dengan berbagai objek. Alkaff & Prasetyo (2022) mengaplikasikan CNN untuk klasifikasi penyakit tanaman tomat yang menyerang daun. Model CNN dibuat dengan mengoptimasi *hyperparameter* menggunakan *Hyperband* dan mencapai tingkat akurasi sebesar 95,69% pada proses *train*, 88,5% pada proses *validation*, dan 88,6% pada proses *test*.

Muhammad & Wibowo (2021) menggunakan CNN dengan arsitektur ResNet50v2 untuk mengklasifikasikan empat jenis tanaman *Aglaonema* menggunakan 1960 gambar. Eksperimen dalam penelitian ini dilakukan dengan mencoba melakukan *training* dengan menghilangkan *background* gambar, mengubah *learning rate*, dan mengubah nilai parameter *dropout* dan menghasilkan akurasi tertinggi pada proses *testing* sebesar 99% menggunakan gambar dengan *background* dan 71% menggunakan gambar tanpa *background*.

Penelitian klasifikasi bunga edelweis juga telah dilakukan oleh Malau & Mulyana (2022) dengan mengklasifikasikan dua jenis bunga edelweis yaitu *Anaphalis javanica* dan *Leontopodium alpinum* menggunakan metode *Linear Discriminant Analysis* (LDA). Model yang dibuat dalam penelitian tersebut mencapai tingkat akurasi 100% dengan menggunakan 1500 data gambar *train* pada saat proses *training* dan 99,77% dengan menggunakan 450 data gambar *test* pada proses *testing*.

Pada penelitian ini dilakukan klasifikasi gambar edelweis seperti yang dilakukan Malau & Mulyana (2022). Perbedaan penelitian ini dengan Malau & Mulyana (2022) adalah penggunaan metode CNN dengan pengoptimal Adam. Perbedaan penggunaan metode CNN pada penelitian ini dengan penelitian Muhammad & Wibowo (2021) terdapat pada penggunaan arsitektur CNN yang berbeda. Jumlah kelas yang digunakan pada penelitian ini juga lebih banyak dibandingkan dengan Malau & Mulyana (2022) dengan tiga buah kelas berupa jenis bunga edelweis yaitu *Anaphalis javanica*, *Leontopodium alpinum*, dan *Leucogenes*

grandiceps. Penelitian ini menggunakan 3498 data gambar *train* yang kemudian dibagi ke dalam data *train* dan *validation* serta 1050 data gambar *test*.

1.2 Identifikasi Masalah

Adapun identifikasi masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan (algoritma) CNN dengan pengoptimal *Adaptive Moment Estimation* dalam mengklasifikasikan spesies bunga edelweis.
2. Bagaimana performa hasil klasifikasi spesies bunga edelweis menggunakan model CNN dengan beberapa *learning rate* berdasarkan akurasi?

1.3 Batasan Masalah

Batasan permasalahan dalam penelitian ini adalah:

1. Klasifikasi spesies bunga edelweis berdasarkan gambar bunga dilakukan menggunakan model *Machine Learning* yaitu *Convolutional Neural Network* dan pengoptimal menggunakan *Adaptive Moment Estimation* (Adam).
2. Besaran parameter *learning rate* yang digunakan adalah 10^{-i} , $i = 1, 2, 3, 4$
3. Performa model *Machine Learning* dihitung menggunakan metrik akurasi.
4. Parameter *decay rate* untuk momentum, *decay rate* untuk *gradient*, dan konstanta ϵ pada Adam secara berturut-turut sebesar 0,9; 0,999; dan 10^{-8} .
5. Data yang digunakan dalam penelitian ini adalah gambar-gambar dari bunga edelweis yang diambil dari *Kaggle* yang terdiri dari tiga spesies bunga edelweis yang berbeda.

6. Bahasa pemrograman yang digunakan adalah bahasa pemrograman Python menggunakan *Google Colaboratory*.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Membangun algoritma CNN dengan pengoptimal *Adaptive Moment Estimation* untuk mengklasifikasikan spesies bunga edelweis.
2. Mengukur performa hasil klasifikasi spesies bunga edelweis menggunakan model CNN dengan beberapa *learning rate* berdasarkan akurasi.

1.5 Kegunaan Penelitian

Kegunaan dari penelitian ini adalah sebagai berikut:

1. Model klasifikasi spesies bunga edelweis menggunakan *Convolutional Neural Network* dan pengoptimal Adam dapat digunakan untuk mengetahui spesies bunga edelweis hanya dengan menggunakan gambar.
2. Alat bantu untuk pembaca khususnya para pendaki maupun masyarakat umum dalam mengenali dan mengidentifikasi bunga edelweis guna mendukung pelestarian bunga Edelweis.

1.6 Metodologi Penelitian

Metodologi Penelitian terdiri dari studi literatur dan eksperimental:

1. Studi literatur dilakukan dengan mempelajari teori terkait *Computer Vision*, klasifikasi gambar, algoritma Adam, dan *Convolutional Neural Network* dari berbagai sumber berupa buku, jurnal, dan artikel yang tersedia secara daring.

2. Studi Eksperimental dilakukan dengan membuat arsitektur model CNN dengan pengoptimal Adam serta mencari nilai parameter *learning rate* terbaik untuk klasifikasi jenis bunga edelweis menggunakan bahasa pemrograman Python pada *Google Colaboratory*.

1.7 Sistematika Penulisan

Sistematika penulisan pada skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN, pada bab ini dijelaskan mengenai latar belakang, identifikasi masalah, batasan masalah, tujuan penelitian, kegunaan penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI, pada bab ini dijelaskan mengenai teori-teori yang menjadi acuan dasar dalam penelitian yaitu klasifikasi gambar, tensor, prapemrosesan gambar, *Convolution Neural Network*, *convolution layer* dua dimensi, *pooling layer* dua dimensi, *flatten layer*, *dense layer*, pengoptimal, metrik *accuracy*, dan Python.

BAB III OBJEK DAN METODE PENELITIAN, pada bab ini berisi tentang objek penelitian, metode penelitian, dan alur penelitian.

BAB IV HASIL DAN PEMBAHASAN, pada bab ini berisi pengolahan data gambar dan hasil penelitian yang dilakukan.

BAB V SIMPULAN DAN SARAN, pada bab ini berisi simpulan dan saran dari pembahasan penelitian yang telah dilakukan untuk peneliti selanjutnya.

BAB II

LANDASAN TEORI

2.1 Klasifikasi Gambar

Klasifikasi gambar atau *Image Classification* adalah proses kategorisasi dan pemberian label dari sekumpulan piksel atau vektor dari suatu gambar berdasarkan aturan tertentu (Shinozuka & Mansouri, 2009). Terdapat dua jenis metode klasifikasi yaitu *supervised* dan *unsupervised*. Metode *unsupervised* adalah proses klasifikasi yang sepenuhnya otomatis tanpa menggunakan *training data* sedangkan metode *supervised* adalah proses klasifikasi menggunakan *training data* dan mengelompokkan ke dalam kategori yang telah dipilih sebelumnya (Shinozuka & Mansouri, 2009).

2.2 Tensor

Tensor adalah susunan angka yang disusun pada *grid* (kisi) biasa dengan jumlah sumbu (dimensi) yang dapat diatur (Goodfellow et al., 2016). Dalam pemrograman, tensor merupakan *array* dengan n dimensi yang juga berarti merupakan perluasan dari matriks yang merupakan *array* dengan dua dimensi (Jean, 2018). Misal \mathbf{X} adalah tensor dengan tiga sumbu (dimensi) maka elemen dari \mathbf{X} pada koordinat (m, n, p) adalah $x_{m,n,p}$. Secara matematis, tensor dinyatakan sebagai berikut.

$$\mathbf{X} = \begin{bmatrix} [x_{1,1,1} & x_{1,1,2} & \dots & x_{1,1,s}] & [x_{1,2,1} & x_{1,2,2} & \dots & x_{1,2,s}] & \dots & [x_{1,r,1} & x_{1,r,2} & \dots & x_{1,r,s}] \\ [x_{2,1,1} & x_{2,1,2} & \dots & x_{2,1,s}] & [x_{2,2,1} & x_{2,2,2} & \dots & x_{2,2,s}] & \dots & [x_{2,r,1} & x_{2,r,2} & \dots & x_{2,r,s}] \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots \\ [x_{q,1,1} & x_{q,1,2} & \dots & x_{q,1,s}] & [x_{q,2,1} & x_{q,2,2} & \dots & x_{q,2,s}] & \dots & [x_{q,r,1} & x_{q,r,2} & \dots & x_{q,r,s}] \end{bmatrix}$$

$$\mathbf{X} \in \mathbb{R}^{m,n,p}$$

2.3 Prapemrosesan Gambar

Prapemrosesan gambar atau *Image Preprocessing* adalah suatu metode untuk mengubah data gambar mentah yang mengandung *noise* dan nilai yang tidak sesuai menjadi data gambar yang bersih dan siap digunakan (Chaki & Dey, 2018). Chaki & Dey (2018) menyatakan bahwa *noise* ataupun nilai yang tidak sesuai dapat diatasi dengan beberapa cara diantaranya: *image correction*, *image enhancement*, *image restoration*, dan *image compression*. *Image compression* atau kompresi gambar digunakan untuk mengurangi ukuran gambar baik dengan mengurangi dimensi maupun kualitas gambar. Salah satu cara untuk menurunkan kualitas gambar dan menyeragamkan ukuran piksel adalah dengan menggunakan *library* PIL. Algoritma untuk melakukan kompresi gambar dengan menggunakan *library* PIL dinyatakan sebagai berikut.

1. Inisiasi lebar gambar baru dalam satuan piksel = 512 px
2. Ambil ukuran lebar dan tinggi gambar asli
3. Hitung rasio gambar asli
4. Cari tinggi gambar baru dengan membagi lebar gambar baru dengan rasio gambar asli
5. Simpan dengan kualitas gambar sebesar 85%

Selain menggunakan *library* PIL, Prapemrosesan gambar juga dapat dilakukan menggunakan TensorFlow menggunakan fungsi *ImageDataGenerator* dan beberapa metode dalam fungsi tersebut sesuai dengan kebutuhan seperti

flow_from_directory (TensorFlow Developer, 2024). Setidaknya terdapat satu prapemrosesan gambar dalam *neural network* yaitu mengubah gambar menjadi Tensor. Beberapa jenis prapemrosesan gambar lainnya yang menggunakan TensorFlow diantaranya adalah penyeragaman dimensi, normalisasi nilai piksel, pembagian data *train* dan *validation*, pembagian *batch*, dll.

Nilai piksel secara *default* dari gambar berada pada rentang 0 hingga 255. Prapemrosesan berupa normalisasi nilai piksel dilakukan dengan membagi setiap piksel dengan 255 sehingga nilai piksel setelah dinormalisasi berada pada rentang 0 dan 1. Secara matematis, normalisasi nilai piksel dinyatakan sebagai berikut.

$$\text{normalisasi}(\mathbf{X}) = \frac{1}{255} \mathbf{X} \quad (2.1)$$

Pembagian *batch* menjadi *mini batch* dilakukan untuk menambah banyak proses iterasi untuk tahap optimasi parameter menggunakan pengoptimal Adam. Ukuran banyak data pada setiap *mini batch* disebut dengan *batch size*. Misal X adalah dataset dengan data sebanyak m , \mathbf{X}_i data ke- i dari dataset X yang berupa tensor dan bs adalah *batch size*.

$$X = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m\}$$

maka jumlah proses iterasi dapat dihitung menggunakan persamaan sebagai berikut.

$$T = \left\lceil \frac{m}{bs} \right\rceil \quad (2.2)$$

mini batch dinotasikan dengan $X^{\{t\}}$ dan dapat dinyatakan dengan sebagai berikut.

$$X^{\{t\}} = \{\mathbf{X}_{(bs \cdot (t-1)) + 1}, \mathbf{X}_{(bs \cdot t) + 2}, \dots, \mathbf{X}_{bs \cdot t}\} \quad (2.3)$$

dimana $t = 1, 2, 3, \dots, T$ sehingga dataset X berubah menjadi sebagai berikut.

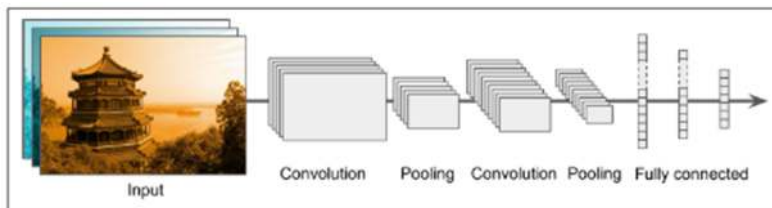
$$X = \{X_1, X_2, \dots, X_{bs}, X_{bs+1}, \dots, X_{2bs}, X_{2bs+1}, \dots, X_{3bs}, \dots, X_{(t-1)bs+1}, \dots, X_{(m)}\} \quad (2.4)$$

$$X = \{X^{\{1\}}, X^{\{2\}}, \dots, X^{\{T\}}\} \quad (2.5)$$

2.4 Convolutional Neural Network

Komputer mengenal sebuah gambar sebagai suatu tensor dengan entri berupa angka yang merepresentasikan warna dari setiap piksel-nya. Pada proses melakukan klasifikasi gambar, angka-angka pada tensor tersebut dipelajari oleh mesin untuk mencari pola tertentu. Salah satu cara untuk mempelajari pola tersebut adalah menggunakan Convolutional Neural Network.

Convolutional Neural Network (CNN) adalah suatu jaringan syaraf tiruan dengan proses konvolusi yang muncul dari studi tentang visual korteks otak dan telah digunakan mulai dari tahun 1980an (Géron, 2019). Menurut Ketkar (2017), proses konvolusi merupakan suatu perhitungan matematika yang digunakan untuk melakukan ekstraksi fitur pada gambar. Secara umum, arsitektur CNN dapat dilihat pada Gambar 2.1.



Gambar 2.1 Arsitektur CNN secara umum (Géron, 2019)

Dari Gambar 2.1 terlihat bahwa arsitektur CNN pada umumnya terdiri dari tiga jenis *layer* yaitu *convolution layer*, *pooling layer*, dan *fully-connected layer* (Géron, 2019).

Dalam proses mempelajari data berupa gambar, CNN melakukan dua jenis propagasi yaitu *forward propagation* dan *backpropagation*. *Forward propagation* bertujuan untuk memprediksi output dari *input layer* hingga *output layer* dengan mengambil input, menghitung dengan bobot, menambahkan bias, mengaplikasikan fungsi aktivasi, dan memberikan memberikan hasil nya ke *layer* selanjutnya hingga mencapai *layer* terakhir(Gupta, 2023).

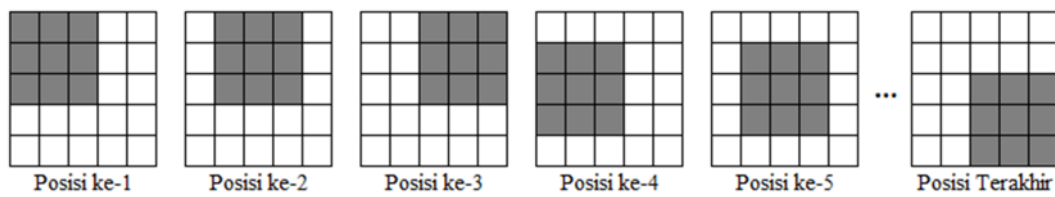
Sedangkan *backpropagation* bertujuan untuk memperbarui nilai parameter (bobot dan bias) dengan menghitung kesalahan hasil klasifikasi pada *forward propagation* (*cost function*). Hasil klasifikasi berupa *cost function* kemudian diturunkan terhadap suatu parameter dan digunakan untuk memperbarui parameter tersebut dengan menggunakan pengoptimal(Gupta, 2023).

2.5 *Convolution Layer Dua Dimensi*

Convolution layer dua dimensi merupakan lapisan neuron dengan operasi konvolusi. Operasi konvolusi dapat dilakukan menggunakan beberapa alat salah satunya adalah menggunakan Python. Pada Python, membuat *convolution layer* dapat menggunakan *library* TensorFlow yang didalamnya telah menyediakan fungsi Conv2D.

Pada fungsi Conv2D terdapat beberapa parameter yang dapat diubah diantaranya adalah *filters*, *kernel_size*, *activation*, dan *strides* (TensorFlow Developer, 2024a). *Filters* merupakan argumen yang menyatakan banyak tensor yang digunakan dalam operasi konvolusi untuk mengekstraksi fitur. Dimensi dari tensor tersebut dinamakan *kernel size*, beberapa *kernel size* yang sering digunakan adalah (3,3), (5,5), dan (7,7).

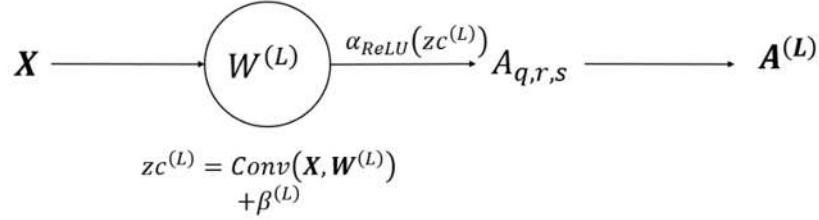
Fungsi Conv2D memiliki parameter *strides* yang menunjukkan banyak perpindahan piksel setelah operasi konvolusi. Nilai *strides* = 1 berarti setiap operasi konvolusi dilakukan, terjadi perpindahan satu piksel untuk operasi konvolusi selanjutnya. Ilustrasi dari *strides* = 1 dengan filter W yang berukuran (3,3) dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi perpindahan piksel dengan *strides* = 1

Proses konvolusi selalu diawali dari melakukan operasi konvolusi *input feature map* kiri atas yang tergambar sebagai area yang diarsir pada Posisi ke-1. Setelah operasi konvolusi pada Posisi ke-1 selesai dilakukan, maka dilakukan perpindahan piksel sebanyak satu posisi seperti Posisi ke-2 dan kembali dilakukan operasi konvolusi dengan input *feature map* berupa area yang diarsir pada Posisi ke-2. Proses perpindahan piksel terus dilakukan hingga mencapai Posisi Terakhir.

Operasi konvolusi pada *convolution layer* dua dimensi meliputi proses aktivasi dengan menggunakan *activation function*. Secara keseluruhan, proses operasi konvolusi menghasilkan hasil $zc^{(L)}$ yang kemudian diproses dengan fungsi aktivasi $\alpha_{ReLU}(zc^{(L)})$ dan menghasilkan komponen *feature map* output $A_{q,r,s}$. Proses konvolusi dilakukan berulang dengan memindahkan input *feature map* dengan satu *strides* sehingga menghasilkan $A^{(L)}$. Ilustrasi dari proses pada *convolution layer* dua dimensi dapat dilihat pada Gambar 2.3.



Gambar 2.3 Ilustrasi *convolution layer* dua dimensi

feature map hasil operasi konvolusi pada *layer* ke- L terhadap suatu tensor \mathbf{X} dan filter \mathbf{W} dengan bias β pada *layer* L dinyatakan sebagai berikut.

$$z_c^{(L)} = \text{Conv}(\mathbf{X}, \mathbf{W}^{(L)}) + \beta^{(L)} \quad (2.6)$$

dimana operasi konvolusi dua dimensi dengan satu filter dan C saluran warna dinyatakan sebagai berikut.

$$\text{Conv}(\mathbf{X}, \mathbf{W}) = \mathbf{X} * \mathbf{W}$$

$$\text{Conv}(\mathbf{X}, \mathbf{W}) = Y_{q,r,s} = \sum_{c=1}^{K_3} \sum_{a=1}^{K_1} \sum_{b=1}^{K_2} x_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c} \quad (2.7)$$

dimana $y_{q,r,s}$ merupakan hasil konvolusi baris ke- q kolom ke- r dan saluran warna ke- s , \mathbf{X} merupakan input *feature map*, dan \mathbf{W} merupakan filter dengan dimensi (K_1, K_2, K_3) . Filter \mathbf{W} didapat dengan menggunakan inisialisasi *glorot uniform*. *Glorot uniform* adalah suatu metode inisialisasi bobot (*weight*) dengan mendistribusikan nilai-nilai acak dari distribusi *uniform* dengan tujuan untuk mempertahankan keseimbangan nilai-nilai pengali saat *forward propagation* dan *backpropagation* (Glorot & Bengio, 2010).

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang sering digunakan dalam *convolution layer* (Ketkar, 2017). ReLU memproses nilai z_c dengan mengubah setiap nilai yang bernilai negatif menjadi nol

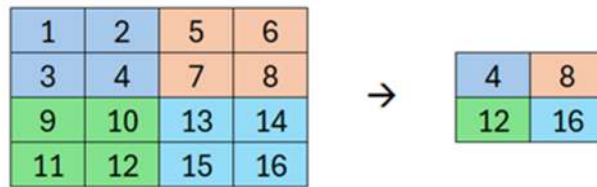
$$A_{q,r,s} = \alpha_{ReLU}(zc^{(L)}) = \max(0, zc^{(L)}) \quad (2.8)$$

sehingga output dari *layer* Conv2D ke- L dapat dilihat pada persamaan 2.9.

$$A^{(L)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ A_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ A_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (2.9)$$

2.6 Pooling Layer Dua Dimensi

Pooling Layer adalah suatu lapisan neuron dengan operasi penggabungan atau *pooling*. Operasi *pooling* dilakukan untuk memperkecil dimensi dengan menggabungkan nilai pada area tertentu sesuai dengan ukuran filter. Terdapat dua buah jenis operasi penggabungan yaitu *max pooling* dan *average pooling*. Operasi *pooling* dapat dilakukan dengan bantuan Python pada *library* TensorFlow yang telah menyediakan fungsi *MaxPooling2D* dengan parameter masukan *pool size* berupa *tuple* yang secara berurutan menyatakan dimensi filter dan nilai *strides*. Ilustrasi operasi *max pooling* 2D dapat dilihat pada Gambar 2.4.



Gambar 2.4 Ilustrasi *max pooling* 2D dengan input (4, 4, 1) dan *pool size* (2, 2)

Operasi *max pooling* dilakukan dengan mengambil nilai maksimal dari sekumpulan nilai piksel pada area tertentu. Misalkan *pool size* dari suatu operasi *max pooling* berukuran (2,2) yang berarti memiliki dimensi filter sebesar 2 dan *strides* dengan nilai 2, maka di setiap area dengan ukuran (2,2) dari input tensor akan diambil nilai terbesarnya dengan perpindahan sebanyak dua piksel. Secara matematis, output dari *pooling layer* ke- L dengan input \mathbf{X} dapat dinyatakan sebagai berikut.

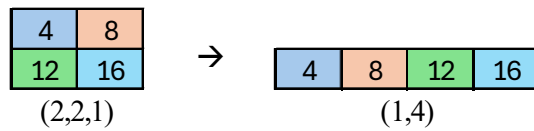
$$\mathbf{P}^{(L)} = \text{MaxPooling2D}(\mathbf{X}) \quad (2.10)$$

2.7 Flatten Layer

Flatten layer adalah adalah suatu lapisan neuron yang memiliki operasi untuk membuat datar atau *flatten* dimensi dari tensor input yang masuk ke dalam *layer* tersebut. Misalkan terdapat \mathbf{X} sebuah tensor dengan dimensi (Q, R, S) , maka output dari *flatten layer* dapat dinyatakan sebagai berikut.

$$FL(\mathbf{X}) = \vec{zf} = [x_{1,1,1} \ x_{1,2,1} \ x_{2,1,1} \ \dots \ x_{q,r,s} \ \dots \ x_{Q,R,S}] \quad (2.11)$$

dimana \vec{zf} merupakan vektor baris, $q = 1, 2, \dots, Q$; $r = 1, 2, \dots, R$; dan $s = 1, 2, \dots, S$. Ilustrasi proses *flatten* pada *flatten layer* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Ilustrasi *flatten layer* dengan dimensi input (2, 2, 1)

Apabila terdapat input suatu tensor dengan dimensi (2,2,1) seperti pada Gambar 2.6 maka *flatten layer* akan mengubah tensor tersebut menjadi satu dimensi secara berurutan dengan aturan prioritas urutan dari baris, kolom, dan saluran warna.

2.8 Dense Layer

Dense layer dalam arsitektur CNN berfungsi sebagai modul pengklasifikasi hasil konvolusi, serangkaian *dense layer* yang saling terhubung disebut dengan *fully-connected layer* (LeCun et al., 2010). TensorFlow telah menyediakan fungsi *Dense* untuk membuat *fully-connected layer*. Terdapat dua parameter dalam fungsi *Dense* yang sering diubah dalam pembuatan model CNN yaitu *units* dan *activation*. *Units* merujuk pada banyaknya sel neuron yang terdapat pada satu *layer* tersebut dan *activation* merujuk pada fungsi aktivasi yang digunakan. Secara matematis, *output* dari setiap *dense layer* dinyatakan sebagai berikut.

$$zd_j^{(L)} = \sum_{k=1}^N w_{jk}^{(L)} \cdot x_k^{(L-1)} + \beta_j^{(L)} \quad (2.12)$$

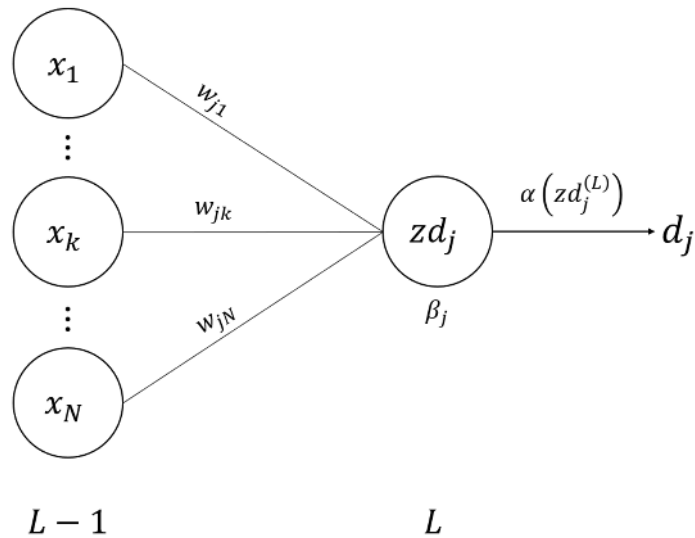
dimana $zd_j^{(L)}$ adalah nilai *neuron* ke- j pada *layer* ke- L , w_{jk} adalah bobot penghubung antara *neuron* ke- j pada *layer* L dan *neuron* ke- k pada *layer* sebelumnya, $x_k^{(L-1)}$ adalah nilai input untuk *dense layer* dari *neuron* pada *layer* sebelumnya, N adalah banyaknya *neuron* pada *layer* sebelumnya, dan $\beta_j^{(L)}$ adalah bias pada *neuron* ke- j pada *layer* L . Secara umum, persamaan fungsi aktivasi dinyatakan sebagai berikut.

$$d_j = \alpha(zd_j^{(L)}) \quad (2.13)$$

sehingga output *dense layer* ke- L dengan fungsi aktivasi α dan memiliki sebanyak N *neuron* dapat dinyatakan sebagai vektor baris berikut.

$$\vec{d}^{(L)} = [d_1 \ d_2 \ \dots \ d_j \ \dots \ d_N] \quad (2.14)$$

Ilustrasi dari proses perhitungan pada *dense layer* dapat dilihat pada Gambar 2.6 dimana terdapat nilai input dari layer sebelumnya yang dinotasikan dengan x dan diproses di dalam *dense layer* dengan mengalikan nilai x dengan bobot penghubung dengan neuron sebelumnya sehingga diperoleh zd_j . Nilai zd_j kemudian ditambahkan dengan bias β_j dan diproses dengan *activation function* sehingga diperoleh d_j .



Gambar 2.6 Ilustrasi *neuron* di *dense layer*

1. Fungsi Aktivasi *Rectified Linear Unit*

Fungsi aktivasi dalam CNN adalah fungsi matematika yang diterapkan pada keluaran setiap neuron dalam jaringan. Fungsi ini menentukan apakah neuron harus diaktifkan atau tidak berdasarkan masukan yang diterimanya. Tujuan dari fungsi aktivasi adalah untuk memperkenalkan non-linearitas ke dalam keluaran neuron, sehingga memungkinkan jaringan untuk belajar dan memodelkan pola kompleks dalam data. Non-linearitas ini sangat penting agar

jaringan dapat belajar dan membuat prediksi yang akurat. Fungsi aktivasi yang umum digunakan dalam CNN meliputi *Rectified Linear Unit* (ReLU) atau unit linear tereduksi, yang banyak digunakan karena kesederhanaan komputasinya dan efektivitas dalam pelatihan, dan fungsi aktivasi linear. Notasi matematis dari *dense layer* dengan fungsi aktivasi *Rectified Linear Unit* (ReLU) dinyatakan sebagai berikut.

$$\alpha_{ReLU}(zd_j^{(L)}) = \max(0, zd_j^{(L)}) \quad (2.15)$$

2. Fungsi Aktivasi Softmax

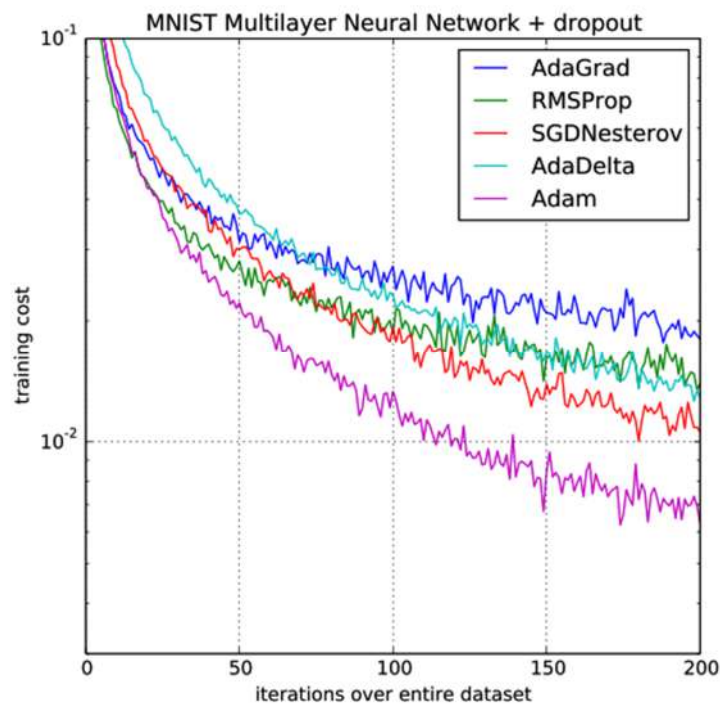
Fungsi aktivasi Softmax adalah suatu fungsi aktivasi *layer* pada *neural network* dengan menggunakan regresi Softmax. Regresi Softmax adalah perluasan dari regresi logistik dengan kemungkinan nilai output sebanyak n nilai. Pada kasus *klasifikasi gambar*, regresi Softmax digunakan untuk melakukan klasifikasi dengan jumlah kelas lebih dari dua (n kelas). Notasi matematis dari fungsi aktivasi Softmax dinyatakan sebagai berikut.

$$\alpha_{sm}(zd_j^{(L)}) = \frac{e^{z_j^{(L)}}}{\sum_{j=1}^n e^{z_j^{(L)}}} \quad (2.16)$$

2.9 Pengoptimal

Pengoptimal merupakan sebuah fungsi dengan tujuan meminimumkan *loss function* dengan mengatur parameter bias dan bobot. Terdapat beberapa pengoptimal yang sering digunakan dalam *neural network* diantaranya adalah *Stochastic Gradient Descend* (SGD), *Root Mean Square Propagation* (RMSprop), *Adaptive Gradient* (AdaGrad), *Adaptive Moment Estimation* (Adam), dll (Chauchan, 2020). Kingma & Ba (2017) menyatakan bahwa *Adam* kuat dan cocok

untuk berbagai macam masalah optimasi non-konveks di bidang *machine learning* dengan keandalannya yang mampu meminimumkan nilai dari fungsi biaya lebih cepat dibandingkan pengoptimal lainnya. Perbandingan performa Adam dengan pengoptimal lainnya dapat dilihat pada Gambar 2.7.



Gambar 2.7 Perbandingan Adam dengan pengoptimal lainnya (Kingma & Ba, 2017)

Dari gambar 2.8 diperoleh bahwa pengoptimal Adam yang digambarkan dengan grafik berwarna ungu memiliki nilai *training cost* terendah dan dapat mencapai nilai tersebut lebih cepat dibandingkan dengan pengoptimal lainnya.

1. Loss function

Categorical cross-entropy adalah fungsi *loss* yang digunakan untuk klasifikasi multi-kelas. Misal \hat{y} adalah hasil prediksi dari model serta y

adalah nilai dari label yang sebenarnya pada data *train*, maka *loss function* dari *categorical cross-entropy* dinyatakan sebagai berikut.

$$Loss(\hat{y}, y) = -\sum_{j=1}^n y_j \cdot \log(\hat{y}_j) \quad (2.17)$$

dimana $j = 1, 2, \dots, n$ dengan n merupakan banyak kelas dari data *train* yang digunakan.

2. Cost Function

Cost function digunakan untuk mencari nilai *loss* rata-rata dengan menghitung setiap nilai *loss* yang diperoleh dari setiap hasil prediksi dan tiap label y pada data *train*. Misal terdapat sebanyak m hasil prediksi dan label y , maka *Cost function* dapat dinyatakan sebagai berikut.

$$J = \frac{1}{m} \sum_{p=1}^m Loss(\hat{y}_j^{(p)}, y_j^{(p)}) \quad (2.18)$$

dimana $p = 1, 2, \dots, m$.

3. Algoritma Adaptive Moment Estimation

Adaptive Moment Estimation atau Adam merupakan suatu algoritma pengoptimal berbasis stokastik gradien yang efisien karena hanya membutuhkan order pertama dari gradien dengan memakan memori yang minimal (Kingma & Ba, 2017). Misal terdapat fungsi $J(\theta)$ yang merupakan *cost function* yang diferensiabel terhadap parameter model θ dan nilai awal $m_t = v_t = 0$, maka aturan pembaruan adam adalah sebagai berikut.

$$g_t = \frac{d}{d\theta} J(\theta) \quad (2.19)$$

$$m_t = \gamma_1 \cdot m_{t-1} + (1 - \gamma_1) \cdot g_t \quad (2.20)$$

$$v_t = \gamma_2 \cdot v_{t-1} + (1 - \gamma_2) \cdot g_t^2 \quad (2.21)$$

$$\hat{m}_t = \frac{m_t}{1-\gamma_1^t} \quad (2.22)$$

$$\hat{v}_t = \frac{v_t}{1-\gamma_2^t} \quad (2.23)$$

$$\theta_t = \theta_{t-1} - lr_i \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (2.24)$$

dimana:

t : iterasi ke- t

lr_i : *learning rate* ke- i

γ_1 : konstanta untuk mengontrol seberapa cepat momentum turun

γ_2 : konstanta untuk mengontrol rata-rata eksponensial dari kuadrat gradien

ϵ : konstanta 10^{-8}

g_t : gradien saat iterasi ke- t

m_t : pembaruan momentum pertama

\hat{m}_t : koreksi bias momentum pertama

v_t : pembaruan momentum kedua

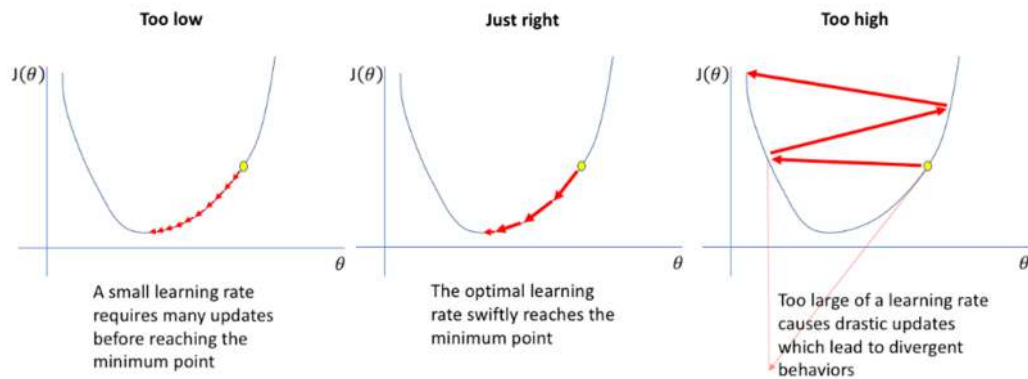
\hat{v}_t : koreksi bias momentum kedua

θ_t : pembaruan parameter model pada iterasi ke- t

4. *Learning Rate*

Learning rate (lr) adalah salah satu parameter dalam pengoptimal yang mengontrol seberapa besar langkah yang diambil dalam penyesuaian bobot model. Pemilihan nilai *learning rate* sangat mempengaruhi hasil model karena ketika nilai *learning rate* terlalu besar, maka melewati nilai *loss* yang minimum sedangkan ketika nilai *learning rate* terlalu kecil, maka membutuhkan waktu yang sangat banyak dalam menuju titik konvergensi

dan mendapatkan nilai *loss* minimum (Jordan, 2018). Perbandingan pemilihan nilai *learning rate* dapat dilihat pada Gambar 2.8.



Gambar 2.8 Perbandingan pemilihan learning rate (Jordan, 2018)

2.10 Metrik *Accuracy*

Evaluasi dilakukan dengan menguji model menggunakan dataset *test* dengan menghitung banyak prediksi yang benar dibagi dengan seluruh prediksi.

$$Acc = \frac{\text{Prediksi benar}}{\text{Seluruh prediksi}} \quad (2.25)$$

2.11 Python

Bahasa Python merupakan suatu bahasa pemrograman buatan Guido van Rossum yang pertama kali dirilis pada tahun 1991 dan merupakan sebuah bahasa pemrograman tingkat tinggi yang populer digunakan berkat kemudahan dalam membaca dan memahami sintaknya (Python Software Foundation, 2012). Bahasa ini mendukung beragam paradigma pemrograman, mencakup pemrograman berorientasi objek, pemrograman fungsional, serta pemrograman prosedural (Python Software Foundation, 2012). Kode menggunakan bahasa Python dapat ditulis dan dijalankan dengan menggunakan berbagai *platform*. Salah satu *platform* yang paling

banyak digunakan adalah Google Colaboratory yang dapat diakses secara *online*. Kelebihan Python terletak pada koleksi modul dan pustaka yang kuat, menjadikannya pilihan yang serbaguna untuk berbagai keperluan, mulai dari pengembangan web, analisis data, hingga pengaplikasian kecerdasan buatan. Beberapa modul dan pustaka yang digunakan dalam penelitian ini diantaranya adalah sebagai berikut.

1. TensorFlow

TensorFlow adalah suatu *end-to-end platform* yang dibuat ramah untuk pemula untuk membuat model *Machine Learning* baik pada *platform desktop, mobile, website*, maupun *cloud* (TensorFlow Developer, 2023). Kenton (2022) menyatakan bahwa *end-to-end platform* berarti *platform* atau program yang menyediakan alat untuk memproses sesuatu dari awal hingga akhir dan memberikan solusi fungsional yang lengkap tanpa menggunakan pihak ketiga.

TensorFlow terdiri dari berbagai macam modul pemrograman dengan fungsi yang beragam. Beberapa modul TensorFlow digunakan untuk prapemrosesan gambar dan membuat model CNN. Prapemrosesan gambar menggunakan modul *keras.preprocessing* sedangkan Model CNN dibuat dengan menggunakan modul Keras yang terdiri dari *models.sequential* dan *layers* dengan jenis Conv2D, MaxPooling2D, Flatten, dan Dense.

2. Matplotlib

Matplotlib adalah salah satu modul atau *library* pada Python untuk melakukan visualisasi data menggunakan Python (Tineges & Davita, 2021). Pada penelitian ini, Matplotlib digunakan untuk memvisualisasikan performa model yang didapat oleh model.

3. *Python Imaging Library* (PIL)

Python Imaging Library (PIL) atau yang biasa dikenal dengan Pillow adalah suatu modul atau *library* pada Python yang digunakan untuk Prapemrosesan gambar yang dibuat oleh Jeffrey A. Clark. Modul ini digunakan untuk pengarsipan, pemrosesan *batch*, menampilkan, dan memproses gambar (Clark, 2024).

4. Modul *csv*

Format CSV adalah format impor dan ekspor data yang paling umum digunakan untuk data file spreadsheet maupun database. Modul *csv* pada Python memungkinkan *interpreter* untuk melakukan penulisan dan pembacaan file dengan format CSV (Python Software Foundation, 2024).

BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

Objek pada penelitian ini adalah dataset gambar spesies bunga edelweis yang terdiri dari data *train* dan data *test*. Secara berurutan, data untuk *training* dan data untuk *test* terdiri dari 3498 dan 1050 gambar yang tersebar ke dalam tiga buah kelompok sesuai spesiesnya yaitu *Anaphalis javanica*, *Leontopodium alpinum*, dan *Leucogenes grandiceps*. Persebaran data *training* dan data *test* secara berturut-turut sebanyak 1166 dan 350 per spesies. Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari *website* Kaggle (Malau, 2022). Tautan dataset yang digunakan terdapat pada Lampiran 1.

3.2 Metode Penelitian

Metode yang digunakan dalam penelitian ini adalah *Convolutional Neural Network* dan menggunakan algoritma Adam sebagai pengoptimal untuk klasifikasi spesies edelweis menggunakan gambar dengan bahasa pemrograman Python. Pada penelitian ini, data diklasifikasikan ke dalam tiga kelas sesuai dengan jenis spesies pada dataset. Secara garis besar, langkah-langkah dalam penelitian ini terbagi ke dalam beberapa tahap dengan rincian sebagai berikut.

1. Prapemrosesan gambar

Prapemrosesan gambar dilakukan dengan melakukan kompresi gambar dengan mengecilkan ukuran *piksel* menjadi lebar sebesar 512px dan tinggi

menyesuaikan dengan rasio tiap gambar serta mengurangi kualitas gambar menjadi 85%. Kompresi dilakukan dengan tujuan untuk mempercepat proses komputasi dan menyeragamkan ukuran gambar. Kemudian prapemrosesan gambar dilanjutkan dengan membuat *image data generator*. Pada tahap ini, komponen warna pada gambar diubah menjadi angka dan dinormalisasi, serta penyeragaman ukuran menjadi (256,256). Selanjutnya data dibagi menjadi data *train* dan data *validation*. Rasio perbandingan antara data *train* dan data *validation* sebesar 8:2 dengan metode pemisahan yang dilakukan secara acak sehingga total data *train* sebanyak 2799 dan data *validation* sebanyak 699. Tahap terakhir dalam prapemrosesan gambar adalah dengan menyeragamkan ukuran gambar menjadi persegi dan membagi data *train* ke dalam *mini batch*. Diketahui bahwa total data *train* sebanyak 2799 dan *batch size* sebesar 32, sehingga dengan menggunakan persamaan 2.2 diperoleh:

$$T = \left\lceil \frac{2799}{32} \right\rceil = 88$$

dengan menggunakan persamaan 2.3, diperoleh:

$$\begin{aligned} Xtn^{\{1\}} &= \{X_1, X_2, \dots, X_{32}\} \\ Xtn^{\{2\}} &= \{X_{33}, X_{34}, \dots, X_{64}\} \\ &\vdots \\ Xtn^{\{t\}} &= \{X_{(32 \cdot (t-1)) + 1}, X_{(32 \cdot t) + 2}, \dots, X_{32 \cdot t}\}; t = 1, 2, \dots, T \end{aligned} \quad (3.2)$$

sehingga setelah dibagi ke dalam *mini batch*, maka dataset train menjadi sebagai berikut.

$$Xtn = \{Xtn^{\{1\}}, Xtn^{\{2\}}, \dots, Xtn^{\{88\}}\}$$

2. Inisiasi nilai *learning rate* sebesar 10^{-i} , dengan $i = 1$

3. *Training Model: forward propagation*

Sebelum dilakukan proses *training*, dilakukan pembuatan arsitektur model CNN seperti pada Lampiran 2. Proses *Training* model diawali dengan tahap *forward propagation* yang bertujuan untuk mendapatkan hasil klasifikasi beserta nilai *Loss* dan *Cost function*. Proses *forward propagation* diawali dengan memasukkan data gambar pertama dari *mini batch* pertama dari persamaan 3.2 ($Xtn_1^{\{1\}}$). Substitusi $Xtn_1^{\{1\}}$ ke persamaan 2.6 sehingga diperoleh:

$$zc^{(1)} = Conv(Xtn_1^{\{1\}}, W^{(1)}) + \beta^{(1)} \quad (3.4)$$

dimana dengan menggunakan persamaan 2.7 dan filter W sebanyak 16 filter yang berukuran (3, 3, 3) maka diperoleh:

$$\begin{aligned} Conv(Xtn_1^{\{1\}}, W^{(1)}) &= A_{q,r,s}^{(1)} \\ &= \sum_{c=1}^3 \sum_{a=1}^3 \sum_{b=1}^3 (x_1^{\{1\}})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(1)} \end{aligned} \quad (3.5)$$

substitusikan $zc^{(1)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{ReLU}(zc^{(1)}) = \max(0, zc^{(1)}) \quad (3.6)$$

pindahkan *feature map* $Xts_1^{\{1\}}$ sejauh 1 piksel kemudian ulangi langkah 3.4 sampai dengan langkah 3.6 sehingga diperoleh:

$$A^{(1)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.7)$$

dengan dimensi output sebesar (254, 254, 16). Substitusi $A^{(1)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(2)} = \text{MaxPooling2D}(\mathbf{A}^{(1)}) \quad (3.8)$$

dengan dimensi output sebesar (127, 127, 16) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(2)}$ ke persamaan 2.6, diperoleh:

$$z_c^{(3)} = \text{Conv}(\mathbf{P}^{(2)}, \mathbf{W}^{(3)}) + \beta^{(3)} \quad (3.9)$$

dimana dengan menggunakan persamaan 2.7 dan filter \mathbf{W} sebanyak 32 filter yang berukuran (3, 3, 16) maka diperoleh:

$$\begin{aligned} \text{Conv}(\mathbf{P}^{(2)}, \mathbf{W}^{(3)}) &= A_{q,r,s}^{(3)} \\ &= \sum_{c=1}^{16} \sum_{a=1}^3 \sum_{b=1}^3 (\mathbf{P}^{(2)})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(3)} \end{aligned} \quad (3.10)$$

substitusikan $z_c^{(3)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{\text{ReLU}}(z_c^{(3)}) = \max(0, z_c^{(3)}) \quad (3.11)$$

pindahkan *feature map* $\mathbf{P}^{(2)}$ sejauh 1 piksel kemudian ulangi langkah 3.9 sampai dengan langkah 3.11 sehingga diperoleh:

$$\mathbf{A}^{(3)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.12)$$

dengan dimensi output sebesar (125, 125, 32). Substitusi $\mathbf{A}^{(3)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(4)} = \text{MaxPooling2D}(\mathbf{A}^{(3)}) \quad (3.13)$$

dengan dimensi output sebesar (62, 62, 32) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(4)}$ ke persamaan 2.6, diperoleh:

$$zc^{(5)} = Conv(\mathbf{P}^{(4)}, \mathbf{W}^{(5)}) + \beta^{(5)} \quad (3.14)$$

dimana dengan menggunakan persamaan 2.7 dan filter \mathbf{W} sebanyak 64 filter yang berukuran (3, 3, 32) maka diperoleh:

$$\begin{aligned} Conv(\mathbf{P}^{(4)}, \mathbf{W}^{(5)}) &= A_{q,r,s}^{(5)} \\ &= \sum_{c=1}^{32} \sum_{a=1}^3 \sum_{b=1}^3 (\mathbf{P}^{(4)})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(5)} \end{aligned} \quad (3.15)$$

substitusikan $zc^{(5)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{ReLU}(zc^{(5)}) = \max(0, zc^{(5)}) \quad (3.16)$$

pindahkan *feature map* $\mathbf{P}^{(4)}$ sejauh 1 piksel kemudian ulangi langkah 3.14 sampai dengan langkah 3.16 sehingga diperoleh:

$$\mathbf{A}^{(5)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.17)$$

dengan dimensi output sebesar (60, 60, 64). Substitusi $\mathbf{A}^{(5)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(6)} = MaxPooling2D(\mathbf{A}^{(5)}) \quad (3.18)$$

dengan dimensi output sebesar (30, 30, 64) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(6)}$ ke persamaan 2.6, diperoleh:

$$zc^{(7)} = Conv(\mathbf{P}^{(6)}, \mathbf{W}^{(7)}) + \beta^{(7)} \quad (3.19)$$

dimana dengan menggunakan persamaan 2.7 dan filter \mathbf{W} sebanyak 128 filter yang berukuran (3, 3, 64) maka diperoleh:

$$Conv(\mathbf{P}^{(6)}, \mathbf{W}^{(7)}) = A_{q,r,s}^{(7)}$$

$$= \sum_{c=1}^{64} \sum_{a=1}^3 \sum_{b=1}^3 (\mathbf{P}^{(6)})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(7)} \quad (3.20)$$

substitusikan $zc^{(7)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{ReLU}(zc^{(7)}) = \max(0, zc^{(7)}) \quad (3.21)$$

pindahkan *feature map* $\mathbf{P}^{(6)}$ sejauh 1 piksel kemudian ulangi langkah 3.19 sampai dengan langkah 3.21 sehingga diperoleh:

$$\mathbf{A}^{(7)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.22)$$

dengan dimensi output sebesar (28, 28, 128). Substitusi $\mathbf{A}^{(7)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(8)} = \text{MaxPooling2D}(\mathbf{A}^{(7)}) \quad (3.23)$$

dengan dimensi output sebesar (14, 14, 128) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(8)}$ ke persamaan 2.6, diperoleh:

$$zc^{(9)} = \text{Conv}(\mathbf{P}^{(8)}, \mathbf{W}^{(9)}) + \beta^{(9)} \quad (3.24)$$

dimana dengan menggunakan persamaan 2.7 dan filter \mathbf{W} sebanyak 256 filter yang berukuran (3, 3, 128) maka diperoleh:

$$\begin{aligned} \text{Conv}(\mathbf{P}^{(8)}, \mathbf{W}^{(9)}) &= A_{q,r,s}^{(9)} \\ &= \sum_{c=1}^{128} \sum_{a=1}^3 \sum_{b=1}^3 (\mathbf{P}^{(8)})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(9)} \end{aligned} \quad (3.25)$$

substitusikan $zc^{(9)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{ReLU}(zc^{(9)}) = \max(0, zc^{(9)}) \quad (3.26)$$

pindahkan *feature map* $\mathbf{P}^{(8)}$ sejauh 1 piksel kemudian ulangi langkah 3.24 sampai dengan langkah 3.26 sehingga diperoleh:

$$\mathbf{A}^{(9)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.27)$$

dengan dimensi output sebesar (12, 12, 256). Substitusi $\mathbf{A}^{(9)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(10)} = \text{MaxPooling2D}(\mathbf{A}^{(9)}) \quad (3.28)$$

dengan dimensi output sebesar (6, 6, 256) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(10)}$ ke persamaan 2.6, diperoleh:

$$z_c^{(11)} = \text{Conv}(\mathbf{P}^{(10)}, \mathbf{W}^{(11)}) + \beta^{(11)} \quad (3.29)$$

dimana dengan menggunakan persamaan 2.7 dan filter \mathbf{W} sebanyak 256 filter yang berukuran (3, 3, 256) yang berukuran (3, 3, 256) maka diperoleh:

$$\begin{aligned} \text{Conv}(\mathbf{P}^{(10)}, \mathbf{W}^{(11)}) &= A_{q,r,s}^{(11)} \\ &= \sum_{c=1}^{256} \sum_{a=1}^3 \sum_{b=1}^3 (\mathbf{P}^{(10)})_{q+a-1, r+b-1, s+c-1} \cdot w_{a,b,c}^{(11)} \end{aligned} \quad (3.30)$$

Substitusikan $z_c^{(11)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.8 sehingga diperoleh:

$$A_{q,r,s} = \alpha_{\text{ReLU}}(z_c^{(11)}) = \max(0, z_c^{(11)}) \quad (3.31)$$

pindahkan *feature map* $\mathbf{P}^{(10)}$ sejauh 1 piksel kemudian ulangi langkah 3.29 sampai dengan langkah 3.31 sehingga diperoleh:

$$\mathbf{A}^{(11)} = \begin{bmatrix} [A_{1,1,1} \ A_{1,1,2} \ \dots \ A_{1,1,s}] & [A_{1,2,1} \ a_{1,2,2} \ \dots \ A_{1,2,s}] & \dots & [A_{1,r,1} \ A_{1,r,2} \ \dots \ a_{1,r,s}] \\ [A_{2,1,1} \ A_{2,1,2} \ \dots \ A_{2,1,s}] & [A_{2,2,1} \ A_{2,2,2} \ \dots \ A_{2,2,s}] & \dots & [A_{2,r,1} \ A_{2,r,2} \ \dots \ A_{2,r,s}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{q,1,1} \ A_{q,1,2} \ \dots \ A_{q,1,s}] & [A_{q,2,1} \ A_{q,2,2} \ \dots \ A_{q,2,s}] & \dots & [A_{q,r,1} \ A_{q,r,2} \ \dots \ A_{q,r,s}] \end{bmatrix} \quad (3.32)$$

dengan dimensi output sebesar (4, 4, 256). Substitusi $\mathbf{A}^{(11)}$ ke persamaan 2.10 sehingga diperoleh:

$$\mathbf{P}^{(12)} = \text{MaxPooling2D}(\mathbf{A}^{(11)}) \quad (3.33)$$

dengan dimensi output sebesar (2, 2, 256) dan akan menjadi input untuk *layer* selanjutnya sehingga substitusi $\mathbf{P}^{(12)}$ ke persamaan 2.11 diperoleh:

$$FL(\mathbf{P}^{(12)}) = \vec{zf} = [P_{1,1,1}^{(12)} P_{1,2,1}^{(12)} P_{2,1,1}^{(12)} P_{2,2,1}^{(12)} \dots P_{2,2,256}^{(12)}] \quad (3.34)$$

substitusi \vec{zf}_k dengan $k = 1, 2, \dots, (2 \cdot 2 \cdot 256)$ ke persamaan 2.12 sehingga diperoleh.

$$zd_j^{(14)} = \sum_{k=1}^{1024} w_{jk}^{(14)} \cdot \vec{zf}_k + \beta_j^{(14)} \quad (3.35)$$

substitusi $zd_j^{(14)}$ ke fungsi aktivasi ReLU menggunakan persamaan 2.13 dan persamaan 2.15 sehingga diperoleh.

$$d_j = \alpha_{ReLU}(zd_j^{(14)}) = \max(0, zd_j^{(14)}) \quad (3.36)$$

sehingga diperoleh output dari *dense layer* pada *layer* ke-14 adalah sebagai berikut

$$\vec{d}^{(14)} = [d_1 \ d_2 \ \dots \ d_{128}] \quad (3.37)$$

substitusi $\vec{d}^{(14)}$ ke persamaan 2.12 sehingga diperoleh.

$$zd_j^{(15)} = \sum_{k=1}^{128} w_{jk}^{(15)} \cdot d_k^{(14)} + \beta_j^{(15)} \quad (3.38)$$

substitusi $zd_j^{(15)}$ ke persamaan 2.13 dan 2.16 sehingga diperoleh.

$$d_j = \alpha_{sm}(zd_j^{(15)}) = \frac{e^{z_j^{(15)}}}{\sum_{j=1}^n e^{z_j^{(15)}}} \quad (3.39)$$

sehingga diperoleh output dari *dense layer* pada *layer* ke 15 adalah sebagai berikut.

$$\vec{d}^{(15)} = [d_1 \ d_2 \ d_3] \quad (3.40)$$

substitusi d_j dari persamaan 3.41 ke *loss function* pada persamaan 2.17 diperoleh:

$$Loss(\vec{d}^{(15)}, \vec{y}) = -\sum_{j=1}^3 y_j \cdot \log(d_j) \quad (3.41)$$

lakukan langkah 3.4 hingga 3.41 menggunakan $X_p^{\{1\}}$ dimana $p = 2, 3, \dots, 32$ sehingga diperoleh:

$$Loss^{(p)}(\vec{d}^{(15)}, \vec{y}) = -\sum_{j=1}^3 y_j^{(p)} \cdot \log(d_j^{(p)}) \quad (3.42)$$

4. Training model: *backpropagation*

Backpropagation diawali dengan menghitung nilai *cost function* dari hasil klasifikasi pada tahap *forward propagation*. Hitung *cost function* dengan substitusi $Loss^{(p)}, p = 1, 2, \dots, 32$ ke persamaan 2.18 diperoleh:

$$J = \frac{1}{32} \sum_{p=1}^{32} Loss(d_j^{(p)}, y_j^{(p)}) \quad (3.43)$$

Setelah mendapatkan nilai *cost function*, hitung gradien J terhadap parameter model yaitu *weight* dan *bias*. Dari proses *forward propagation* dengan memperhatikan persamaan 2.6 dan persamaan 2.12 didapat:

$$zc^{(1)} = Conv(X_1^{\{1\}}, W^{(1)}) + \beta^{(1)} \quad (3.44)$$

$$zc^{(3)} = Conv(P^{(2)}, W^{(3)}) + \beta^{(3)} \quad (3.45)$$

$$zc^{(5)} = Conv(P^{(4)}, W^{(5)}) + \beta^{(5)} \quad (3.46)$$

$$zc^{(7)} = Conv(P^{(6)}, W^{(7)}) + \beta^{(7)} \quad (3.47)$$

$$zc^{(9)} = Conv(\mathbf{P}^{(8)}, \mathbf{W}^{(9)}) + \beta^{(9)} \quad (3.48)$$

$$zc^{(11)} = Conv(\mathbf{P}^{(10)}, \mathbf{W}^{(11)}) + \beta^{(11)} \quad (3.49)$$

$$zd_j^{(14)} = \sum_{k=1}^{1024} w_{jk}^{(14)} \cdot \vec{zf}_k + \beta_j^{(14)} \quad (3.50)$$

$$zd_j^{(15)} = \sum_{k=1}^{128} w_{jk}^{(15)} \cdot d_k^{(14)} + \beta_j^{(15)} \quad (3.51)$$

dari persamaan 3.44 hingga persamaan 3.51 didapat parameter-parameter dari model yang dapat dilatih adalah sebagai berikut:

$$\vec{\mu} = [\mathbf{W}^{(1)} \beta^{(1)} \mathbf{W}^{(3)} \beta^{(3)} \mathbf{W}^{(5)} \beta^{(5)} \mathbf{W}^{(7)} \beta^{(7)} \mathbf{W}^{(9)} \beta^{(9)} \mathbf{W}^{(11)} \beta^{(11)} w_{jk}^{(14)} \beta^{(14)} w_{jk}^{(15)} \beta^{(15)}] \quad (3.52)$$

Dari persamaan 3.43 dan persamaan 3.52, substitusi $\vec{\mu}_l$ dimana $l =$

1,2, ...,16 ke persamaan 2.19 sehingga didapat:

$$g_t = \frac{d}{d\vec{\mu}_l} J(\vec{\mu}_l) \quad (3.53)$$

kemudian gunakan persamaan 2.19 hingga persamaan 2.24 untuk memperbaharui nilai μ_l sehingga didapat:

$$(\vec{\mu}_l)_t = (\vec{\mu}_l)_{t-1} - lr_i \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (3.54)$$

Proses pembaharuan parameter w dan β dilakukan di setiap *mini batch*. Setelah seluruh *mini batch* telah selesai, maka proses *training* model dilanjutkan dengan menguji model dengan menggunakan data *validation* untuk mengetahui akurasi model untuk mengklasifikasi data diluar dataset yang digunakan untuk *training*. Proses pengujian menggunakan data *validation* dilakukan hanya hingga proses *forward propagation* dengan hasil akhir berupa nilai akurasi dan *loss*. *Dataset validation* memiliki total data sebanyak 699 gambar dengan *batch size* 32, substitusi ke persamaan 2.2 diperoleh.

$$T = \left\lceil \frac{699}{32} \right\rceil = 22$$

sehingga *mini batch* dari *dataset validation* dapat dinyatakan sebagai berikut.

$$Xval = \{X^{t\}}; t = 1, 2, \dots, T$$

dimana

$$Xval^{t\} = \{X_{(32 \cdot (t-1)) + 1}, X_{(32 \cdot t) + 2} \dots, X_{(t) \cdot 32}\} \quad (3.55)$$

dari $Xval^{t\}$, $t = 1, 2, \dots, T$ dengan substitusi nilai $Xtn = Xval$ pada persamaan 3.5 diperoleh.

$$zc^{(1)} = Conv(Xval_1^{1\}, W^{(1)}) + \beta^{(1)} \quad (3.56)$$

cari nilai *Cost function* dengan melanjutkan perhitungan hingga langkah ke 3.43 dan dicari nilai akurasi.

Kecil nya nilai *loss* dan tinggi nya akurasi menjadi penanda bahwa performa model baik. Proses *training* hingga pengujian menggunakan data *validation* dilakukan secara berulang hingga 20 *epoch*.

5. *Testing model*

Setelah diperoleh model terbaik dari proses *training* sebanyak 20 *epoch*. Model yang disimpan pada *epoch* terakhir kemudian diuji menggunakan dataset *test*. *Dataset test* diubah menjadi *mini batch* dan dimasukkan ke dalam model untuk dilakukan proses klasifikasi. *Dataset test* memiliki total data sebanyak 1050 gambar dengan *batch size* 32, substitusi ke persamaan 2.2 diperoleh.

$$T = \left\lceil \frac{1050}{32} \right\rceil = 33$$

sehingga *mini batch* dari *dataset test* dapat dinyatakan sebagai berikut.

$$Xts = \{X^{t\}}; t = 1, 2, \dots, T$$

dimana

$$Xts^{\{t\}} = \left\{ \mathbf{X}_{(32 \cdot (t-1)) + 1}, \mathbf{X}_{(32 \cdot t) + 2} \dots, \mathbf{X}_{(t) \cdot 32} \right\} \quad (3.57)$$

dari $Xts^{\{t\}}, t = 1, 2, \dots T$ dengan substitusi nilai $Xtn = Xts$ pada persamaan 3.5 diperoleh.

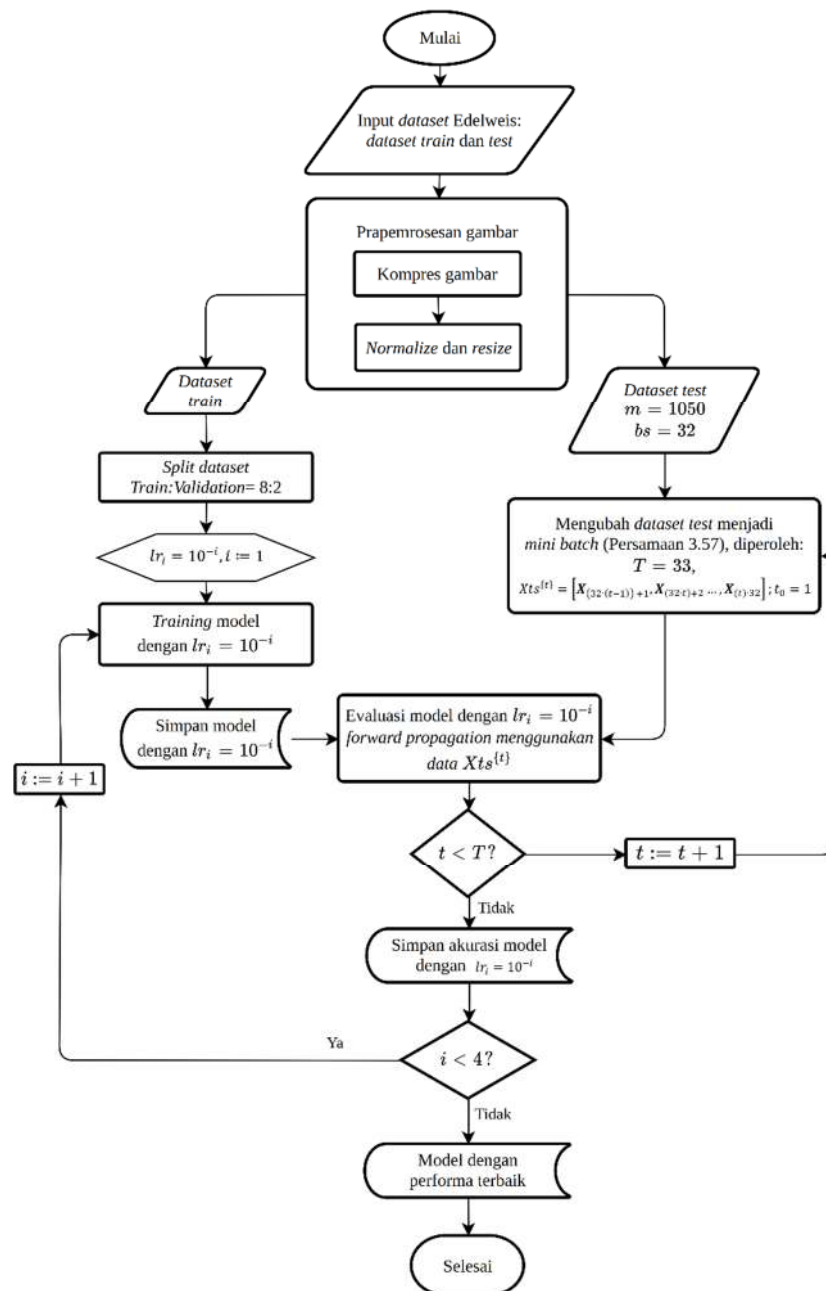
$$zc^{(1)} = Conv\left(\mathbf{Xts}_1^{\{1\}}, \mathbf{W}^{(1)}\right) + \beta^{(1)} \quad (3.58)$$

cari nilai *Cost function* dengan melanjutkan perhitungan hingga langkah ke 3.43. Kemudian cari nilai akurasi dengan menggunakan persamaan 2.25.

6. Perbarui nilai i pada *learning rate* sehingga $i = i + 1$.
7. Ketika $i > 4$, proses *training* model berhenti.

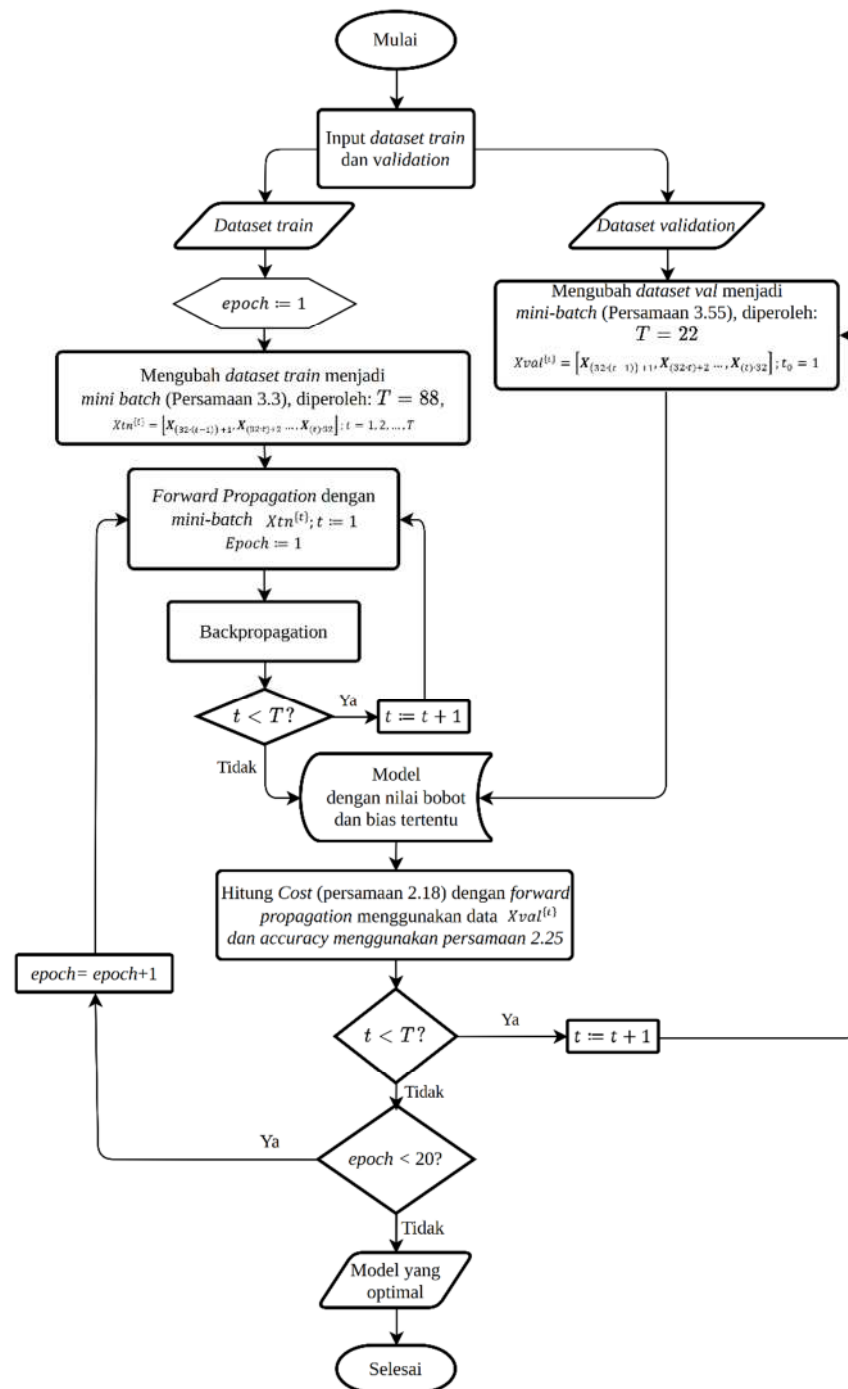
3.3 Diagram Alir Penelitian

Diagram alir penelitian klasifikasi spesies bunga edelweis menggunakan *Convolutional Neural Network* dan pengoptimal Adam terdapat pada Gambar 3.1.



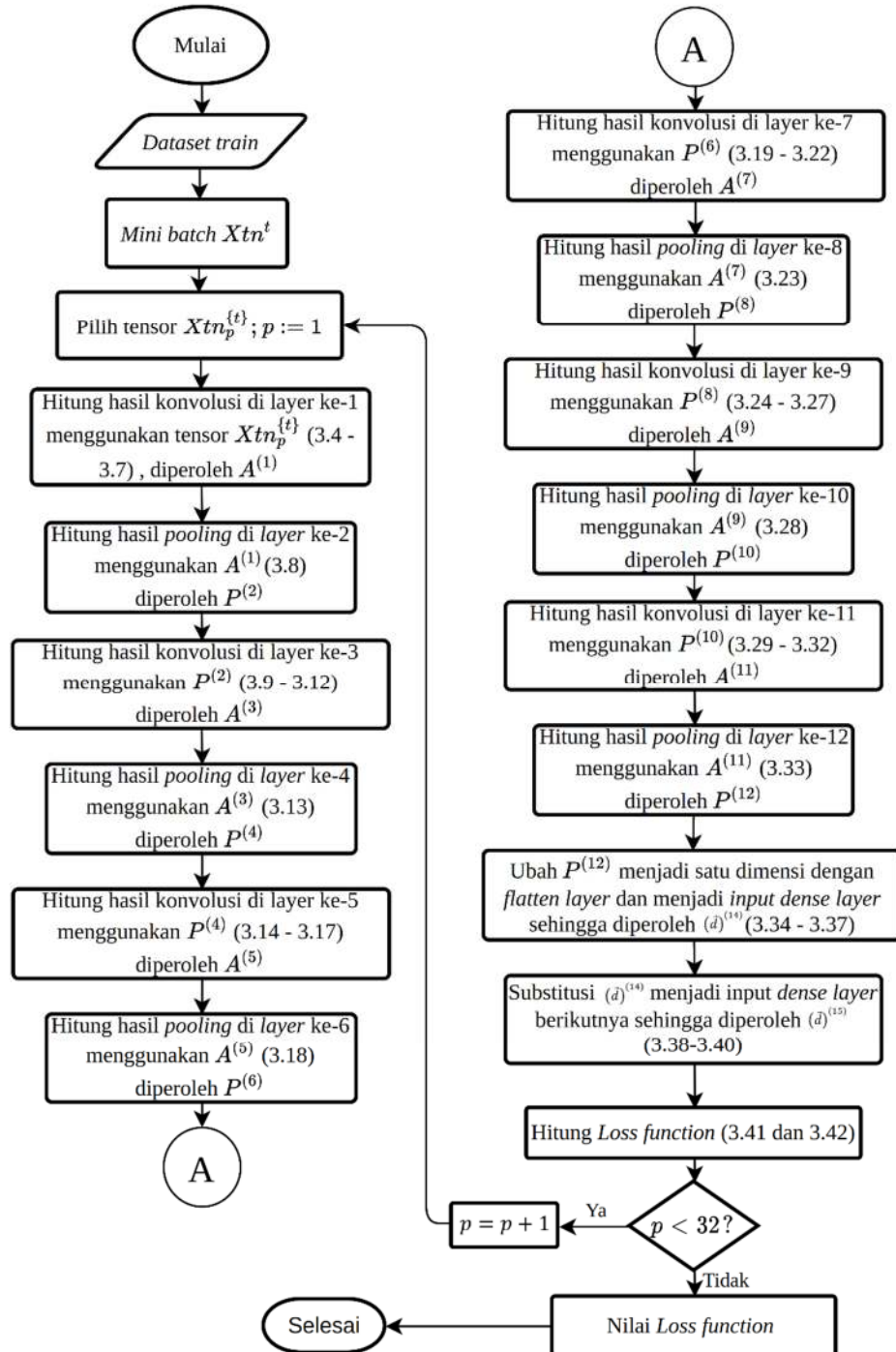
Gambar 3.1 Diagram alir penelitian

Diagram alir untuk *training* model dapat dilihat pada Gambar 3.2.



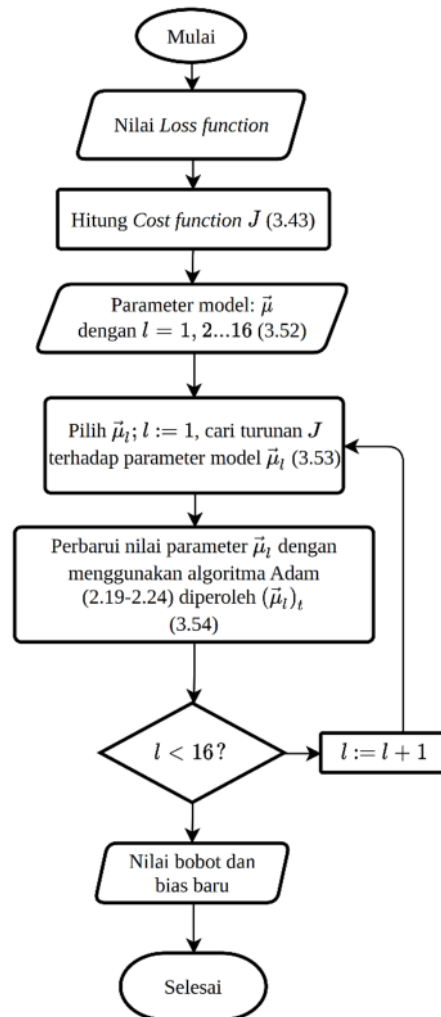
Gambar 3.2 Diagram alir tahap *training* model

Diagram alir pada tahap *forward propagation* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram alir pada tahap *forward propagation*

Diagram alir pada tahap *backpropagation* dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram alir pada tahap *backpropagation*

BAB IV

HASIL DAN PEMBAHASAN

4.1 Prapemrosesan Gambar

Sebelum dataset gambar digunakan untuk melatih model, dilakukan prapemrosesan gambar. Prapemrosesan gambar yang pertama kali dilakukan adalah melakukan kompresi gambar dengan mengurangi ukuran gambar menjadi maksimal 512px dan mengurangi kualitas gambar hingga 85% dengan menggunakan *library* PIL. Contoh perbedaan gambar saat sebelum dan sesudah dikompresi dapat dilihat pada Gambar 4.1.



Gambar 4.1 Perubahan ukuran dan kualitas gambar

(a) Gambar asli sebelum dikompresi

(b) Gambar setelah dilakukan kompresi

Gambar (a) merupakan gambar asli dengan dimensi 2048px × 1152px dengan ukuran 3,5MB sedangkan gambar (b) merupakan gambar setelah proses kompresi sehingga dimensinya berkurang menjadi 512px × 288px dengan kualitas ukuran 289KB. Perhitungan dari nilai lebar gambar dilakukan sebagai berikut.

1. Inisiasi lebar gambar baru sebesar 512px
2. Cari rasio panjang berbanding lebar

$$rasio = \frac{2048px}{1152px} = 1,7777$$

3. Hitung tinggi baru dengan membagi tinggi asli dengan rasio

$$new\ height = \frac{512px}{1,77} = 288px$$

Prapemrosesan gambar dilanjutkan dengan melakukan *resize* menjadi (256,256) dan mengubah gambar menjadi tensor dengan menggunakan *syntax flow_from_directory* dari modul ImageDataGenerator.



(a)

(b)

Gambar 4.2 Perbedaan gambar setelah dan sebelum dilakukan *resize*

(a) Gambar sebelum dilakukan *resize*

(b) Gambar setelah dilakukan *resize*

Pada Gambar 4.2 terlihat bahwa setelah dilakukan *resize*, ukuran gambar berubah menjadi persegi dengan dimensi (256,256). Proses selanjutnya yaitu dengan mengubah gambar menjadi tensor dan kemudian dilakukan Normalisasi menggunakan modul ImageDataGenerator. Berikut nilai tensor sebelum dilakukan normalisasi.

$$X = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} \square & \dots & \square \end{bmatrix} & \begin{bmatrix} 5 & 8 & 5 \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} \dots & [11 & 13 & 10] & \dots & \begin{bmatrix} 5 & 7 & 2 \end{bmatrix} \end{bmatrix} \\ \vdots & \vdots & \vdots \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} \square & \dots & \square \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

setelah dilakukan normalisasi menggunakan persamaan 2.1, maka tensor tersebut berubah menjadi sebagai berikut.

Normalisasi(X)

$$\begin{aligned}
 &= \text{Normalisasi} \left(\begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [5 & 8 & 5] \\ \vdots & \square & \square & \vdots \\ [0 & 0 & 0] & \dots & [11 & 13 & 10] & \dots & [5 & 7 & 2] \\ \vdots & \square & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix} \right) \\
 &= \frac{1}{255} \begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [5 & 8 & 5] \\ \vdots & \square & \square & \vdots \\ [0 & 0 & 0] & \dots & [11 & 13 & 10] & \dots & [5 & 7 & 2] \\ \vdots & \square & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix} \\
 &= \begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [0.019 & 0.031 & 0.019] \\ \vdots & \square & \square & \vdots \\ [0 & 0 & 0] & \dots & [0.043 & 0.051 & 0.039] & \dots & [0.019 & 0.027 & 0.008] \\ \vdots & \square & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix}
 \end{aligned} \tag{4.1}$$

Prapemrosesan gambar kemudian dilanjutkan dengan membagi dataset train yang sudah dirubah ke dalam bentuk tensor menjadi dataset *training* dan dataset *validation*. Dataset *training* kemudian dibagi ke dalam *mini batch* dengan ukuran *batch size* sebesar 32. Artinya setiap satu *mini batch* mengandung sebanyak 32 tensor sebagai representasi dari sebanyak 32 gambar dalam dataset. Dengan menggunakan persamaan 3.2 diperoleh:

$$Xtn^{\{t\}} = [X_{(32 \cdot (t-1)) + 1} \ X_{(32 \cdot t) + 2} \ \dots \ X_{32 \cdot t}]; t = 1, 2, \dots, 88$$

pilih $t = 1$, sehingga diperoleh:

$$Xtn^{\{1\}} = [X_1 \ X_2 \ \dots \ X_{32}]$$

$$\begin{aligned}
& \mathbf{Xtn}^{\{1\}} \\
& = \left\{ \begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [0.019 & 0.031 & 0.019] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0 & 0 & 0] & \dots & [0.043 & 0.051 & 0.039] & \dots & [0.019 & 0.027 & 0.008] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix}, \right. \\
& \quad \left. \begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0.255 & 0.247 & 0.217] & \dots & [0.231 & 0.216 & 0.192] & \dots & [0 & 0 & 0] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0.474 & 0.486 & 0.412] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix}, \dots, \right. \\
& \quad \left. \begin{bmatrix} [0.114 & 0.145 & 0.094] & \square & \dots & \square & [0.196 & 0.353 & 0.149] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0.094 & 0.149 & 0.09] & \dots & [0.059 & 0.074 & 0.031] & \dots & [0.031 & 0.062 & 0.016] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix} \right\}
\end{aligned}$$

4.2 Training Model

Proses *training* model dengan nilai *learning rate* pertama yaitu 10^{-i} dimana $i = 1$ yang dilakukan sebanyak 20 epoch. Setiap *epoch* memproses seluruh *mini batch* melalui proses *forward propagation* dan proses *backpropagation* untuk mendapatkan nilai Cost dan memperbarui parameter model. Dari persamaan 4.1 pilih tensor $\mathbf{Xtn}_1^{\{1\}}$.

$$\begin{aligned}
& \mathbf{Xtn}_1^{\{1\}} \\
& = \begin{bmatrix} [0 & 0 & 0] & \square & \dots & \square & [0.019 & 0.031 & 0.019] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0 & 0 & 0] & \dots & [0.043 & 0.051 & 0.039] & \dots & [0.019 & 0.027 & 0.008] \\ \vdots & \square & \square & \vdots & \vdots & \vdots & \vdots \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{bmatrix} \quad (4.2)
\end{aligned}$$

substitusi $\mathbf{Xtn}_1^{\{1\}}$ ke *convolution layer* pertama (persamaan 3.4) diperoleh:

$$z^{(1)} = \text{Conv}(\mathbf{Xtn}_1^{\{1\}}, \mathbf{W}^{(1)}) + \beta^{(1)}$$

pilih filter pertama dari $\mathbf{W}^{(1)}$ sehingga diperoleh:

$$\begin{aligned}
zc^{(1)} &= \begin{bmatrix} [0 & 0 & 0] & \begin{bmatrix} \square & \dots & \square \end{bmatrix} & [0.019 & 0.031 & 0.019] \\ \vdots & \begin{bmatrix} \square \end{bmatrix} & \vdots \\ [0 & 0 & 0] & \dots & [0.043 & 0.051 & 0.039] & \dots & [0.019 & 0.027 & 0.008] \\ \vdots & \begin{bmatrix} \square \end{bmatrix} & \vdots \\ [0 & 0 & 0] & \begin{bmatrix} \square & \dots & \square \end{bmatrix} & [0 & 0 & 0] \end{bmatrix} * \\
&\begin{bmatrix} [0.0378 & 0.1227 & 0.1058] & [-0.0015 & -0.1656 & 0.0132] & [0.1793 & -0.1849 & 0.0517] \\ [0.0793 & 0.1721 & -0.0265] & [0.1311 & -0.0322 & -0.0376] & [0.1518 & 0.0552 & 0.0821] \\ [0.1406 & -0.0945 & -0.0474] & [-0.1018 & 0.1746 & -0.1126] & [0.1058 & -0.0129 & 0.0864] \end{bmatrix} + \\
&0 \tag{4.3}
\end{aligned}$$

pilih *input feature map* posisi 1 dari $\mathbf{Xtn}_1^{\{1\}}$, lakukan operasi konvolusi dengan menggunakan persamaan 3.5 sehingga diperoleh:

$$\begin{aligned}
A_{q,r,s}^{(1)} &= \sum_{c=1}^3 \sum_{a=1}^3 \sum_{b=1}^3 (x_1^{\{1\}})_{q+a-1,r+b-1,s+c-1} \cdot w_{a,b,c}^{(1)} \\
A_{1,1,1}^{(1)} &= (x_1^{\{1\}})_{1,1,1} \cdot w_{1,1,1} + (x_1^{\{1\}})_{1,2,1} \cdot w_{1,2,1} + (x_1^{\{1\}})_{1,3,1} \cdot w_{1,3,1} \\
&+ (x_1^{\{1\}})_{2,1,1} \cdot w_{2,1,1} + (x_1^{\{1\}})_{2,2,1} \cdot w_{2,2,1} + (x_1^{\{1\}})_{2,3,1} \cdot w_{2,3,1} \\
&+ (x_1^{\{1\}})_{3,1,1} \cdot w_{3,1,1} + (x_1^{\{1\}})_{3,2,1} \cdot w_{3,2,1} + (x_1^{\{1\}})_{3,3,1} \cdot w_{3,3,1} \\
&+ (x_1^{\{1\}})_{1,1,2} \cdot w_{1,1,2} + (x_1^{\{1\}})_{1,2,2} \cdot w_{1,2,2} + (x_1^{\{1\}})_{1,3,2} \cdot w_{1,3,2} \\
&+ (x_1^{\{1\}})_{2,1,2} \cdot w_{2,1,2} + (x_1^{\{1\}})_{2,2,2} \cdot w_{2,2,2} + (x_1^{\{1\}})_{2,3,2} \cdot w_{2,3,2} \\
&+ (x_1^{\{1\}})_{3,1,2} \cdot w_{3,1,2} + (x_1^{\{1\}})_{3,2,2} \cdot w_{3,2,2} + (x_1^{\{1\}})_{3,3,2} \cdot w_{3,3,2} \\
&+ (x_1^{\{1\}})_{1,1,3} \cdot w_{1,1,3} + (x_1^{\{1\}})_{1,2,3} \cdot w_{1,2,3} + (x_1^{\{1\}})_{1,3,3} \cdot w_{1,3,3} \\
&+ (x_1^{\{1\}})_{2,1,3} \cdot w_{2,1,3} + (x_1^{\{1\}})_{2,2,3} \cdot w_{2,2,3} + (x_1^{\{1\}})_{2,3,3} \cdot w_{2,3,3} \\
&+ (x_1^{\{1\}})_{3,1,3} \cdot w_{3,1,3} + (x_1^{\{1\}})_{3,2,3} \cdot w_{3,2,3} + (x_1^{\{1\}})_{3,3,3} \cdot w_{3,3,3} \\
A_{1,1,1}^{(1)} &= 0 * 0.0378 + 0.149 * (-0.0015) + 0.1921 * 0.1793
\end{aligned}$$

$$\begin{aligned}
&+0 * 0.0793 + 0.0784 * 0.1311 + 0.1137 * 0.1518 \\
&+0 * 0.1406 + 0.0588 * -0.1018 + 0.1686 * 0.1058 \\
&+0 * 0.1227 + 0.149 * (-0.1656) + 0.1843 * (-0.1849) \\
&+0 * 0.1721 + 0.0863 * (-0.0322) + 0.1098 * 0.0552 \\
&+0 * (-0.0945) + 0.0667 * 0.1746 + 0.1647 * (-0.0129) \\
&+0 * 0.1058 + 0.1372 * 0.0132 + 0.1647 * 0.0517 \\
&+0 * (-0.0265) + 0.0823 * (-0.037) + 0.1058 * 0.0821 \\
&+0 * (-0.0474) + 0.0588 * (-0.1126) + 0.1607 * 0.0864
\end{aligned}$$

$$A_{1,1,1}^{(1)} = 0.050864193588$$

$$\text{sehingga } zc^{(1)} = A_{1,1,1}^{(1)} + \beta^{(1)} = 0.050864193588 \quad (4.4)$$

substitusi nilai $zc^{(1)}$ ke persamaan 3.7 sehingga diperoleh:

$$A_{1,1,1}^{(1)} = \max(0; 0.050864193588) = 0.050864193588 \quad (4.5)$$

pindahkan *feature map* $Xtn_1^{\{1\}}$ sebanyak satu piksel kemudian ulangi operasi

konvolusi hingga *feature map* terletak pada posisi terakhir sehingga diperoleh:

$$A^{(1)} = \begin{bmatrix} [0.0508 \ 0.0728 \ \dots \ 0] & \boxed{} & \dots & \boxed{} & [0.0100 \ 0 \ \dots \ 0] \\ & \vdots & & \vdots & \\ [0.1324 \ 0.1239 \ \dots \ 0] & \dots & [0.1778 \ 0.0508 \ \dots \ 0] & \dots & [0 \ 0.2147 \ \dots \ 0.03042] \\ & \vdots & & \ddots & \vdots \\ [0.0315 \ 0.0016 \ \dots \ 0] & \boxed{} & \dots & \boxed{} & [0 \ 0 \ \dots \ 0] \end{bmatrix} \quad (4.6)$$

Diperoleh output dari *convolution layer* pertama adalah $A^{(1)}$ dengan dimensi (124, 124, 16). Output layer pertama akan menjadi output layer selanjutnya yaitu *pooling layer*. Berikut adalah hasil dari *pooling layer*. Dari persamaan 3.8, didapat output pooling layer adalah sebagai berikut.

$$P^{(2)} = \begin{bmatrix} [0.1743 \ 0.1133 \dots 0] & \square & \dots & \square & [0.1863 \ 0.0754 \dots 0] \\ \vdots & \square & \square & \square & \vdots \\ [0.1598 \ 0.1672 \dots 0] & \dots & [0.2419 \ 0.094 \dots 0] & \dots & [0.2385 \ 0.1094 \dots 0] \\ \vdots & \square & \square & \ddots & \vdots \\ [0.0095 \ -0.0218 \dots -0.0048] & \square & \dots & \square & [0 \ 0 \dots 0] \end{bmatrix} \quad (4.7)$$

Proses *feature extraction* menggunakan *convolution layer* dan *pooling layer* terus berlangsung hingga *layer* ke-12 dengan seluruh proses perhitungan yang analog dengan perhitungan pada persamaan 4.2 hingga persamaan 4.7. Output *layer* ke-12 dengan menggunakan persamaan 3.33 adalah sebagai berikut.

$$P^{(12)} = \begin{bmatrix} [0 \ \dots \ 0.0081 \ \dots \ 0.0313] & [0 \ \dots \ 0.0275 \ \dots \ 0] \\ [0 \ \dots \ 0.0108 \ \dots \ 0] & [0 \ \dots \ 0.0184 \ \dots \ 0] \end{bmatrix}$$

substitusi $P^{(12)}$ ke persamaan 3.35 sehingga diperoleh:

$$\begin{aligned} \vec{zf} &= [0 \ \dots \ 0.0081 \ \dots \ 0.0313 \ 0 \ \dots \ 0.0275 \ \dots \ 0 \\ &\quad 0 \ \dots \ 0.0108 \ \dots \ 0 \ 0 \ \dots \ 0.0184 \ \dots \ 0] \end{aligned} \quad (4.7)$$

substitusi \vec{zf}_k dengan $k = 1, 2, \dots, 1024$ ke persamaan 3.35 sehingga diperoleh:

$$\begin{aligned} zd_1^{(14)} &= \sum_{k=1}^{1024} w_{1k}^{(14)} \cdot \vec{zf}_k + \beta_1^{(14)} \\ &= \left(w_{11}^{(14)} \cdot \vec{zf}_1 + w_{12}^{(14)} \cdot \vec{zf}_2 + w_{13}^{(14)} \cdot \vec{zf}_3 + \dots + w_{11024}^{(14)} \cdot \vec{zf}_{1024} \right) + 0 \\ &= (-0.0295 \cdot 0 + (-0.0603 \cdot 0) + (-0.0349 \cdot 0.0081) + \dots + 0 \cdot 0) + 0 \\ &= 0.087312981486 \end{aligned} \quad (4.8)$$

substitusi nilai $zd_1^{(14)}$ ke persamaan 3.36 sehingga diperoleh:

$$\begin{aligned} d_1 &= \max(0, zd_1^{(14)}) \\ d_1 &= \max(0, 0.087312981486) = 0.087312981486 \end{aligned} \quad (4.9)$$

lakukan perhitungan hingga $zd_{128}^{(14)}$ sehingga diperoleh $d_j, j = 2, 3, \dots, 128$. Dengan memperhatikan persamaan 3.37, maka diperoleh:

$$\begin{aligned}\vec{d}^{(14)} &= [d_1 \ d_2 \ d_3 \ d_4 \ \dots \ d_{128}] \\ \vec{d}^{(14)} &= [0.0873 \ 0 \ 0 \ 0.0071 \ \dots \ 0]\end{aligned}\quad (4.10)$$

substitusi $\vec{d}^{(14)}$ ke persamaan 3.38 sehingga diperoleh:

$$\begin{aligned}zd_j^{(15)} &= \sum_{k=1}^{128} w_{jk}^{(15)} \cdot d_k^{(14)} + \beta_j^{(15)} \\ zd_1^{(15)} &= \sum_{k=1}^{128} w_{1k}^{(15)} \cdot d_k^{(14)} + \beta_1^{(15)} \\ zd_2^{(15)} &= \sum_{k=1}^{128} w_{2k}^{(15)} \cdot d_k^{(14)} + \beta_2^{(15)} \\ zd_3^{(15)} &= \sum_{k=1}^{128} w_{3k}^{(15)} \cdot d_k^{(14)} + \beta_3^{(15)}\end{aligned}$$

dengan proses perhitungan yang analog pada persamaan 4.8 hingga persamaan 4.10 maka didapat:

$$\vec{zd}^{(15)} = [-0.027052525431 \ 0.016233012080 \ -0.016346495599] \quad (4.11)$$

substitusi ke fungsi aktivasi Softmax pada persamaan 3.40 sehingga diperoleh:

$$\begin{aligned}d_j &= \alpha_{sm}(zd_j^{(15)}) = \frac{e^{z_j^{(15)}}}{\sum_{j=1}^n e^{z_j^{(15)}}} \\ d_1 &= \alpha_{sm}(zd_1^{(15)}) \\ &= \frac{e^{z_1^{(15)}}}{e^{z_1^{(15)}} + e^{z_2^{(15)}} + e^{z_3^{(15)}}}\end{aligned}$$

$$= \frac{e^{-0.027052525431}}{e^{-0.027052525431} + e^{0.016233012080} + e^{-0.016346495599}}$$

$$d_1 = 0.327332288027 \quad (4.12)$$

$$d_2 = \alpha_{sm}(zd_2^{(15)})$$

$$= \frac{e^{z_2^{(15)}}}{e^{z_1^{(15)}} + e^{z_2^{(15)}} + e^{z_3^{(15)}}}$$

$$= \frac{e^{0.016233012080}}{e^{-0.027052525431} + e^{0.016233012080} + e^{-0.016346495599}}$$

$$d_2 = 0.341812163591 \quad (4.12)$$

$$d_3 = \alpha_{sm}(zd_3^{(15)})$$

$$= \frac{e^{z_3^{(15)}}}{e^{z_1^{(15)}} + e^{z_2^{(15)}} + e^{z_3^{(15)}}}$$

$$= \frac{e^{-0.016346495599}}{e^{-0.027052525431} + e^{0.016233012080} + e^{-0.016346495599}}$$

$$d_3 = 0.330855548382 \quad (4.12)$$

substitusi nilai d_1 , d_2 , dan d_3 ke persamaan 3.40 sehingga diperoleh:

$$\vec{d}^{(15)} = [0.327332288027 \ 0.341812163591 \ 0.330855548382] \quad (4.13)$$

Output dari perhitungan lengkap tahap *forward propagation* dapat dilihat pada Lampiran 7. Dari dataset diketahui bahwa data gambar tersebut memiliki label *Anaphalis javanica* sehingga diperoleh:

$$\vec{y} = [1 \ 0 \ 0]$$

proses selanjutnya adalah menghitung nilai *loss function* dengan mensubstitusi $\vec{d}^{(15)}$ dan \vec{y} ke persamaan 3.41 diperoleh:

$$Loss(\vec{d}^{(15)}, \vec{y}) = -\sum_{j=1}^3 y_j \cdot \log(d_j) \quad (4.14)$$

$$\begin{aligned}
&= -(1 \cdot \log(0.327332288027) + 0 \cdot \log(0.3418) + 0 \cdot \log(0.33085)) \\
&= -\log(0.327332288027) \\
&= 0.4850111538
\end{aligned}$$

pilih tensor selanjutnya dari *mini batch* $Xtn^{\{1\}}$ yaitu $Xtn_p^{\{1\}}$ dimana $p = 2, 3, \dots, 32$ kemudian lakukan proses *forward propagation* dari awal hingga mendapatkan nilai *loss function* dari setiap hasil klasifikasi tensor pada *mini batch* $Xtn^{\{1\}}$ yaitu $Loss^{(p)}(\vec{d}^{(15)}, \vec{y})$.

Setelah seluruh tensor pada *mini batch* $Xtn^{\{1\}}$ telah melewati proses *forward propagation*, maka *training* model dilanjutkan dengan melakukan *backpropagation*. *Backpropagation* diawali dengan menghitung nilai *cost function* dengan menggunakan persamaan 3.43 sehingga diperoleh:

$$\begin{aligned}
J &= \frac{1}{32} \sum_{p=1}^{32} Loss(d_j^{(p)}, y_j^{(p)}) \\
&= \frac{1}{28} (1.00781 + 1.00781 + 1.00781 + 1.17536 + 1.12 + \dots + 1.175 \\
&\quad + 1.0078) \\
&= 1.1193
\end{aligned} \tag{4.15}$$

Setelah mendapatkan nilai *cost function*, proses *backpropagation* dilanjutkan dengan mencari turunan dari *cost function* terhadap setiap parameter model. Dari persamaan 3.52, pilih $\vec{\mu}_1 = \beta^{(15)}$.

$$\beta^{(15)} = [0.000099999997 - 0.000099999997 - 0.000099999997] \tag{4.16}$$

Pilih $\beta_1^{(15)} = 0.000099999997$. Dari persamaan 3.57, diperoleh nilai gradien dari parameter bias pada *layer* ke-15 pada iterasi *mini batch* pertama sebagai berikut:

$$g_1 = \frac{d}{d\beta_1^{(15)}} J(\beta_1^{(15)})$$

dengan menggunakan fungsi `tf.GradientTape()`, diperoleh nilai turunan *cost function* terhadap $\beta_1^{(15)}$ adalah sebagai berikut:

$$g_1 = -5.607854843140$$

substitusi g_1 ke persamaan 2.20 dan persamaan 2.21 sehingga diperoleh:

$$\begin{aligned} m_1 &= \gamma_1 \cdot m_0 + (1 - \gamma_1) \cdot g_1 \\ &= 0,9 \cdot 0 + (1 - 0,9) \cdot (-5.607854843140) \\ m_1 &= -0.560785472393 \end{aligned} \tag{4.17}$$

$$\begin{aligned} v_1 &= \gamma_2 \cdot v_0 + (1 - \gamma_2) \cdot g_1^2 \\ &= 0,999 \cdot 0 + (1 - 0,999) \cdot (-0.560785472393)^2 \\ v_1 &= 0.031448036432 \end{aligned} \tag{4.18}$$

substitusi m_1 ke persamaan 2.22, diperoleh:

$$\begin{aligned} \hat{m}_1 &= \frac{-0.560785472393}{1 - 0,9} \\ \hat{m}_1 &= -5.60785472393 \end{aligned} \tag{4.19}$$

substitusi v_1 ke persamaan 2.23, diperoleh:

$$\begin{aligned} \hat{v}_1 &= \frac{0.031448034286499}{1 - 0,999} \\ \hat{v}_1 &= 31.448034286499 \end{aligned} \tag{4.20}$$

substitusi 4.18 dan 4.19 ke persamaan 3.54 sehingga diperoleh:

$$\begin{aligned} \left(\beta_1^{(15)}\right)_1 &= \left(\beta_1^{(15)}\right)_0 - 0,1 \cdot \frac{-5.60785472393}{\sqrt{31.448034286499} + 10^{-8}} \\ \left(\beta_1^{(15)}\right)_1 &= -0.0000999999997 \end{aligned}$$

lakukan secara berulang hingga seluruh parameter model telah diperbarui. Setelah proses *backpropagation* dari *mini batch* pertama selesai, maka ulangi proses *forward propagation* dengan *mini batch* selanjutnya kemudian lakukan kembali *backpropagation* dengan menggunakan nilai *cost function* yang baru, proses berulang hingga seluruh *mini-batch* telah digunakan untuk *training* (satu *epoch*).

Proses *training* model dilanjutkan dengan menguji model menggunakan dataset *validation*. Sama seperti dataset *training*, juga dibagi ke dalam *mini batch* dengan ukuran *batch size* sebesar 32. Artinya setiap satu *mini batch* mengandung sebanyak 32 tensor sebagai representasi dari sebanyak 32 gambar dalam dataset. Dengan menggunakan persamaan 3.55 diperoleh:

$$Xval^{\{t\}} = \{X_{(32 \cdot (t-1)) + 1}, X_{(32 \cdot t) + 2}, \dots, X_{32 \cdot t}\}; t = 1, 2, \dots, 22$$

pilih $t = 1$, sehingga diperoleh:

$$Xval^{\{1\}} = \{X_1, X_2, \dots, X_{32}\}$$

$X_{val}^{\{1\}}$

$$= \left\{ \begin{bmatrix} [0.117 \ 0.117 \ 0.117] & \square & \dots & \square & [0 \ 0 \ 0] \\ \vdots & \square & \square & \square & \vdots \\ [0.09 \ 0.09 \ 0.09] & \dots & [0.267 \ 0.267 \ 0.267] & \dots & [0 \ 0 \ 0] \\ \vdots & \square & \square & \backslash & \vdots \\ [0.043 \ 0.043 \ 0.043] & \square & \dots & \square & [0.035 \ 0.035 \ 0.035] \end{bmatrix}, \dots, \begin{bmatrix} [0 \ 0 \ 0] & \square & \dots & \square & [0 \ 0 \ 0] \\ \vdots & \square & \square & \square & \vdots \\ [0.565 \ 0.576 \ 0.627] & \dots & [0.619 \ 0.631 \ 0.674] & \dots & [0 \ 0 \ 0] \\ \vdots & \square & \square & \backslash & \vdots \\ [0.427 \ 0.447 \ 0.462] & \square & \dots & \square & [0 \ 0 \ 0] \end{bmatrix} \right\}$$

Lakukan perhitungan analog dengan langkah 4.2 hingga 4.15 sehingga diperoleh nilai akurasi menggunakan persamaan 2.25 dan nilai *cost* dengan menggunakan persamaan 2.18.

4.3 Testing Model

Sama seperti dataset *training*, juga dibagi ke dalam *mini batch* dengan ukuran *batch size* sebesar 32. Artinya setiap satu *mini batch* mengandung sebanyak 32 tensor sebagai representasi dari sebanyak 32 gambar dalam dataset. Dengan menggunakan persamaan 3.57 diperoleh:

$$X_{ts}^{\{t\}} = \{X_{(32 \cdot (t-1)) + 1}, X_{(32 \cdot t) + 2}, \dots, X_{32 \cdot t}\}; t = 1, 2, \dots, 33$$

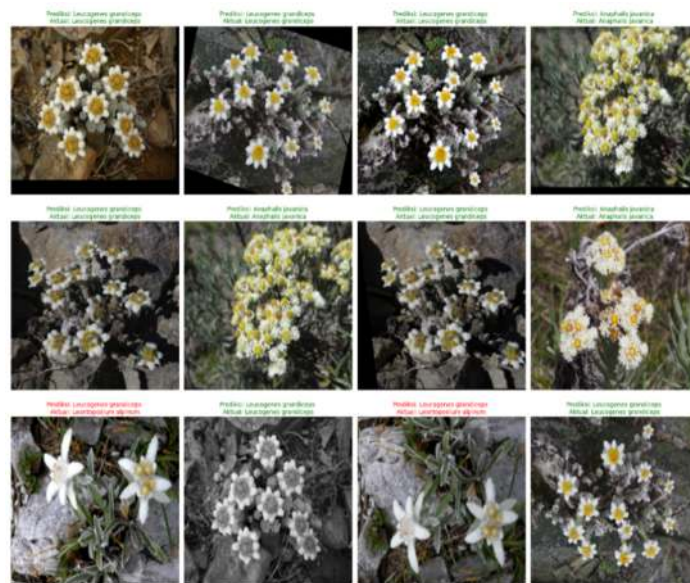
pilih $t = 1$, sehingga diperoleh:

$$X_{ts}^{\{1\}} = \{X_1, X_2, \dots, X_{32}\}$$

$Xts^{\{1\}}$

$$= \left\{ \begin{array}{l} \left[\begin{array}{ccc|ccc|ccc} [0.067 & 0.067 & 0.067] & \square & & \dots & \square & [0.168 & 0.164 & 0.176] \\ \vdots & & & \square & & \square & \square & \vdots & & \\ [0.067 & 0.067 & 0.067] & \dots & [0.074 & 0.074 & 0.074] & \dots & [0.047 & 0.047 & 0.05] \\ \vdots & & & \square & & \square & \backslash & \vdots & & \\ [0 & 0 & 0] & \square & & \dots & \square & [0 & 0 & 0] \end{array} \right] \\ \left[\begin{array}{ccc|ccc|ccc} [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \\ \vdots & & & \square & \square & \square & \vdots & & \\ [0 & 0 & 0] & \dots & [0 & 0 & 0] & \dots & [0 & 0 & 0] \\ \vdots & & & \square & \square & \backslash & \vdots & & \\ [0 & 0 & 0] & \square & \dots & \square & [0 & 0 & 0] \end{array} \right], \dots, \\ \left[\begin{array}{ccc|ccc|ccc} [0.38 & 0.412 & 0.231] & \square & & \dots & \square & [0.235 & 0.329 & 0.192] \\ \vdots & & & \square & & \square & \square & \vdots & & \\ [0.322 & 0.427 & 0.231] & \dots & [0.204 & 0.298 & 0.082] & \dots & [0.215 & 0.337 & 0.176] \\ \vdots & & & \square & & \square & \backslash & \vdots & & \\ [0.141 & 0.153 & 0.156] & \square & & \dots & \square & [0.114 & 0.094 & 0.067] \end{array} \right] \end{array} \right\}$$

Lakukan perhitungan analog dengan langkah 4.2 hingga 4.15 sehingga diperoleh nilai *loss* dan akurasi. Contoh hasil klasifikasi pada tahap *testing* model pada satu *batch* dapat dilihat pada Gambar 4.3. Pada gambar tersebut terlihat bahwa apabila hasil klasifikasi sama dengan label sebenarnya maka label di atas gambar akan berwarna hijau dan sebaliknya apabila salah akan berwarna merah.



(a)



(b)

Gambar 4. 3 Contoh Hasil klasifikasi pada tahap *testing* satu *mini batch*

(a) Indeks 1 hingga 12

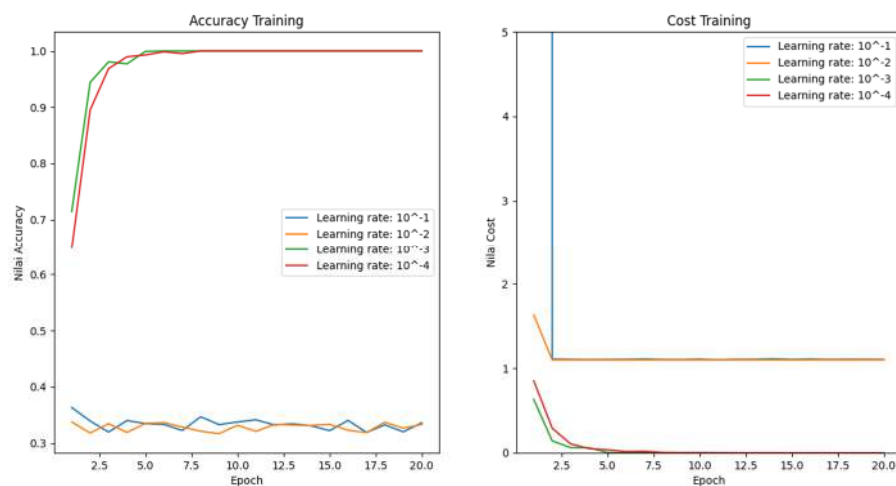
(b) Indeks 13 hingga 32

4.4 Performa Model

Proses *training* model dilakukan secara berulang menggunakan arsitektur model yang sama namun dengan nilai *learning rate* yang berbeda. *Learning rate* dengan nilai 10^{-i} dimana $i \leq 4$ akan diperbaharui setiap model telah selesai disimpan dan dites menggunakan *dataset test*. Performa model dihitung pada saat *training* dan *testing*.

1. Performa model saat proses *training*

Proses *training* model menghasilkan output berupa model dengan format “.keras” yang telah dilatih menggunakan dataset *train* dan *validation*. Setelah melakukan proses *training* dengan beberapa nilai *learning rate*, didapat nilai akurasi *Loss* dari yang dapat dilihat pada Gambar 4.3.



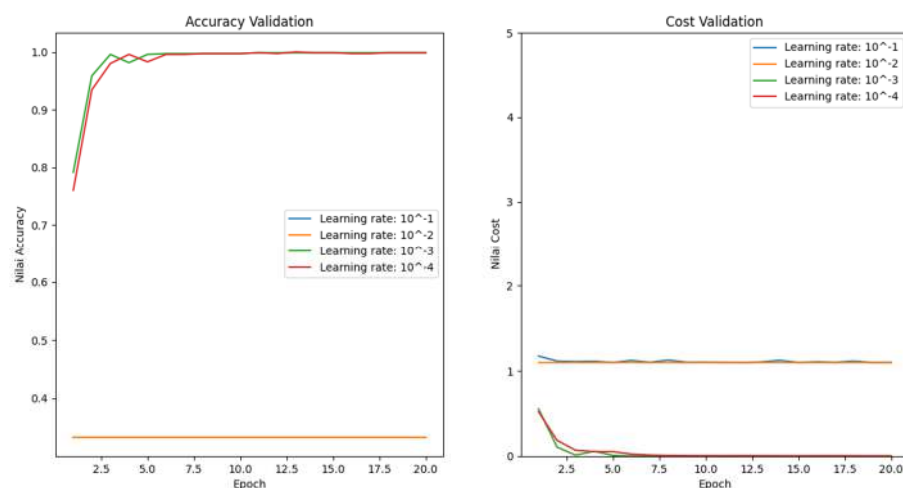
Gambar 4.4 Perbandingan grafik akurasi dan *cost* pada proses *training*

Dari Gambar 4.4 terlihat bahwa pada tahap *training*, model dengan *learning rate* 10^{-1} dan 10^{-2} tidak dapat bergerak ke titik optimal yang ditandai dengan nilai akurasi dan *cost* yang stasioner. Hal ini berbeda dengan kedua model lainnya dimana model dengan *learning rate* 10^{-3} dan 10^{-4} dapat bergerak ke titik

optimal yang ditandai dengan semakin besarnya nilai akurasi dan semakin kecilnya nilai *cost*.

Model dengan *learning rate* 10^{-3} telah mencapai akurasi *train* 100% hanya dengan enam *epoch* sedangkan model dengan *learning rate* 10^{-4} dapat mencapai akurasi *train* dengan sebanyak delapan *epoch*. Hasil tersebut sesuai dengan (Jordan, 2018) yang menyatakan bahwa nilai *learning rate* yang terlalu besar akan melewati nilai *loss* yang minimum dan tidak menuju ke titik konvergensi sedangkan semakin kecil nilai *learning rate* akan membutuhkan waktu yang semakin banyak dalam menuju titik konvergensi dan mendapatkan nilai *loss* minimum.

Setelah model dilatih sebanyak satu epoch, maka proses *training* dilanjutkan dengan menguji model menggunakan dataset *validation*. Performa model pada pengujian dengan menggunakan dataset *validation* dapat dilihat pada Gambar 4.5.



Gambar 4.5 Perbandingan grafik akurasi dan *cost* pada proses validasi

Serupa dengan pada saat *training* model menggunakan dataset *train*, model dengan *learning rate* 10^{-1} dan 10^{-2} tidak dapat bergerak ke titik optimal yang ditandai dengan nilai akurasi dan *cost* yang stasioner. Sedangkan model dengan *learning rate* 10^{-3} dan 10^{-4} memiliki akurasi yang mengalami kenaikan akurasi cukup drastis pada *epoch* awal hingga akhirnya melandai di sekitar angka 99%. Begitu juga dengan nilai *cost* model tersebut yang turun cukup drastis pada *epoch* awal hingga akhirnya melandai kurang dari 0,01. Hasil lengkap performa model pada saat proses *training* dapat dilihat pada Lampiran 3 hingga Lampiran 6.

2. Performa model saat proses *testing*

Setelah model dilatih hingga 20 *epoch* dan disimpan, langkah terakhir adalah dengan menguji model menggunakan data yang tidak digunakan pada saat proses *training* yaitu dataset *test*. Setelah dilakukan pengujian terhadap model dengan *learning rate* 10^{-i} dimana $i = 1, 2, 3, 4$ diperoleh hasil performa akurasi sebagai berikut.

Tabel 4.1 Performa model pada tahap *training*

Model Hasil	<i>Learning rate</i>			
	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Prediksi benar	350	350	811	851
Prediksi salah	700	700	239	199
Akurasi	33,33%	33,33%	77,238%	81,048%

Dari tabel 4.1 diperoleh bahwa nilai akurasi terbaik didapat oleh model dengan *learning rate* sebesar 10^{-4} . Model dengan *learning rate* sebesar 10^{-4} berhasil mengklasifikasikan 851 data gambar dari 1050 dataset *train* atau 81,048%.

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan tahapan penelitian yang telah dilakukan dan hasil performa model, dapat disimpulkan:

1. Algoritma CNN telah berhasil dibangun dengan pengoptimal *Adaptive Moment Estimation* untuk mengklasifikasikan spesies bunga edelweis dengan langkah sebagai berikut: tahap *forward propagation* dengan menggunakan sebanyak enam buah rangkaian *convolution layer* dan *pooling layer* untuk mengekstraksi fitur dan menggunakan *flatten layer*, *dense layer* sebanyak 128 neuron dengan aktivasi ReLU dan *dense layer* sebanyak 3 neuron dengan aktivasi *softmax*; tahap *backpropagation* dengan menggunakan pengoptimal Adam.
2. Hasil klasifikasi spesies bunga edelweis menggunakan model CNN didapatkan dengan beberapa model *learning rate* yang berbeda. Model dengan *learning rate* 10^{-1} dan 10^{-2} hanya berhasil memprediksi dengan benar sebanyak 33,33% gambar. Model dengan *learning rate* 10^{-3} dan 10^{-4} secara berturut-turut berhasil memprediksi 77,238% dan 81,048% gambar dengan tepat menggunakan dataset *test*. Sehingga dapat disimpulkan bahwa *learning rate* terbaik untuk model CNN yang telah dibangun adalah sebesar 10^{-4} dimana pada tahap *testing* nilai akurasi nya sebesar 81,048% yang berarti apabila terdapat 100.000 data *test* maka

81.048 data dapat diklasifikasikan secara tepat sesuai dengan label sebenarnya.

5.2 Saran

Berdasarkan hasil dan tahapan penelitian, penulis memberikan saran untuk penelitian serupa di masa depan antara lain sebagai berikut.

1. Pada penelitian ini penulis hanya membangun satu arsitektur CNN. Penelitian selanjutnya diharapkan dapat menggunakan beberapa arsitektur CNN menggunakan *transfer learning* dengan *pre-trained* model seperti MobileNet, VGG19, dan ResNet50 (Bichri et al., 2023).
2. Pada penelitian ini penulis tidak menggunakan *object detection* untuk mengambil bagian utama objek berupa kelopak bunga dari gambar. Penelitian selanjutnya diharapkan dapat menggunakan *object detection* untuk menyeleksi bagian objek dan mengabaikan bagian yang tidak diinginkan seperti tangkai dan daun bunga (Zhang et al., 2020).

DAFTAR PUSTAKA

- Alkaff, A. K., & Prasetyo, B. (2022). Hyperparameter Optimization on CNN Using Hyperband on Tomato Leaf Disease Classification. *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 479–483.
<https://doi.org/10.1109/CyberneticsCom55287.2022.9865317>
- Anam, C. (2021). *Menengok Desa Wisata Edelweis Wonokitri Pasuruan, Surganya Bunga Abadi*. Solopos Bisnis.
<https://bisnis.solopos.com/menengok-desa-wisata-edelweis-wonokitri-pasuruan-surganya-bunga-abadi-1191018#sp-sharing>
- Bichri, H., Chergui, A., & Hain, M. (2023). Image Classification with Transfer Learning Using a Custom Dataset: Comparative Study. *Procedia Computer Science*, 220, 48–54.
<https://doi.org/https://doi.org/10.1016/j.procs.2023.03.009>
- Chaki, J., & Dey, N. (2018). *A beginner's guide to image preprocessing techniques*. CRC Press.
- Chauchan, N. S. (2020). *Optimization Algorithms in Neural Networks*. KDnugget.
<https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html>
- Clark, J. (2024). *Pillow (PIL Fork) 10.3.0 documentation*. Python Software Foundation. <https://pillow.readthedocs.io/en/stable/about.html>
- Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (N. Tache, Ed.; 2nd ed.). O'Reilly Media, Inc.
<https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh & M. Titterton (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Vol. 9, pp. 249–256). PMLR.
<https://proceedings.mlr.press/v9/glorot10a.html>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

- Gupta, K. (2023). *Forward Propagation and Backpropagation Simplified*. Medium. https://medium.com/@kavita_gupta/forward-propagation-and-back-propagation-simplified-0b49f4e8732f
- Jean, H. (2018). *Deep Learning Book Series · 2.1 Scalars Vectors Matrices and Tensors*. <https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.1-Scalars-Vectors-Matrices-and-Tensors/>
- Jordan, J. (2018). *Setting the learning rate of your neural network*. <https://www.jeremyjordan.me/nn-learning-rate/>
- Kartika, N. (2023). *Hulun Hyang Menabur Benih Edelweiss, Menuai Cinta yang Abadi*. Dirjen KSDAE MENLHK. <https://ksdae.menlhk.go.id/artikel/12214/Hulun-Hyang-Menabur-Benih-Edelweiss-Menuai-Cinta-yang-Abadi.html>
- Kenton, W. (2022). *What Is End-To-End? A Full Process, From Start to Finish*. <https://www.investopedia.com/terms/e/end-to-end.asp>
- Ketkar, N. (2017). Convolutional Neural Networks. In *Deep Learning with Python: A Hands-on Introduction* (pp. 63–78). Apress. https://doi.org/10.1007/978-1-4842-2766-4_5
- Kingma, D. P., & Ba, J. L. (2017). Adam: A Method for Stochastic Optimization. *The 3rd International Conference for Learning Representations*.
- LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 253–256. <https://doi.org/10.1109/ISCAS.2010.5537907>
- Malasari, T. (2022). *Perbedaan Bunga Edelweis yang Tumbuh Liar dan Dibudidayakan*. Sariagri. <https://hortikultura.sariagri.id/96735/perbedaan-bunga-edelweis-yang-tumbuh-liar-dan-dibudidayakan>
- Malau, F. R. (2022). *Edelweiss Flower Dataset*. Kaggle. <https://www.kaggle.com/datasets/ndomalau/edelweis-flower>
- Malau, F. R., & Mulyana, D. I. (2022). Classification of Edelweiss Flowers Using Data Augmentation and Linear Discriminant Analysis Methods. *Journal of Applied Engineering and Technological Science*, 4(1), 139–148. <https://doi.org/10.37385/jaets.v4i1.960>

- Menteri LHK. (2018). Peraturan Menteri Lingkungan Hidup dan Kehutanan Republik Indonesia Nomor P.106/Menlhk/Setjen/Kum.1/12/2018 Tentang Perubahan Kedua Atas Peraturan Menteri Lingkungan Hidup dan kehutanan Nomor P.20/Menlhk/Setjen/Kum.1/6/2018 Tentang Jenis Tumbuhan dan Satwa Yang Dilindungi. *Kementrian Lingkungan Hidup Dan Kehutanan Republik Indonesia, Jakarta*.
- Muhammad, S., & Wibowo, A. T. (2021). Klasifikasi Tanaman Aglaonema Berdasarkan Citra Daun Menggunakan Metode Convolutional Neural Network (CNN). *EProceeding of Engineering*, 10621–10636.
- Presiden RI. (1990). Undang Undang No. 5 Tahun 1990 Tentang: Konservasi Sumberdaya Alam Hayati Dan Ekosistemnya. *Jakarta: Dephut*.
- Presiden RI. (1999). Peraturan Pemerintah No. 7 Tahun 1999 Tentang: Pengawetan Jenis Tumbuhan Dan Satwa. *Menteri Negara Sekretaris Negara Republik Indonesia, Jakarta*.
- Python Software Foundation. (2012). *About Python*.
<https://web.archive.org/web/20120420010049/http://www.python.org/about/>
- Python Software Foundation. (2024). *CSV File Reading and Writing*.
<https://docs.python.org/3/library/csv.html>
- Riani, A. (2024). *Beda dari Gunung Lain, Mengapa Bunga Edelweiss di Bromo Dijual ke Wisatawan?* Liputan 6.
<https://www.liputan6.com/lifestyle/read/5509148/beda-dari-gunung-lain-mengapa-bunga-edelweis-di-bromo-dijual-ke-wisatawan?page=4>
- Shinozuka, M., & Mansouri, B. (2009). 4 - Synthetic aperture radar and remote sensing technologies for structural health monitoring of civil infrastructure systems. In V. M. Karbhari & F. Ansari (Eds.), *Structural Health Monitoring of Civil Infrastructure Systems* (pp. 113–151). Woodhead Publishing.
<https://doi.org/10.1533/9781845696825.1.114>
- TensorFlow Developer. (2023). *Introduction to TensorFlow*.
<https://www.tensorflow.org/learn>
- TensorFlow Developer. (2024a). *Conv2D*.
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
- TensorFlow Developer. (2024b). *Preprocessing Image: ImageDataGenerator*.
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

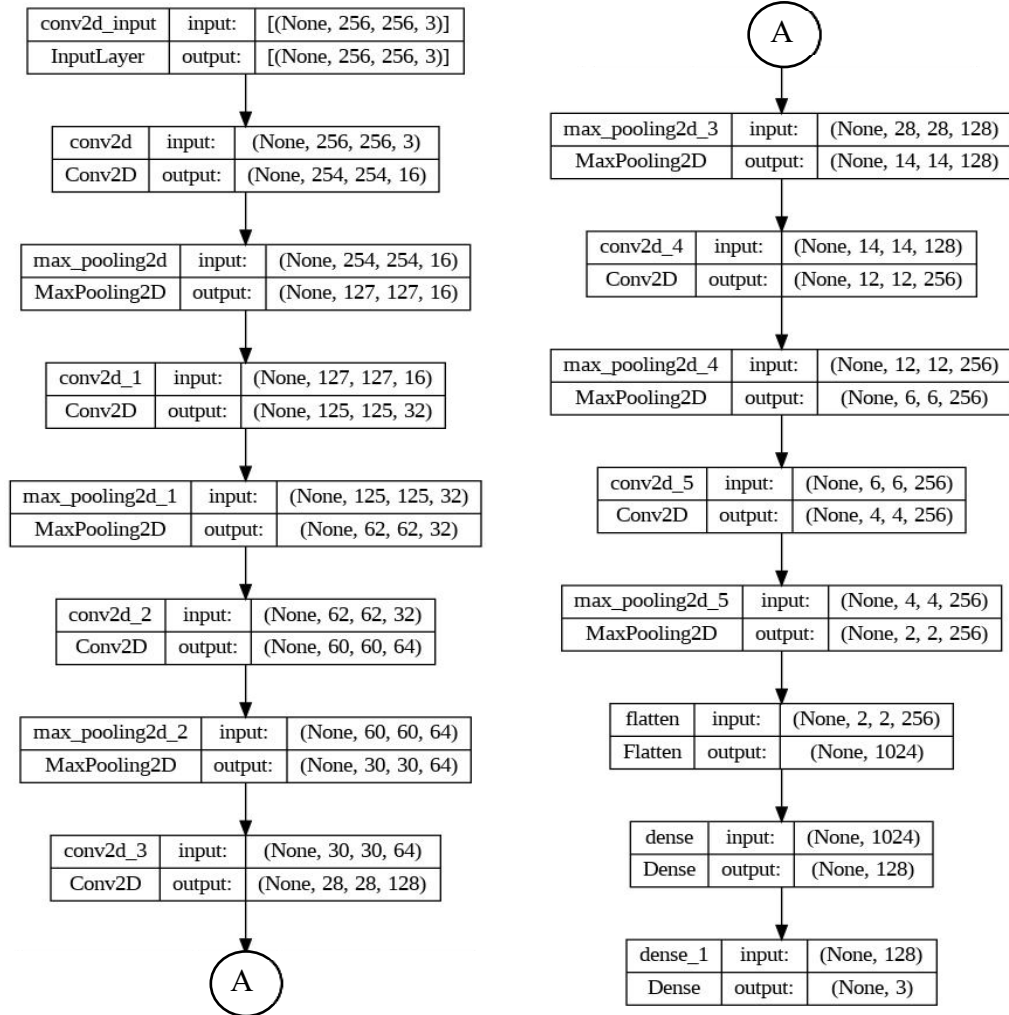
- Tineges, R., & Davita, A. W. (2021). *Mengenal Matplotlib untuk Visualisasi Data dengan Python*. <https://dqlab.id/mengenal-matplotlib-untuk-visualisasi-data-dengan-python>
- Zhang, Y., Song, C., & Zhang, D. (2020). Deep Learning-Based Object Detection Improvement for Tomato Disease. *IEEE Access*, 8, 56607–56614. <https://doi.org/10.1109/ACCESS.2020.2982456>

LAMPIRAN

Lampiran 1 Tautan *dataset* edelweiss

Nama file	Dataset Edelweis/README.md
Format	Markdown (.md)
Penyimpanan	GitHub
Tautan	https://github.com/WibiAnto/Skripsi/tree/main/Dataset%20Edelweis
Pratinjau	 <p>README.md</p> <h3>Dataset Edelweis</h3> <p>Tautan dataset train: Data Edelweis Compressed 512/Train Tautan dataset test: Data Edelweis Compressed 512/Test</p> <p>Sumber dataset: Edelweis Flower Dataset(Malau, 2022)</p> <p>Pra-pemrosesan gambar:</p> <ol style="list-style-type: none"> 1. Pengurangan dimensi dan penurunan kualitas menjadi 85% menggunakan <i>library</i> PIL 2. Normalisasi dengan membagi nilai piksel dengan 255 dan <i>split dataset train</i> menjadi <i>dataset train</i> dan <i>validation</i> dengan perbandingan trainvalidation = 80:20 menggunakan ImageDataGenerator 3. Seragamkan ukuran menjadi (256,256) dan buat mini-batch dengan batch size 32 menggunakan flow_from_directory

Lampiran 2 Arsitektur model CNN



Lampiran 3 Tabel hasil performa *training* dengan *learning rate* = 10^{-1}

Epoch	Accuracy	Cost	Val_accuracy	Val_cost
1	0.3633440435	1.10861.6484	0.3333333433	1.175189018
2	0.3394069374	1.109919429	0.3333333433	1.115606308
3	0.3197570443	1.106159091	0.3333333433	1.110463381
4	0.3401214778	1.103856921	0.3333333433	1.113955498
5	0.3347624242	1.105246186	0.3333333433	1.099225521
6	0.3333333433	1.106697083	0.3333333433	1.12331152
7	0.3226152062	1.110185266	0.3333333433	1.102322817
8	0.3469096124	1.105091214	0.3333333433	1.127149343
9	0.332976073	1.104145408	0.3333333433	1.102667332
10	0.3376205862	1.108542562	0.3333333433	1.1044451
11	0.3415505588	1.101005435	0.3333333433	1.100508809
12	0.3322615325	1.106595635	0.3333333433	1.09875989
13	0.334405154	1.107091427	0.3333333433	1.105635881
14	0.3308324516	1.11155057	0.3333333433	1.124777198
15	0.322257936	1.106076837	0.3333333433	1.09943819
16	0.3404787481	1.109796166	0.3333333433	1.108031034
17	0.3190425038	1.105489016	0.3333333433	1.101595521
18	0.3326188028	1.107105732	0.3333333433	1.116239667
19	0.3201143146	1.107440829	0.3333333433	1.098652959
20	0.3361915052	1.105361462	0.3333333433	1.099937558

Lampiran 4 Tabel hasil performa *training* model dengan *learning rate* = 10^{-2}

Epoch	Accuracy	Cost	Val_accuracy	Val_cost
1	0.3376205862	1.631802917	0.3333333433	1.098868966
2	0.3179706931	1.099529862	0.3333333433	1.098649621
3	0.3347624242	1.099072695	0.3333333433	1.098708034
4	0.3190425038	1.099094272	0.3333333433	1.098713517
5	0.3351196945	1.099026918	0.3333333433	1.098816395
6	0.3365487754	1.099006295	0.3333333433	1.09952271
7	0.3286888301	1.099741936	0.3333333433	1.098629475
8	0.3211861253	1.099277258	0.3333333433	1.098665595
9	0.3168988824	1.099322438	0.3333333433	1.098703265
10	0.3319042623	1.099245787	0.3333333433	1.098998547
11	0.320828855	1.099300861	0.3333333433	1.098837495
12	0.3333333433	1.09935832	0.3333333433	1.098917961
13	0.3322615325	1.099536419	0.3333333433	1.099244475
14	0.3315469921	1.099435925	0.3333333433	1.098823309
15	0.3333333433	1.099085212	0.3333333433	1.098723888
16	0.3229724765	1.099726081	0.3333333433	1.098693252
17	0.3186852336	1.099522591	0.3333333433	1.098614097
18	0.3372633159	1.098974347	0.3333333433	1.099264979
19	0.3265452087	1.09949255	0.3333333433	1.09882617
20	0.3333333433	1.099499345	0.3333333433	1.098760486

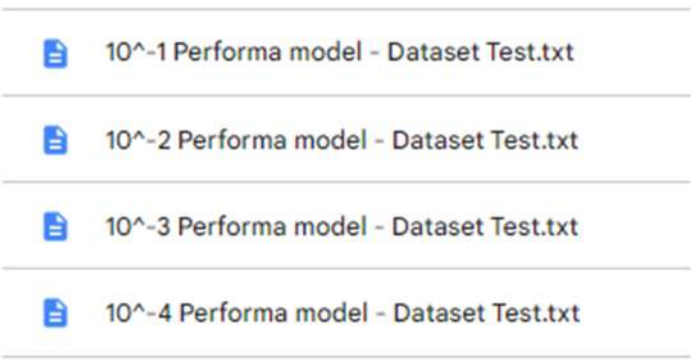
Lampiran 5 Tabel hasil performa *training* model dengan *learning rate* = 10^{-3}

Epoch	Accuracy	Cost	Val_accuracy	Val_cost
1	0.7145408988	0.6309927702	0.7911301851	0.5548452139
2	0.9442657828	0.1424323916	0.958512187	0.10712713
3	0.980707407	0.06133207679	0.9957081676	0.01366868149
4	0.9771347046	0.0610816367	0.9814019799	0.0599697046
5	0.9992854595	0.003591101384	0.9957081676	0.0100772744
6	1	0.0005103245494	0.9971387982	0.003599840915
7	1	0.0001236185199	0.9971387982	0.005364870187
8	1	5.43E-05	0.9971387982	0.004116348922
9	1	3.66E-05	0.9971387982	0.004094284028
10	1	2.84E-05	0.9971387982	0.003805993125
11	1	2.21E-05	0.9985693693	0.003461544868
12	1	1.74E-05	0.9985693693	0.003120372305
13	1	1.43E-05	0.9985693693	0.0029458059
14	1	1.19E-05	0.9985693693	0.002800464164
15	1	1.01E-05	0.9985693693	0.002772984793
16	1	8.31E-06	0.9985693693	0.002525513992
17	1	6.95E-06	0.9985693693	0.002325404435
18	1	5.31E-06	0.9985693693	0.003027451457
19	1	4.38E-06	0.9985693693	0.001964530675
20	1	3.46E-06	0.9985693693	0.001882268465

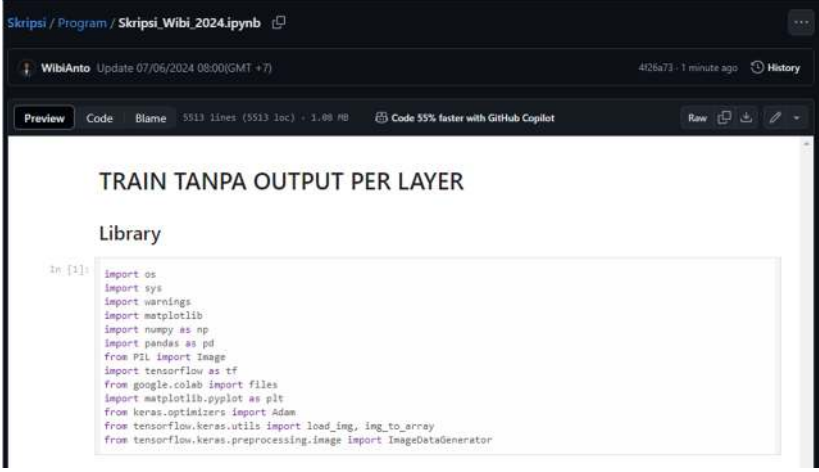

Lampiran 6 Tabel hasil performa *training* model dengan *learning rate* = 10^{-4}

Epoch	Accuracy	Cost	Val_accuracy	Val_cost
1	0.6482537389	0.8520622849	0.7599999905	0.5220874548
2	0.8952245116	0.2928564548	0.9342857003	0.1849969923
3	0.9686386585	0.105661951	0.9800000191	0.070398435
4	0.9896650314	0.05057924613	0.9957143068	0.05450107157
5	0.992872417	0.0355530642	0.9828571677	0.05464889482
6	0.9985744953	0.01522418112	0.9957143068	0.02608193457
7	0.9953670502	0.01868626848	0.9957143068	0.01365053281
8	1	0.00388872507	0.9971428514	0.009760740213
9	1	0.002491851104	0.9971428514	0.007670096587
10	1	0.002808282385	0.9971428514	0.007035227958
11	1	0.001547174295	0.9985714555	0.006000065245
12	1	0.001100948546	0.9971428514	0.006866776384
13	1	0.0009268952999	1	0.006436590571
14	1	0.0009894989198	0.9985714555	0.00527616078
15	1	0.0005485510337	0.9985714555	0.004929999355
16	1	0.0004485327227	0.9971428514	0.006413518451
17	1	0.0003537735029	0.9971428514	0.006462403107
18	1	0.0003105897049	0.9985714555	0.004798488226
19	1	0.0002686932858	0.9985714555	0.004817258567
20	1	0.0002389399015	0.9985714555	0.004645100329

Lampiran 7 Tautan hasil *testing* per *mini batch*

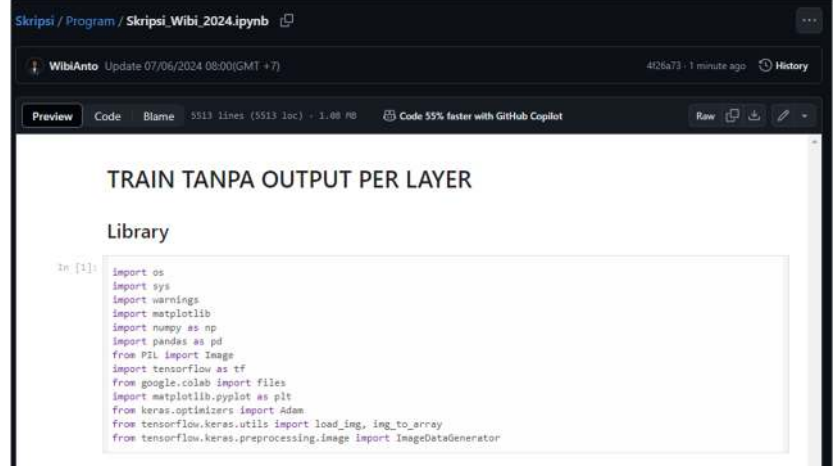
Nama file	Performa model – Dataset test
Format	Folder Google Drive
Penyimpanan	Google Drive
Tautan	https://drive.google.com/drive/folders/1aAa7Eea7eXwloGri5TD4F6yUpoI-u-WU?usp=drive_link
Pratinjau	

Lampiran 8 Output dari perhitungan lengkap tahap *forward propagation*

Nama file	Dataset Edelweis/README.md
Format	Folder Google Drive
Penyimpanan	Google Drive
Tautan	
Pratinjau	

Lampiran 9 *Syntax* Python

Tautan :

Nama file	Skripsi Wibi 2024
Format	IPython Notebook (.ipynb)
Penyimpanan	GitHub
Tautan	https://github.com/WibiAnto/Skripsi/blob/main/Program/Skripsi_Wibi_2024.ipynb
Pratinjau	

Lampiran 10. Spesifikasi *software* dan *hardware* yang digunakan dalam penelitian

Software:

- Google Colaboratory dengan Python 3 GCE
- Versi library matplotlib : 3.7.1
- Versi library numpy : 1.25.2
- Versi library PIL : 9.4.0
- Versi library tensorflow : 2.15.0

Hardware:

- Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz
- Windows 11 Home Single Language 64-bit (versi 23H2)
- RAM: 12GB DDR4 1333 MHz (Dual channel 4GB+8GB)
- ROM: 1TB SATA HDD 2,5" 5400rpm (HDD WD10SPZX)

RIWAYAT HIDUP PENULIS

DATA PRIBADI

Nama Lengkap : Wibi Anto
NPM : 140110200025
Tempat, Tanggal Lahir : Cirebon, 16 Mei 2003
Jenis Kelamin : Laki-laki
Agama : Islam
Alamat : Blok Karang Rame, Desa Gamel, Kec. Plered,
Kabupaten Cirebon, 45154
Email : wibi20001@mail.unpad.ac.id

RIWAYAT PENDIDIKAN

2009 – 2015 SD Negeri 1 Gamel
2015 – 2018 SMP Negeri 1 Sumber
2018 – 2020 SMA Negeri 2 Cirebon
2020 – 2024 Program Studi S-1 Matematika, Fakultas Matematika dan Ilmu
Pengetahuan Alam, Universitas Padjadjaran