

**KLASIFIKASI PENYAKIT KANKER KULIT MENGGUNAKAN DEEP
LEARNING BERBASIS APLIKASI WEB**

SKRIPSI



Disusun Oleh:

MUHAMMAD NAUFAL

11170970000033

**PROGRAM STUDI FISIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH
JAKARTA
2021 M / 1443 H**

LEMBAR PERSETUJUAN PEMBIMBING

KLASIFIKASI PENYAKIT KANKER KULIT MENGGUNAKAN DEEP LEARNING BERBASIS APLIKASI WEB

SKRIPSI

Diajukan Untuk Memenuhi Persyaratan Memperoleh Gelar Sarjana Sains (S.Si)

Muhammad Naufal

NIM. 11170970000033

Menyetujui,

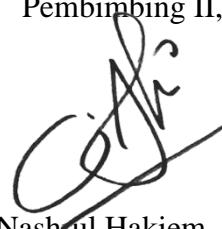
Pembimbing I,



Anugrah Azhar M.Si

NIP. 199210312 01801 1 003

Pembimbing II,



Ir. Nashrul Hakiem, Ph.D

NIP. 19710608 200501 1 005

Mengetahui,

Ketua Program Studi Fisika

UIN Syarif Hidayatullah Jakarta



Tati Zera M.Si

NIP. 19690608 200501 2 002

LEMBAR PENGESAHAN UJIAN

Skripsi berjudul "**Klasifikasi Penyakit Kanker Kulit Menggunakan Deep Learning Berbasis Aplikasi Web**" yang ditulis oleh **Muhammad Naufal** dengan NIM 11170970000033 telah diuji dihadapan dewan pengaji dan dinyatakan lulus dalam sidang Munaqasah Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta pada Kamis, 18 November 2021. Skripsi ini telah diterima sebagai salah satu syarat memperoleh gelar Sarjana S1 (S.Si) Program Studi Fisika.

Jakarta, 18 November 2021

Menyetujui,

Pengaji I,

DR. Ambran Hartono, M.Si
NIP. 19710408 200212 1 002

Pengaji II,

Elvan Yuniarti, M.Si
NIP. 19791227 200801 2 015

Pembimbing I,

Anugrah Azhar, M.Si
NIP. 199210312 01801 1 003

Pembimbing II,

Ir. Nashrul Hakiem, Ph.D
NIP. 19710608 200501 1 005

Mengetahui,

Dekan Fakultas Sains dan Teknologi,
Universitas Islam Negeri Syarif Hidayatullah Jakarta



Ir. Nashrul Hakiem, Ph.D
NIP. 19710608 200501 1 005

Ketua Program Studi Fisika,
Fakultas Sains dan Teknologi

Tati Zera, M.Si
NIP. 19690608 200501 2 002

LEMBAR PERNYATAAN

Yang bertandatangan dibawah ini:

Nama : Muhammad Naufal

NIM : 11170970000033

Dengan ini menyatakan bahwa skripsi yang berjudul **KLASIFIKASI PENYAKIT KANKER KULIT MENGGUNAKAN DEEP LEARNING BERBASIS APLIKASI WEB** adalah benar merupakan karya saya sendiri dan tidak melakukan tindakan plagiat dalam penyusunannya. Adapun kutipan yang ada dalam penyusunan karya ini telah saya cantumkan sumber kutipannya dalam skripsi.

Demikian pernyataan ini dibuat untuk dipergunakan seperlunya.

Jakarta, 18 November 2021




Muhammad Naufal

NIM. 11170970000033

ABSTRAK

Nama : Muhammad Naufal

Program Studi : Fisika

Judul : Klasifikasi Penyakit Kanker Kulit Menggunakan Deep Learning Berbasis Aplikasi Web

Pembimbing : 1. Anugrah Azhar M.Si

2. Ir. Nashrul Hakiem, Ph.D

Belakangan ini, istilah *deep learning* menjadi populer di bidang edukasi dan industri karena performa dan fitur yang dimiliki yang dapat mempermudah manusia khususnya dalam menyelesaikan tugas-tugas yang tidak dapat atau sulit diselesaikan oleh manusia. *Deep learning* merupakan bagian dari *machine learning* dengan meniru jaringan syaraf di dalam tubuh. Salah satu hal yang dapat dilakukan oleh deep learning adalah klasifikasi, klasifikasi termasuk bagian dari supervised learning yang berarti data mempunyai label. Selain di dunia industri dan teknologi, salah satu penerapan metode deep learning ini adalah instrumentasi medis seperti pada penelitian ini. Sehingga tujuan dari penelitian ini adalah mengklasifikasikan penyakit kanker kulit dengan menggunakan metode deep learning dan menggunakan teknik *convolutional neural network* dikarenakan data berupa gambar, kemudian menggunakan fitur transfer learning dan pre-trained model kemudian setelah model terlatih, maka akan diterapkan ke dalam aplikasi berbasis web. Kemudian dari hasil pelatihan model, dapat disimpulkan bahwa *pre-trained model* dengan hasil akurasi, *precision* dan *recall* yang paling tinggi adalah *xception* dengan tingkat akurasi berada di 90%, kemudian *precision* rata-rata di 72% dan *recall* rata-rata berada di 60%. Dengan jenis kanker kulit yang paling banyak dideteksi secara benar adalah Melanocytic Nevi dengan presentase benar sebesar 80% - 90% dari setiap *pre-trained model*

Kata kunci : Klasifikasi, *Convolutional Neural Network*, Kanker Kulit, *Transfer Learning*.

ABSTRACT

Name : Muhammad Naufal
Program : Physics
Title : Web Based Application to Classify Skin Cancer with Deep Learning
Adviser : 1. Anugrah Azhar M.Si
2. Ir. Nashrul Hakiem, Ph.D

Recently, deep learning term became popular in industrial and educational field because it's performance that help human being especially for finishing the task that human can't solved or hard to solved only by hand. Deep learning is part of machine learning by applying algorithms that are similar to the neural networks in the brain. A thing that can be done by deep learning is classification, classification is part of supervised learning which means data has labels. Beside industry and technology, deep learning can be applied in the instrumentation medical field as in this study. So, the purpose of this study is to classify skin cancer by using deep learning methods and using convolutional neural network techniques as the data is images, then using transfer learning features and pre-trained models, after the model is trained, the model will be applied to a web-based application. From the results of the training model, it could be concluded that the highest accuracy, precision and recall is exception. With the average of accuracy in 90%, then the average of precision is 72% and the recall 60%. With Melanocytic Nevi as the most right predicted skin cancer labels that machine can predict with average true percentage is 80% - 90% for each pre-trained models.

Keywords : Classification, Convolutional Neural Network, Skin Cancer, Transfer Learning.

KATA PENGANTAR

Dengan mengucap *Bismillahirahmanirahim*. Segala puji bagi Allah SWT yang telah memberikan nikmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul “Klasifikasi Penyakit Kanker Kulit Menggunakan Deep Learning Berbasis Aplikasi Web” dengan baik dan benar. Tidak lupa shalawat dan salam selalu tercurahkan kepada baginda Nabi Muhammad SAW beserta para keluarganya, sahabatnya dan umatnya.

Tugas Akhir ini dapat selesai dengan baik pastinya tidak lepas dari dukungan dan bantuan berbagai pihak. Pada kesempatan kali ini, izinkan penulis mengucapkan terima kasih kepada:

1. Ibu, Adik dan Saudara-saudara saya yang selalu memberikan dukungan doa, moril dan materiil hingga saat ini.
2. Bapak Anugrah Azhar, M.Si. selaku dosen pembimbing pertama dan Ir. Nashrul Hakiem Ph. D selaku dosen pembimbing kedua yang senantiasa selalu membimbing, memberikan pengetahuan, saran, dan arahan dalam penelitian ini.
3. Ibu Tati Zera, M.Si selaku Ketua Program Studi Fisika Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta.
4. Ir. Nashrul Hakiem Ph. D selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Syarif Hidayatullah Jakarta.
5. Bapak/Ibu selaku dosen penguji dalam siding munaqasyah
6. Seluruh dosen Prodi Fisika yang telah memberikan pengetahuan dan saran untuk penelitian ini
7. Seluruh instruktur dan staff di program Bangkit 2021 yang telah memberikan pengetahuan dan ide untuk penelitian ini
8. Seluruh instruktur di Coursera, Dicoding, DQLab, Udemy, dan Youtube yang telah memberikan pengetahuan dan ide untuk penelitian ini.
9. Teman-teman kontrakan Ikhwan Sholeh yang selalu memberikan semangat dan motivasi kepada penulis.

10. Teman-teman se-angkatan 2017 Fisika yang selalu memberikan semangat dan motivasi kepada penulis.

Penulis telah berusaha menyusun laporan tugas akhir ini dengan sebaiknya. Namun, pasti masih banyak kekurangan di sana-sini. Penulis mengharapkan kritik dan saran yang membangun demi perbaikan di masa yang akan datang, dan penulis berharap agar penelitian ini dapat dikembangkan dengan apa yang sudah dilakukan oleh penulis. Diskusi, kritik serta saran yang membangun dari pembaca dapat disampaikan melalui alamat surat elektronik penulis, muhammad.naufal17@mhs.uinjkt.ac.id atau mnaufal0999@gmail.com. Semoga laporan tugas akhir ini bermanfaat bagi penulis pribadi maupun bagi siapa pun yang membacanya.

Bekasi, 18 November 2021

Muhammad Naufal

DAFTAR ISI

LEMBAR PERSETUJUAN PEMBIMBING	ii
LEMBAR PENGESAHAN UJIAN.....	iii
LEMBAR PERNYATAAN.....	iv
ABSTRAK	v
ABSTRACT	vi
KATA PENGANTAR	vii
DAFTAR ISI.....	ix
DAFTAR GAMBAR	xii
DAFTAR TABEL.....	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	4
1.6 Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA.....	6
2.1 <i>Artificial Intelligence</i>	6
2.2 <i>Machine Learning</i>	7
2.3 <i>Deep Learning</i>	11
2.3.1 <i>Backpropagation</i>	18
2.3.2 <i>Convolutional Neural Network</i>	20
2.4 <i>Tools</i>	23
2.4.1 <i>Python</i>	23
2.4.2 <i>NumPy</i>	23
2.4.3 <i>TensorFlow dan Keras</i>	24
2.4.4 <i>Pandas</i>	25
2.4.5 <i>Matplotlib</i>	25
2.4.6 <i>Scikit-Learn</i>	25
2.5 Evaluasi.....	26

2.5.1	<i>Overfitting</i> dan <i>Underfitting</i>	26
2.5.2	<i>Confusion Matrix</i>	26
2.6	Kanker Kulit.....	28
2.7	Penelitian Sejenis	30
BAB III METODOLOGI PENELITIAN.....		31
3.1	Waktu dan Tempat Penelitian.....	31
3.2	Peralatan.....	31
3.3	Diagram Alur Penelitian	32
3.4	Pemrosesan Dataset	33
3.4.1	<i>Exploratory Data Analysis (EDA)</i>	33
3.4.2	Konversi Dataset.....	34
3.4.3	Pembagian Data Latih, Validasi dan Tes	35
3.4.4	Augmentasi dataset	36
3.5	Membangun Model.....	38
3.6	Alur Model.....	39
3.6.1	Inisialisasi Input	39
3.6.2	Inisialisasi Pre-Trained Model	40
3.6.3	<i>AveragePooling</i>	40
3.6.4	<i>Flatten Layer</i>	40
3.6.5	<i>Dense Layer</i>	41
3.6.6	<i>Dropout Layer</i>	41
3.6.7	Pre-Trained Model	43
3.7	Pelatihan Model	48
3.7.1	<i>Compile</i>	48
3.7.2	<i>Fit</i>	49
3.8	<i>Deployment</i>	50
BAB IV HASIL DAN PEMBAHASAN		53
4.1	Hasil dan Evaluasi Model	53
4.1.1	Evaluasi Hasil Pelatihan.....	53
4.1.2	<i>Confusion Metrics</i>	58
BAB V KESIMPULAN DAN SARAN.....		68
5.1	Kesimpulan	68
5.2	Saran	68
DAFTAR PUSTAKA		70

LAMPIRAN	80
a. <i>Script code</i> untuk <i>Training</i>	80
b. Dokumentasi proses penelitian	86

DAFTAR GAMBAR

Gambar 2.1 Bagian dari Artificial Intelligence[7]	6
Gambar 2.2 Bagian dari <i>Artificial Intelligence</i> [10]	7
Gambar 2.3 Alur Pemrograman Biasa dengan <i>Machine Learning</i> [12]	8
Gambar 2.4 <i>Supervised Learning</i> [17]	9
Gambar 2.5 <i>Unsupervised Learning</i> [18]	10
Gambar 2.6 <i>Semi-Supervised Learning</i> [19].....	11
Gambar 2.7 <i>Deep Learning</i> [20]	11
Gambar 2.8 Jaringan Syaraf Manusia[26]	12
Gambar 2.9 Diagram <i>Artificial Neural Network</i> [29]	13
Gambar 2.10 Diagram <i>Neural Network</i> sederhana	14
Gambar 2.11 Fungsi Aktivasi[33].....	14
Gambar 2.12 Macam Fungsi Aktivasi[35].....	14
Gambar 2.13 Grafik <i>ReLU</i> [36]	15
Gambar 2.14 Grafik <i>Softmax</i> [39].....	16
Gambar 2.15 Bagaimana <i>Softmax</i> Bekerja[41].....	16
Gambar 2.16 Contoh Diagram Neural Network yang mempunyai banyak nodes	17
Gambar 2.17 Diagram <i>Backpropagation</i> [30]	18
Gambar 2.18 Diagram <i>Convolutional Neural Network</i> [45].....	20
Gambar 2.19 Ilustrasi <i>AveragePool2D</i>	22
Gambar 2.20 Bagaimana <i>Dropout</i> Bekerja[48]	22
Gambar 2.21 Ilustrasi overfitting dan underfitting[61].....	26
Gambar 2.22 Contoh Confusion Matrix[63].....	27
Gambar 2.23 Kanker Kulit[64]	28
Gambar 2.24 Spektrum Gelombang Elektromagnetik[67]	29
Gambar 3.1 Diagram Alur Penelitian.....	32
Gambar 3.2 Sample 25 Gambar yang dipilih acak	34
Gambar 3.3 Ilustrasi konversi dataset	34
Gambar 3.4 Ilustrasi pembagian dataset[72].....	35
Gambar 3.5 Ilustrasi augmentasi dataset[73].....	36
Gambar 3.6 Diagram Alur Model Transfer Learning	39
Gambar 3.7 Visualisasi Model Transfer Learning	39
Gambar 3.8 Ilustrasi <i>Flatten Layer</i>	40
Gambar 3.9 Jumlah Parameter <i>Train</i> yang dihilangkan.....	42
Gambar 3.10 Diagram Model DenseNet201.....	43
Gambar 3.11 Diagram Model <i>InceptionV3</i>	44
Gambar 3.12 Diagram Model <i>ResNet152V2</i>	45
Gambar 3.13 Diagram Model <i>Xception</i>	46
Gambar 3.14 Perbandingan <i>Trainable</i> Parameter antar <i>Pre-Trained</i> Model.....	47
Gambar 3.15 Perbandingan Non- <i>Trainable</i> Parameter antar Pre-Trained Model.	47
Gambar 3.16 Diagram Alur Proses <i>Deployment</i>	50
Gambar 3.17 Demonstrasi Aplikasi Berbasis <i>Web</i> Klasifikasi Penyakit Kulit....	52
Gambar 4.1 Metrics Accuracy, Loss, Val_Accuracy, Val_Loss	53
Gambar 4.3 DenseNet201 Confusion Metrics.....	59
Gambar 4.4 InceptionV3 Confusion Metrics.....	61

Gambar 4.5 <i>ResNet152 Confusion Metrics</i>	63
Gambar 4.6 <i>Xception Confusion Metrics</i>	65
Gambar Lampiran 1. Proses Konversi Gambar ke Tensor	86
Gambar Lampiran 2. Proses Menampilkan Beberapa Gambar Kanker Kulit.....	86
Gambar Lampiran 3. Proses Pelatihan Salah Satu Pre-Trained Model (Xception)	87
Gambar Lampiran 4. Proses Pelatihan Salah Satu Pre-Trained Model (Xception)	87
Gambar Lampiran 5. Menyimpan Model Untuk Kemudian di Deploy	87
Gambar Lampiran 6. Proses Download dari Kaggle	88
Gambar Lampiran 7. Proses Ekstraksi Dataset dan Resizing Dataset	88
Gambar Lampiran 8. Proses Ekstraksi Dataset dan Resizing Dataset	88
Gambar Lampiran 9. Proses Testing Dataset.....	89
Gambar Lampiran 10. Proses Pembuatan API.....	90
Gambar Lampiran 11. Proses Pembuatan Aplikasi Web	90
Gambar Lampiran 12. Proses Running Server Flask.....	91
Gambar Lampiran 13. Aplikasi Web	91
Gambar Lampiran 14. Aplikasi Web	92

DAFTAR TABEL

Tabel 2. 1 Penelitian Sejenis	30
Tabel 3.1 Jumlah Data dan Label.....	33
Tabel 4.1 Perbandingan antara <i>pre-trained model</i> terhadap <i>Metrics Accuracy</i>	54
Tabel 4.2 Perbandingan antara <i>pre-trained model</i> terhadap <i>Metrics Loss</i>	55
Tabel 4.3 Perbandingan antara <i>pre-trained model</i> terhadap <i>Metrics Val_Accuracy</i>	56
Tabel 4.4 Perbandingan antara <i>pre-trained model</i> terhadap <i>Metrics Val_Loss</i>	57
Tabel 4.5 Jumlah Data dan Label <i>Test</i>	58
Tabel 4.6 <i>Precision, Recall</i> dan <i>Accuracy</i> dari <i>DenseNet201</i>	60
Tabel 4.7 <i>Precision, Recall</i> dan <i>Accuracy</i> dari <i>InceptionV3</i>	62
Tabel 4.8 <i>Precision, Recall</i> dan <i>Accuracy</i> dari <i>ResNet152</i>	64
Tabel 4.9 <i>Precision, Recall</i> dan <i>Accuracy</i> dari <i>Xception</i>	66

BAB I

PENDAHULUAN

1.1 Latar Belakang

Belakangan ini, istilah *Machine learning* sedang populer di dunia pendidikan maupun industri, karena kemampuan dan fitur yang dimiliki untuk mempermudah manusia dalam mengerjakan tugas-tugas yang sulit ataupun yang tidak bisa dikerjakan oleh manusia. *Machine learning* merupakan cabang ilmu dari ilmu komputer yang mempelajari tentang algoritma komputer yang dapat membuat komputer belajar layaknya manusia dengan membaca dan menggunakan pola yang tersedia dari data[1]. *Machine learning* ini dapat belajar berbagai jenis data, mulai dari data regresi, data klasifikasi sampai data yang abstrak seperti gambar, video, suara

Kemudian *deep learning*, yang merupakan turunan dari *Machine Learning* yang dapat mempelajari data dengan lebih efisien dalam hal manajemen memori dan performa. Algoritma ini meniru sistem kerja dari jaringan syaraf manusia yang mengirimkan sinyal dari neuron satu ke neuron yang lainnya[2]. Dimana neuron-neuron tersebut disebut sebagai hidden layer dan sinyal-sinyal tersebut terdiri dari data yang dikirimkan dari hidden layer satu ke hidden layer lainnya dan hidden layer terakhir sebagai output[3].

Untuk klasifikasi gambar, video, suara dan sejenisnya itu menggunakan model deep learning dua dimensi atau yang disebut sebagai convolutional neural network[4]. Sehingga di dalam pengklasifikasianya, gambar dan sejenisnya yang merupakan bentuk data dua dimensi tersebut, harus melalui proses feature extraction yang berarti penyesuaian bentuk dari data dua dimensi ke dalam satu dimensi dan kemudian baru bisa dilatih ke dalam model neural network[5].

Digunakan pre-trained model, artinya adalah model dari deep learning (seperti jumlah hidden layer, fungsi aktivasi dan lainnya) yang sudah pernah dilatih sebelumnya oleh dataset yang berbeda. Kemudian dari pre-trained model tersebut, kita juga dapat menggunakannya dan memodifikasi parameter yang ada

didalamnya seperti mengganti fungsi aktivasi, menambah dan mengurangi jumlah hidden layer dan beberapa metode lainnya. Metode ini dinamakan transfer learning[6].

Selain diterapkan dibidang industri dan teknologi, *machine learning* ini juga dapat digunakan dalam instrumentasi medis. Dalam implementasinya di instrumentasi medis ini, *machine learning* dapat mengklasifikasikan sebuah penyakit hanya dengan memasukkan beberapa parameter data dari penyakit, baik itu data berupa pola seperti *time series* maupun data gambar sebuah penyakit seperti luka, jerawat maupun kanker. Sehingga dapat mempermudah pekerjaan dokter dalam mendiagnosis suatu penyakit.

Oleh karena itu, penelitian ini dilakukan untuk mengklasifikasikan sebuah penyakit yang dapat diinterpretasikan oleh gambar yaitu penyakit yang berhubungan dengan kulit seperti kanker kulit dengan menggunakan *machine learning* dengan metode *convolutional neural network*.

Penelitian ini merupakan sebagai salah satu bentuk implementasi *machine learning* dalam bidang penelitian fisika khususnya dalam bidang instrumentasi medis.

1.2 Perumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah dalam penulisan ini adalah:

1. Manakan pre-trained model yang menghasilkan nilai tertinggi?
2. Apa saja parameter yang digunakan dalam pelatihan dataset kanker kulit tersebut sehingga akan menghasilkan akurasi yang tinggi?
3. Berapa tingkat akurasi, *loss*, *val_accuracy*, *val_loss* dan *confusion metrics* yang dihasilkan oleh *pre-trained* model yang menghasilkan nilai tertinggi?
4. Dari data yang dilatih terdapat 7 jenis kanker kulit, dari 7 jenis kanker kulit tersebut, jenis kanker kulit apa yang dapat dibedakan secara baik oleh algoritma *deep learning*?

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini adalah sebagai berikut:

1. Dataset yang digunakan hanya dataset yang berasal dari *Skin Cancer MNIST HAM10000* dan hanya menggunakan 7 label.
2. Metode yang digunakan hanya menggunakan metode *deep learning* dengan menggunakan algoritma *Convolutional Neural Network* dengan *pre-trained model* dan hanya menggunakan 4 *pre-trained model* yaitu *DenseNet201*, *InceptionV3*, *ResNet152*, dan *Xception*.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang ada, maka tujuan penelitian ini adalah sebagai berikut:

1. Merancang dan membuat model untuk klasifikasi penyakit kanker kulit dengan mengimplementasikan metode algoritma *deep learning*.
2. Menganalisis performa dari masing-masing *pre-trained* model dalam melatih dataset.
3. Menyimpulkan *pre-trained* model apa saja yang cocok untuk mengidentifikasi dan mengklasifikasikan dataset kanker kulit dengan baik.
4. Menyimpulkan kanker kulit jenis apa saja yang dengan mudah dapat diidentifikasi dan diklasifikasikan oleh algoritma *deep learning*.
5. Membuat program berbasis web untuk klasifikasi penyakit kanker kulit berdasarkan model yang telah dibuat.
6. Mempermudah tim medis dalam mengidentifikasi jenis dari penyakit kanker kulit, sehingga tim medis dapat dengan mudah untuk mengobati penyakit nya, dikarenakan karakteristik dari pola penyakit kanker kulit telah diketahui.

1.5 Manfaat Penelitian

Adapun penelitian ini diharapkan memiliki manfaat sebagai berikut:

1. Sebagai salah satu solusi dalam mendiagnosis penyakit kanker kulit, apakah penyakit kanker kulit tersebut sudah ganas atau masih jinak, sehingga dapat dihindarkan (dengan petunjuk dari dokter tentunya).
2. Sebagai langkah awal dari penelitian lanjutan yang akan menggunakan model *deep learning* seperti ini, sehingga model akan terus *improve* performanya, atau lebih baik lagi jika di penelitian selanjutnya dapat membuat *pre-trained* model dari dataset yang digunakan dalam penelitian ini.
3. Menambah pengetahuan bagi mahasiswa, pembaca dan masyarakat umum mengenai *deep learning* dan penggunaannya di bidang instrumentasi medis.

1.6 Sistematika Penulisan

Penulisan penelitian tugas akhir ini dibagi menjadi lima bab, yang pada umumnya diuraikan sebagai berikut:

BAB I PENDAHULUAN

Bab ini meliputi: Latar Belakang, Perumusan Masalah, Tujuan Penelitian, Manfaat Penelitian dan Sistematika Penulisan

BAB II TINJAUAN PUSTAKA

Pada bab ini ditinjau teori-teori dasar yang menunjang penelitian ini.

BAB III METODE PENELITIAN

Pada bab ini dijelaskan mengenai tempat dan waktu melakukan penelitian, peralatan dan bahan yang digunakan dan tahapan penelitian.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai hasil berupa data yang telah diambil sebelumnya dan juga dijelaskan tentang pembahasan berdasarkan data yang telah diperoleh

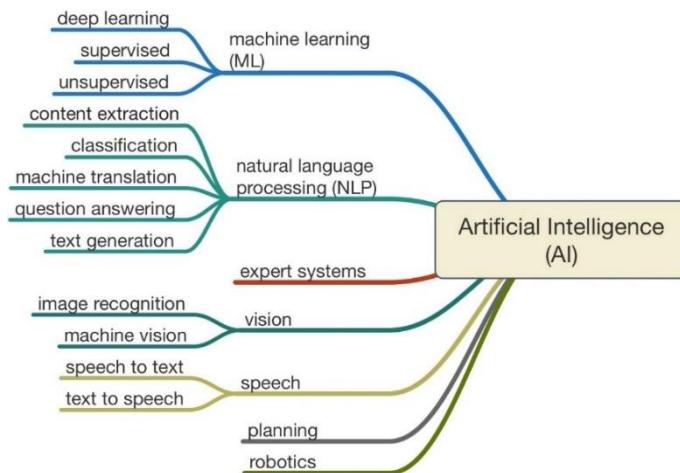
BAB V KESIMPULAN DAN SARAN

Pada bab ini diisi dengan kesimpulan dan semua hasil penelitian yang menjawab tujuan penelitian ini dibuat. Selain itu, pada bab ini diisi dengan saran untuk penelitian dan pengembangan selanjutnya berdasarkan pada hasil dan kesimpulan yang diperoleh pada penelitian ini.

BAB II

TINJAUAN PUSTAKA

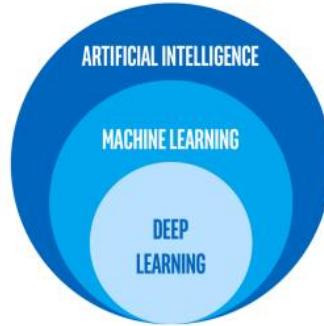
2.1 *Artificial Intelligence*



Gambar 2.1 Bagian dari Artificial Intelligence[7]

Artificial intelligence adalah bidang ilmu terapan yang mempelajari dan melakukan eksperimen dan menerapkan tentang kecerdasan, kecerdasan yang dilakukan oleh mesin yang meniru kecerdasan makhluk hidup seperti hewan dan manusia yang berarti kemampuan mesin untuk melakukan sesuatu untuk mencapai sebuah tujuan[1]. Konsep dari *artificial intelligence* pertama kali diperkenalkan oleh *John McCarthy* pada tahun 1956 di *Dartmouth College*, Amerika Serikat[8].

Artificial intelligence mulai populer pada tahun 1990-an, seiring dengan semakin banyaknya data yang diciptakan manusia dan masalah-masalah didalamnya[9]. Pada awal 2000-an, konsep *artificial intelligence* mulai bergeser dari tempatnya, pada awalnya *artificial intelligence* hanya penelitian-penelitian di bidang teori komputer sains, namun seiring semakin banyaknya penelitian-penelitian, kemudian *artificial intelligence* mulai mendapat tempatnya di bidang matematika, statistika, ekonomi dan bahkan fisika[9].



Gambar 2.2 Bagian dari *Artificial Intelligence*[10]

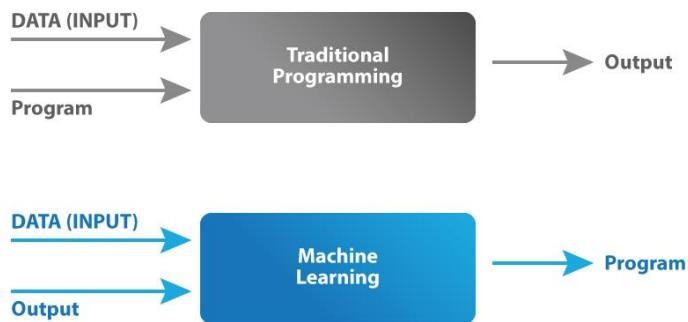
Sehingga untuk saat ini, *artificial intelligence* sangat luas sekali maknanya, bukan hanya kecerdasan buatan saja, tetapi sudah ke dalam beberapa bidang seperti *machine learning*, *deep learning*, *neural network* (jaringan syaraf tiruan), *data science*, logika *fuzzy*, robotika dan *IoT*, *bot* bahkan sampai ke *game*.

Beberapa contoh hal-hal yang menggunakan sistem *artificial intelligence* adalah seperti pendekripsi suara (*google assistant*, *siri*), sistem automasi berbasis robotik atau *IoT*, sistem rekomendasi tempat, film, musik dan lain sebagainya.

2.2 *Machine Learning*

Machine learning adalah salah satu dari cabang *artificial intelligence* yang mempelajari tentang algoritma komputer yang berkembang berdasarkan *experience* (seperti belajar pada manusia) melalui data. Dalam hal lain, machine learning adalah algoritma yang merancang dan mengembangkan perilaku komputer berdasarkan data dan statistik, sehingga komputer dapat mengenali pola kompleks dari data dan membuat *decision* berdasarkan data[11].

Alur pemrograman machine learning berbeda dari pemrograman tradisional, dimana pemrograman tradisional membutuhkan *input* dan *rules* untuk memperoleh *output*, sedangkan pemrograman *machine learning* membutuhkan *input* dan *output* untuk memperoleh *rules*[6].



Gambar 2.3 Alur Pemrograman Biasa dengan *Machine Learning*[12]

Algortima *machine learning* membangun model berdasarkan dari sampel data atau yang biasa disebut data latih. Data tersebut digunakan untuk program membuat prediksi atau *decision* secara otomatis tanpa harus membuat menulis kode pemrograman yang panjang, sehingga akan menghemat waktu dan lebih produktif[13].

Ada 2 jenis data di dalam *machine learning*, jenis-jenis data tersebut berbeda cara pengolahannya, bergantung pada bagaimana struktur datanya, adapun jenis datanya ada, data terstruktur seperti data regresi, data logistik, dan data yang disimpan dalam tabel. Kemudian ada data tidak terstruktur seperti data gambar, suara, video dan data yang tidak berbentuk tabel dan tidak dapat disimpan dalam bentuk tabel. Semua itu dikumpulkan menjadi satu membentuk sebuah set dari data atau disebut dataset

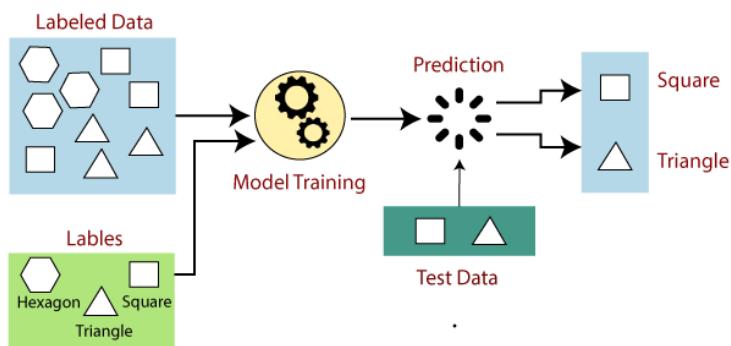
Dataset adalah kumpulan dari sebuah data-data yang dikumpulkan dalam satu tempat. Dataset dapat berupa gambar, teks, audio, video ataupun data-data regresi. Dataset biasanya memiliki format *SQL*, *CSV* atau *XSLX*, tetapi ada juga dataset yang memiliki format gambar, audio ataupun video[14], [15].

Dataset berupa gambar, teks, audio dan video merupakan dataset yang berbentuk unstructured data, artinya data tersebut tidak terstruktur ke dalam sebuah format tabel dan berbentuk abstrak, sedangkan untuk dataset seperti tabel,

regresi, time-series merupakan structured data, artinya data tersebut terstruktur ke dalam sebuah format tabel[16].

Adapun setiap data mempunya metode dalam pengolahannya, metode yang banyak digunakan *machine learning* ada 3, metode-metode itu antara lain adalah:

a. *Supervised Learning*



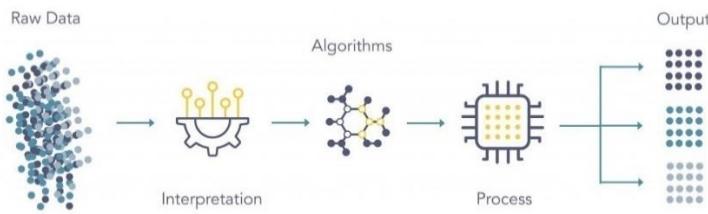
Gambar 2.4 *Supervised Learning*[17]

Supervised learning adalah metode dalam *machine learning* yang memberikan label pada data, sehingga komputer dapat mempelajari dan mengidentifikasi pola dari data melalui label dengan menggunakan fitur yang ada untuk melakukan prediksi maupun klasifikasi. Dengan mengamati data-data tersebut, komputer ini akan menghasilkan sebuah model yang mampu memprediksi atau mengklasifikasikan input data yang baru dan belum dipelajari oleh model sebelumnya menjadi output yang tepat dan akurat[9].

Contoh implementasi metode *supervised learning* adalah pada kasus pendekripsi berita *hoax* yang ada di sosial media. Data latih diberi label berupa *hoax* dan bukan *hoax*. Kemudian komputer akan mempelajari data-data tersebut melalui proses *learning* dan mencari pola dari data tersebut melalui label yang diberikan sehingga dapat menghasilkan output berupa pengelompokan berita yang *hoax* atau bukan *hoax*.

Contoh algoritma yang menggunakan metode *supervised learning* adalah fitting data regresi (*regression*), klasifikasi dan *decision making* dengan *logistic regression*, *SVM*, *decision tree*, *random forest*, KNN, *naïve bayes*, dan *gradient descend*.

b. *Unsupervised Learning*



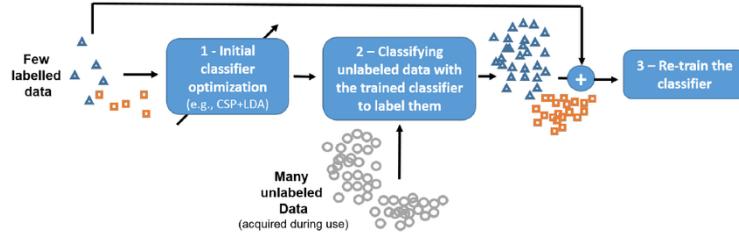
Gambar 2.5 Unsupervised Learning[18]

Metode ini bertujuan untuk membuat komputer mencari pola dari sebuah data dikarenakan data tersebut tidak mempunyai label sehingga komputer dapat mempelajari data-data tersebut berdasarkan fitur dan pola-pola dari data yang ada. Metode ini tidak memiliki target seperti halnya *supervised learning* dan metode ini bertujuan untuk mengelompokkan objek yang mempunyai pola, fitur dan ciri-ciri yang mirip atau disebut *clustering*[9].

Contoh *unsupervised learning* adalah komputer diberikan beberapa data berupa pasien yang pernah mengecek gula darah di sebuah rumah sakit, kemudian dari data tersebut komputer belajar dan mengelompokan data tersebut dan dari data yang dikelompokan tersebut kemudian dokter akan menyimpulkan dari kelompok mana saja yang terkena diabetes atau tidak.

Contoh algoritma yang digunakan dalam metode ini adalah clustering (pengelompokan) dengan menggunakan *K-means* dan PCA.

c. *Semi-Supervised Learning*

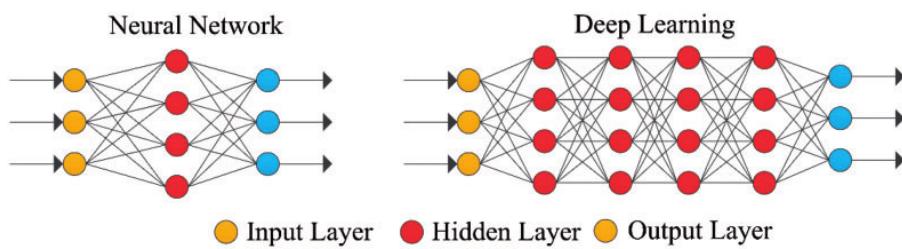


Gambar 2.6 *Semi-Supervised Learning*[19]

Metode ini menggabungkan *supervised learning* dan *unsupervised learning* di mana beberapa data diberikan label, dan beberapa data lainnya tidak diberikan label, sehingga komputer harus mencari pola dari data tersebut melalui fitur-fiturnya dan pola dari data yang mempunyai label[9].

Contoh dari metode *semi-supervised learning* adalah identifikasi wajah seseorang yang berada di *smartphone*, diberikan label yang sangat sedikit tentang siapa dari pemilik *smartphone* tersebut, sehingga komputer harus belajar untuk menebak beberapa wajah yang berada di *smartphone* tersebut dan menebak mana yang merupakan pemilik *smartphone* tersebut.

2.3 Deep Learning



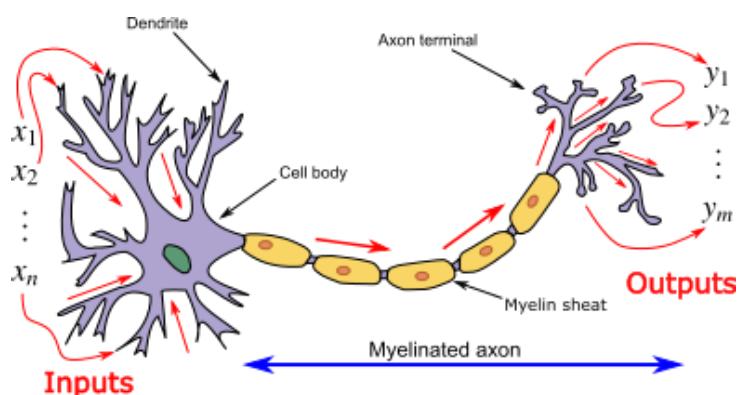
Gambar 2.7 *Deep Learning*[20]

Deep learning merupakan cabang dari *machine learning* yang merupakan cabang dari *artificial intelligence* yang menggunakan *neural network*[5]. *Deep learning* adalah versi lanjut-nya dari *machine learning* dan *artificial neural network*, dimana algoritma *deep learning* dapat mempelajari data *supervised*, *unsupervised*, *semi-supervised* dan *reinforcement learning*[21], [22]. Selain itu,

algoritma *deep learning* adalah bentuk lanjut dari *artificial neural network* dikarenakan banyaknya *neuron* dan *hidden layer* yang terkoneksi, sehingga algoritma tersebut dinamakan “*deep*” *learning* yang berarti pembelajaran dalam (karena banyaknya jumlah *neuron* dan *hidden layer* yang terkoneksi)[5], [21]

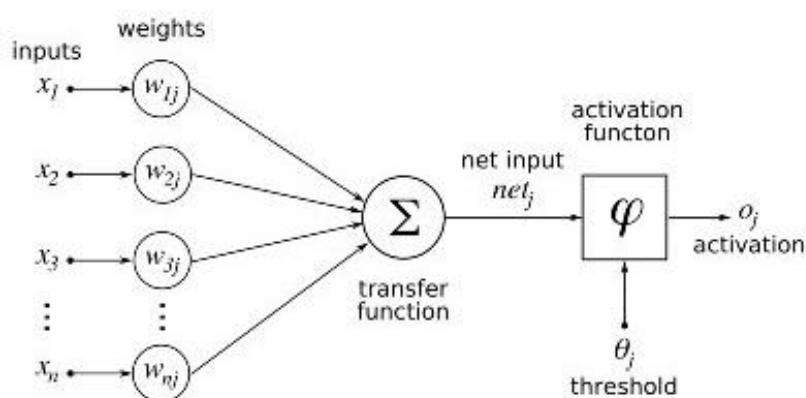
Sifat fisis dari algoritma *deep learning* adalah pada implementasi dari cara kerja *neuron* yang ada di jaringan syaraf yang ada di otak makhluk hidup (khususnya manusia) dalam mengolah informasi yang diberikan oleh syaraf, kemudian diproses oleh otak, dan kemudian diinterpretasikan oleh indera. Sehingga karena kemampuannya itu, dinamakan *artificial neural network* atau jaringan syaraf tiruan[23]. *Artificial neural network* juga dapat disebut dengan sistem jaringan yang adaptif dimana sistem tersebut dapat mengubah struktur jaringannya (sesuai dengan masalah yang dihadapi oleh *neural network*) untuk memecahkan masalah berdasarkan *input* yang telah diberikan yang mengalir melalui jaringan tersebut untuk mencapai *output* yang diinginkan[24].

Secara sederhana, *artificial neural network* adalah sebuah algoritma permodelan statistik yang berbentuk data *non-linier*. *Artificial neural network* digunakan untuk memodelkan hubungan antara *input* dan *output* dengan menggunakan neural network sehingga algoritma akan mengamati pola dari data kemudian belajar dari pola tersebut[25].



Gambar 2.8 Jaringan Syaraf Manusia[26]

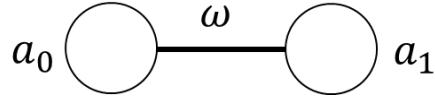
Cara kerja dari *artificial neural network* adalah sekumpulan dari unit atau titik yang saling terkoneksi satu sama lain yang meniru sistem syaraf yang ada di otak. Setiap koneksi dapat mengirim data ke *neuron* yang lain yang kemudian *neuron* yang lain menerima data dari *neuron* lainnya dan kemudian *neuron* itu memproses data yang dikirim dari *neuron* lainnya. Data yang diterima merupakan representasi dari bilangan *real* dan data yang dikirim merupakan data yang dihitung dengan total jumlah dari fungsi *non-linier* dari *input*-nya[4], [27], [28].



Gambar 2.9 Diagram *Artificial Neural Network*[29]

Pada dasarnya, setiap *neuron* hanya bisa menerima satu dimensi data dan mengeluarkan *output* satu dimensi data ke dalam *neuron* lain yang terkoneksi dengan *neuron* tersebut (prinsip ini dinamakan *perceptrons* dan pada awalnya hanya ada satu *neuron*). *Neuron* yang saling terkoneksi tersebut menghasilkan sebuah *weight* atas satu *neuron* yang saling terkoneksi, dengan semakin banyak *weight* yang terkoneksi, maka akan semakin banyak pula *weight*-nya. Untuk mencari nilai *output* dari *neuron*, pertama dihitung jumlah semua *weight*, kemudian dari fungsi tersebut ditambahkan *bias*. *Bias* berasal dari fungsi *propagation* yang menghitung *input* *neuron* dari *neuron* lain dan koneksinya berupa jumlah *weight* sehingga akan menghasilkan *bias*. Jumlah semua *weight* tersebut melewati fungsi *non-linear* yang disebut fungsi aktivasi[4].

Persamaan matematika sederhana untuk *neural network* yang terdiri dari satu *node* yaitu hanya input dan output adalah[30],



Gambar 2.10 Diagram *Neural Network* sederhana

$$a_1 = \sigma(\omega a_0 + b) \quad (1)$$

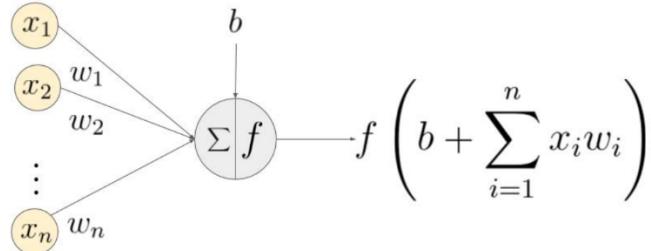
dengan:

a = neuron (input, output)

ω = weight (beban)

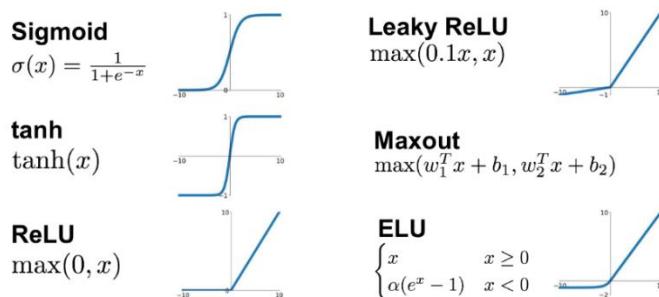
b = bias (error)

σ = fungsi aktivasi



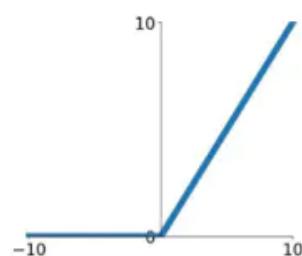
Gambar 2.11 Fungsi Aktivasi[31]

Dikarenakan *output* dari *neural network* merupakan *output* yang bukan *nonlinear*[4], sehingga dalam prosesnya dibutuhkan sebuah fungsi aktivasi yang akan membuat data yang dimasukan ke *neural network* tersebut menjadi *nonlinear*[32].



Gambar 2.12 Macam Fungsi Aktivasi[33]

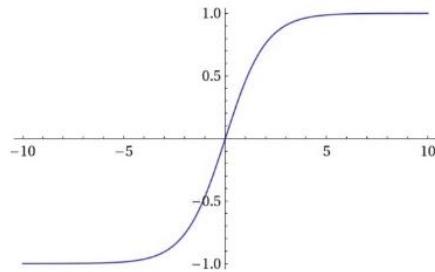
Ada beberapa fungsi aktivasi dalam *neural network*, dan setiap fungsi aktivasi mempunyai perannya masing-masing dalam memberikan *output* ke *neural network* yang lain. Yang biasa dipakai dalam pelatihan *neural network* adalah fungsi aktivasi *ReLU*, sedangkan fungsi aktivasi yang biasa dipakai untuk memberikan *output* akhir dari *neural network* bisa bermacam-macam bergantung dari data *input* dan data *output* yang diberikan, berikut adalah penjelasan singkat tentang fungsi aktivasi,



Gambar 2.13 Grafik *ReLU*[34]

ReLU atau *Rectified Linear Unit* merupakan fungsi aktivasi yang sering dipakai dalam *neural network* untuk memberikan *input* dan menghasilkan *output* ke *neural network* lainnya. Fungsi ini membuat nilai x (x merupakan data yang diberikan terhadap *neural network*) menjadi bernilai untuk $x > 0$ dan akan bernilai 0 untuk $x < 0$ [35], [36]. Berikut adalah persamaan untuk fungsi aktivasi ini,

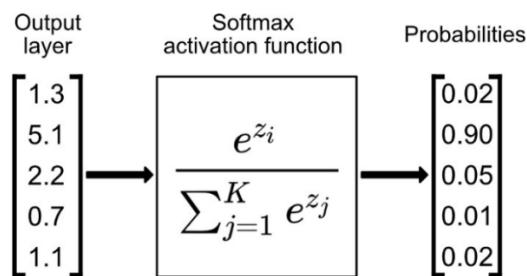
$$f(x) = x^+ = \max(0, x) \quad (2)$$



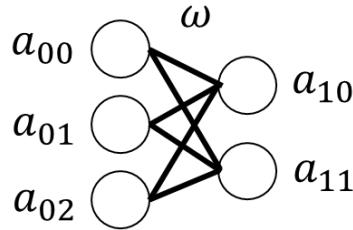
Gambar 2.14 Grafik *Softmax*[37]

Selanjutnya ada fungsi aktivasi *softmax*, fungsi aktivasi *softmax* ini merupakan fungsi aktivasi yang banyak digunakan untuk klasifikasi dengan *multilabel*. Fungsi aktivasi ini berguna untuk menghasilkan *output* probabilitas di tiap-tiap label. Fungsi aktivasi ini berfungsi untuk me-*nonlinear* kan data di layer *output neural network*[36], [38]. Berikut adalah persamaan untuk fungsi aktivasi ini dan illustrasi bagaimana fungsi aktivasi ini bekerja,

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (3)$$



Gambar 2.15 Bagaimana *Softmax* Bekerja[39]



Gambar 2.16 Contoh Diagram Neural Network yang mempunyai banyak nodes

Kemudian untuk neuron yang mempunyai banyak *nodes* dan hidden layer dapat disimpulkan dengan menggunakan matriks, sehingga bentuk dari persamaannya adalah[30],

$$\begin{bmatrix} a_{00} \\ \dots \\ a_{m-1} \end{bmatrix} = \sigma \begin{bmatrix} \omega_{0,0} & \dots & \omega_{0,n-1} \\ \vdots & \ddots & \vdots \\ \omega_{m-1,0} & \dots & \omega_{m-1,n-1} \end{bmatrix} \begin{bmatrix} a_{00} \\ \dots \\ a_{n-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ \dots \\ b_{m-1} \end{bmatrix} \quad (4)$$

Dengan menggunakan notasi matriks, bentuknya dapat disederhanakan menjadi,

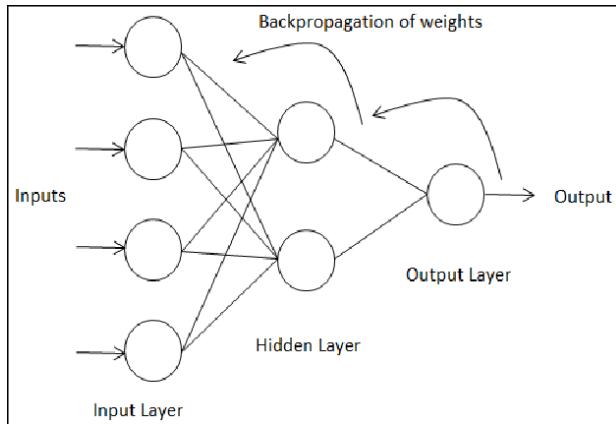
$$\mathbf{a}_1 = \sigma(\boldsymbol{\omega}\mathbf{a}_0 + \mathbf{b}) \quad (5)$$

Kemudian, untuk mempermudah, persamaan dapat ditulis dengan memisalkan nilai di dalam kurung tersebut ke dalam variable baru yaitu variabel z. Sehingga,

$$\mathbf{z} = (\boldsymbol{\omega}\mathbf{a}_0 + \mathbf{b}) \quad (6)$$

$$\mathbf{a}_1 = \sigma(\mathbf{z}) \quad (7)$$

2.3.1 Backpropagation



Gambar 2.17 Diagram *Backpropagation*[40]

Setelah itu, pelatihan dilakukan. Ada beberapa algoritma seperti *backpropagation*, *backpropagation* dipakai untuk melatih model *neural network* yang berbentuk *feedforward neural network*[2]. *Backpropagation* menggunakan *multilayer perceptron* dalam pelatihannya[41].

Feedforward neural network adalah jenis *neural network* dimana *neural network* yang saling terkoneksi akan menciptakan *nodes* (titik) dimana titik tersebut tidak akan membentuk sebuah *loop* atau tidak berputar-putar dan akan terus terkoneksi sampai mencapai *output*[2]. *Feedforward neural network* adalah model *neural network* pertama dan merupakan model yang paling sederhana[5].

Algoritma *backpropagation* merupakan algoritma deep learning yang paling sederhana[5], berfungsi untuk menghitung *gradient* dari *weights* yang di update untuk memperkecil *loss* dan memperbesar akurasi[41]. *Backpropagation* hanya bisa digunakan untuk data yang berbentuk *supervised*[42].

Selain itu, algoritma *backpropagation* ini juga memiliki beberapa komponen (*hyperparameter*) yang membuat performa dari algoritma *backpropagation*, walaupun merupakan model *neural network* yang paling sederhana, tetapi performa algoritma *backpropagation* dalam klasifikasi akan baik

sekali (jika menggunakan *hyperparameter* yang tepat bergantung dari dataset), adapun *hyperparameter* dalam algoritma *backpropagation* antara lain:

a. *Optimizer*

Optimizer adalah algoritma yang berfungsi untuk mengoptimasi *neural network* seperti memberikan *learning rate* untuk memperkecil *loss function*[43].

b. *Learning Rate*

Learning rate merupakan salah satu algoritma *optimizer* yang berfungsi untuk mengatur seberapa besar perubahan yang dilakukan oleh model terhadap *error* atau *loss function* pada saat setiap kali *weights* pada model diperbarui[44]. *Learning rate* idealnya berada pada rentang $0 \leq lr \leq 1$. Semakin kecil *learning rate* maka pelatihan akan berlangsung lama dan semakin besar maka pelatihan akan semakin cepat, jadi penting untuk mengatur supaya *learning rate* tidak begitu besar dan tidak begitu kecil dan tidak mempengaruhi *loss* untuk semakin besar[43], [44].

c. *Loss Function*

Loss function merupakan sebuah algoritma untuk mengevaluasi bagaimana performa dari model untuk mempelajari sebuah dataset[32].

Backpropagation mempunyai persamaan matematika. Persamaan dari *backpropagation* adalah sebagai berikut[30],

$$C = (a_1 - y)^2 \quad (8)$$

dengan:

C = Laju *Backpropagation*

a_1 = Output

y = *hyperparameter*

Karena laju, maka ada perubahan, maka digunakan pers differensial, karena yang berubah tiap layer itu adalah *weights* (omega) dan bias, maka laju

tersebut diturunkan terhadap weights dan bias. Dengan menggunakan metode *chain rule*, maka persamaanya[30]:

Untuk turunan terhadap *weights* (beban),

$$\frac{\partial \mathcal{C}}{\partial \omega} = \frac{\partial \mathcal{C}}{\partial a_1} \frac{\partial a_1}{\partial \omega} \quad (9)$$

$$\frac{\partial \mathcal{C}}{\partial \omega} = \frac{\partial \mathcal{C}}{\partial a_1} \frac{\partial a_1}{\partial z} \frac{\partial z}{\partial \omega} \quad (10)$$

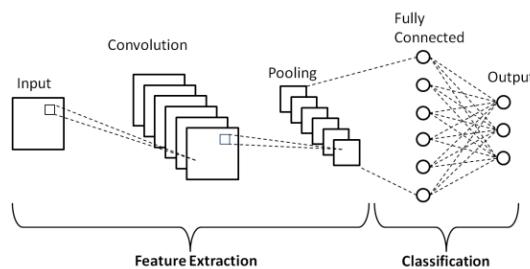
Untuk turunan terhadap *bias* (error),

$$\frac{\partial \mathcal{C}}{\partial b} = \frac{\partial \mathcal{C}}{\partial a_1} \frac{\partial a_1}{\partial b} \quad (11)$$

$$\frac{\partial \mathcal{C}}{\partial b} = \frac{\partial \mathcal{C}}{\partial a_1} \frac{\partial a_1}{\partial z} \frac{\partial z}{\partial b} \quad (12)$$

2.3.2 Convolutional Neural Network

Untuk melatih data yang berisi gambar, maka akan sangat sulit untuk hanya menerapkan metode *backpropagation*, dikarenakan pada metode *backpropagation*, *neuron* yang terkoneksi masih didalam satu dimensi, oleh karena itu perlu ada metode lain untuk mengolah data gambar, salah satu metode yang bisa digunakan adalah metode *convolutional neural network*[6], [41], [45].



Gambar 2.18 Diagram *Convolutional Neural Network*[46]

Convolutional neural network adalah salah satu metode *deep learning* yang bersifat *multilayer perceptron* yang di-desain untuk mengolah data yang berukuran dua dimensi khususnya data pada gambar[41].

Untuk mengubah gambar menjadi format yang dapat dikenali oleh backpropagation neural network, convolutional neural network menggunakan proses *feature extraction*. *Feature extraction* merupakan proses untuk mengubah struktur dari sebuah data supaya dapat dikenali oleh neural network kemudian untuk menghemat sumber daya dan performa untuk pelatihan *neural network* serta untuk membuat model *neural network* tidak *overfit* dikarenakan pada dasarnya, *neural network* merupakan proses yang terjadi pada data satu dimensi[47], [48]

Proses ini (pada kasus *convolutional neural network*) mengubah data dua dimensi kedalam data bentuk satu dimensi (yaitu berupa *array*) agar dapat diproses kedalam *backpropagation neural network*[48].

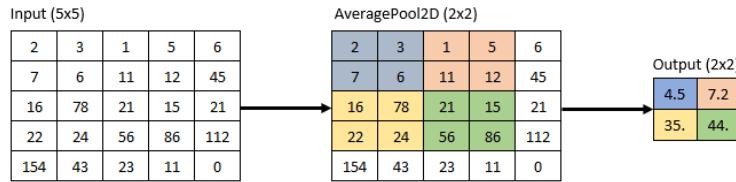
Feature extraction mempunyai beberapa layer, beberapa layer yang dimiliki oleh *convolutional neural network* antara lain adalah:

a. *Transfer Learning*

Transfer learning adalah *layer* dalam deep learning khususnya dalam *convolutional neural network* yang memanfaatkan *pre-trained* model (model yang sudah pernah dilatih dengan dataset yang berbeda sebelumnya) terhadap dataset baru yang kemudian *pre-trained* model tersebut dimodifikasi dan diubah beberapa parameter model tersebut supaya sesuai dengan dataset yang baru, sehingga *neural network* dapat mempelajari dataset yang baru[41].

b. *AveragePooling*

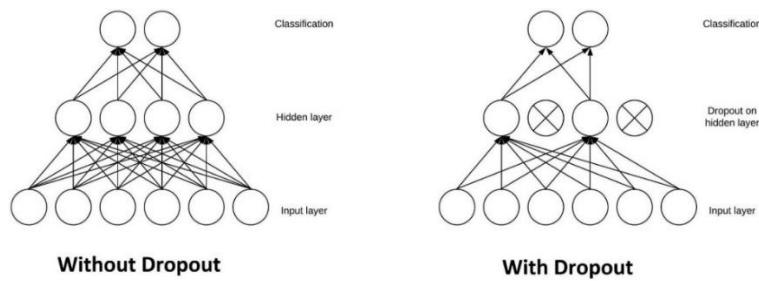
AveragePooling adalah *layer* dalam *convolutional neural network* pengurangan ukuran matriks dengan menggunakan operasi *pooling* dengan mengambil nilai rata-rata dari setiap *pool_size* (batasan ukuran matriks yang akan diambil, dalam kasus ini akan diambil 2x2 dari setiap matriks yang ada) dari matriks tersebut[32], [41]. Illustrasi dari *AveragePooling* adalah sebagai berikut:



Gambar 2.19 Illustrasi *AveragePool2D*

c. *Dropout Layer*

Kemudian adalah *Dropout* layer, yang fungsinya untuk membuang atau mematikan beberapa *neuron* yang ada di dalam *hidden layer*, sehingga akan bermanfaat untuk mencegah *overfitting* dan mempercepat proses latih[32], [41].



Gambar 2.20 Bagaimana *Dropout* Bekerja[49]

d. Augmentasi Data

Augmentasi data adalah layer yang memproses suatu proses dalam pengolahan data gambar. Augmentasi merupakan proses mengubah atau memodifikasi gambar sedemikian rupa sehingga komputer akan mendeteksi bahwa gambar yang diubah adalah gambar yang berbeda, namun manusia masih dapat mengetahui bahwa gambar yang diubah tersebut adalah gambar yang sama[50], [51].

Sehingga kesimpulan mengapa digunakan algoritma deep learning ini adalah, karena deep learning dapat mempelajari data yang tak terstruktur seperti gambar, video, suara dan teks dengan performa yang baik[52].

2.4 Tools

Berikut adalah penjelasan dari *tools* atau program (*library*, *framework*) apa saja yang akan digunakan untuk penelitian. Berikut adalah *tools*-nya,

2.4.1 Python

Python[53] adalah bahasa pemrograman yang diciptakan oleh *Guido van Rossum* pada tahun 1991 dan bersifat *open source*. Bahasa ini merupakan bahasa pemrograman tingkat tinggi, yang berarti bahasa ini tidak terkoneksi langsung oleh mesin dan memerlukan perantara untuk berkomunikasi oleh mesin. Oleh karena itu, bahasa ini menggunakan metode interpretasi, dimana untuk menjalankan *script* dari bahasa ini tidak perlu untuk men-*compile script*-nya terlebih dahulu. Untuk berkomunikasi dengan mesin, bahasa ini diperantari melalui bahasa pemrograman C. Bahasa ini merupakan bahasa berorientasi objek, dimana seluruh atribut pada bahasa ini adalah merupakan turunan (*inherit*) dari *class object*[54], [55].

Bahasa ini banyak digunakan oleh saintis, terutama saintis yang melakukan riset khususnya di bidang fisika, matematika, data sains dan kecerdasan buatan (*artificial intelligent*). Walaupun bahasa ini tergolong bahasa dengan performa lamban, tetapi modul dan *library*-nya sangat lengkap. Selain itu, bahasa ini merupakan bahasa yang mudah untuk dipahami, karena *syntax*-nya sederhana dan mendekati bahasa manusia, walaupun bahasa ini tidak menggunakan kurung kurawal sebagai pembuka dan penutup *scope* melainkan menggunakan indentasi yang tentu saja akan membuat bingung orang yang baru saja mempelajari bahasa ini.

2.4.2 NumPy

NumPy[56] merupakan library *open source* dari bahasa pemrograman *python* yang ditulis oleh *Travis Oliphant* pada tahun 1995. *Library* ini berfungsi untuk melakukan kalkulasi matematika dengan menggunakan dimensi dan kecepatan tinggi seperti *multi-dimensional array*, melakukan operasi vektor,

matriks dan *tensor*, serta melakukan segala sesuatu yang berhubungan dengan perhitungan, mulai dari perhitungan dasar sampai perhitungan kompleks[57].

Library ini ditulis dengan menggunakan bahasa C sebagai *low-level API* dan menggunakan bahasa *python* sebagai *high-level API*. *Library* ini diciptakan karena tipe data di bahasa *python* untuk kumpulan data seperti *list*, *dictionary*, *tuple*, *set* dan *iterators* sangat lambat dalam kalkulasi data dengan dimensi tinggi dan *python* juga tidak mempunya *array* yang akan sangat membantu untuk mempercepat kalkulasi data dimensi tinggi, sehingga *library* ini diciptakan[57].

2.4.3 TensorFlow dan Keras

Keras[58] merupakan *library open source* dari bahasa pemrograman *python* yang ditulis oleh author pertamanya adalah *Francois Chollet* pada tahun 2015. *Library* ini ditulis sebagai *interface (high-level API)* dari *TensorFlow* untuk bahasa pemrograman *python*[6].

Tensorflow[59] merupakan *library open source* dari bahasa pemrograman *python* yang ditulis oleh *Google Brain Team* pada tahun 2015 dengan versi *alpha* nya dimulai pada tahun 2011. *Library* ini ditulis dengan bahasa pemrograman *python* sebagai *high-level API* (melalui *keras*) dan C dan CUDA sebagai *low-level API* dan tersedia dengan berbagai macam bahasa selain *python* seperti *java*, *javascript*, dan untuk perangkat *embedded system* atau *low power system* seperti *raspberry* dan *smartphone* melalui *tensorflow lite*. *Library* ini berfungsi untuk melakukan hal-hal yang berkaitan dengan *machine learning* dan *deep learning*, seperti memproses data, melakukan prediksi, klasifikasi, melatih model dan lain sebagainya[6], [57].

Library ini memungkinkan penggunanya untuk melatih *model neural network* dengan menggunakan teknik *multithreading* (bekerja dengan berbagai macam sumber daya seperti *CPU (Central Processing Unit)*, *GPU (Graphics Processing Unit)*, *TPU (Tensor Processing Unit)* dan *NPU (Neural Network Processing Unit)* secara bersamaan) sehingga akan mempercepat proses pelatihan.

2.4.4 *Pandas*

Pandas[61], [62] merupakan *library open source* yang ditulis dengan bahasa pemrograman *Python* dengan *author* pertamanya adalah *Wes McKinney* yang ditulis pada tahun 2008[61], [62].

Pandas merupakan singkatan dari “*Python Data Analysis*”, merupakan kumpulan *class* beserta *method*-nya yang biasanya digunakan untuk analisis data dan manipulasi data di bahasa pemrograman *Python* yang berbentuk data tabel. *Pandas* dapat membaca data dari *CSV*, *SQL* (*databases*), dan *NoSQL* (*JSON*, *XML*) dan mengubahnya ke dalam bentuk yang lebih bisa dibaca yaitu tabel[61].

2.4.5 *Matplotlib*

Matplotlib[63], [64] merupakan *library open source* yang ditulis dengan bahasa pemrograman *Python* dengan *author* pertamanya adalah *John D. Hunter* pada tahun 2003[63].

Matplotlib merupakan *library* yang menyediakan *tools-tools* untuk visualisasi data dengan menggunakan *GUI toolkits*_seperti *Tkinter*, *wxPython*, *Qt*, dan *GTK*. *Library* ini juga disebut *pylab* karena *library* ini merupakan implementasi dari *tools-tools* untuk *plotting* yang ada di *Matlab*[63].

2.4.6 *Scikit-Learn*

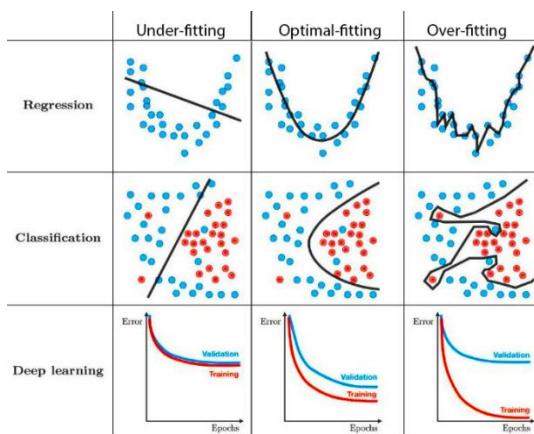
Scikit-Learn[65], [66] merupakan *library open source* yang ditulis dengan bahasa pemrograman *Python*, *C* dan *C++* yang ditulis oleh *author* pertamanya *David Cournapeau* pada tahun 2007[65].

Library ini merupakan *tools-tools* dari *machine learning* mulai dari *preprocessing* sampai proses *evaluation* nya. *Library* ini mempunya fitur-fitur untuk membuat model *machine learning* seperti klasifikasi, regresi, clustering dan beberapa *tools* seperti *preprocessing dataset*, *normalization data*, *transforming data* dan beberapa *tools* lainnya[65].

2.5 Evaluasi

2.5.1 Overfitting dan Underfitting

Overfitting adalah data yang hanya bisa mempelajari satu atau beberapa jenis data dan gagal dalam mempelajari data yang lain[67], sedangkan *underfitting* dimana model hanya sedikit mempelajari data[67]. Berikut adalah ilustrasi dari data *underfitting* dan *overfitting*



Gambar 2.21 Ilustrasi overfitting dan underfitting[68]

2.5.2 Confusion Matrix

Confusion matrix adalah perbandingan hasil klasifikasi model terhadap sebuah data atau dataset dimana *confusion matrix* tersebut akan membandingkan berapa banyak data yang benar di prediksi, dan berapa banyak data yang salah di prediksi[69].

True positive dan *true negative* adalah *metrics* dimana model memprediksi data test secara benar sesuai dengan label pada data test, dimana maksud *positive* dan *negative* disini adalah membandingkan dua buah label, misalnya ada label 1 yang menyatakan bahwa pasien itu tidak positif kanker dan diprediksi tidak positif kanker itu bisa dikatakan *true positive*. Kemudian jika ada pasien dengan positif kanker dengan label 0, lalu model memprediksi bahwa pasien tersebut positif kanker, itu bisa dikatakan *true negative*.

		Actual Values	
		1 (Positive)	0 (Negative)
Predicted Values	1 (Positive)	TP (True Positive)	FP (False Positive) <small>Type I Error</small>
	0 (Negative)	FN (False Negative) <small>Type II Error</small>	TN (True Negative)

Gambar 2.22 Contoh Confusion Matrix[70]

False positive dan *false negative* adalah *metrics* dimana model memprediksi data test secara salah dan tidak sesuai dengan label pada data test, dimana maksud *positive* dan *negative* disini adalah membandingkan dua buah label, misalnya ada label 1 yang menyatakan bahwa pasien itu tidak positif kanker tetapi diprediksi positif kanker itu bisa dikatakan *false negative*. Kemudian jika ada pasien dengan positif kanker dengan label 0, lalu model memprediksi bahwa pasien tersebut tidak positif kanker, itu bisa dikatakan *false positive*.

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

$$\text{precision} = \frac{TP}{TP + FP} \quad (14)$$

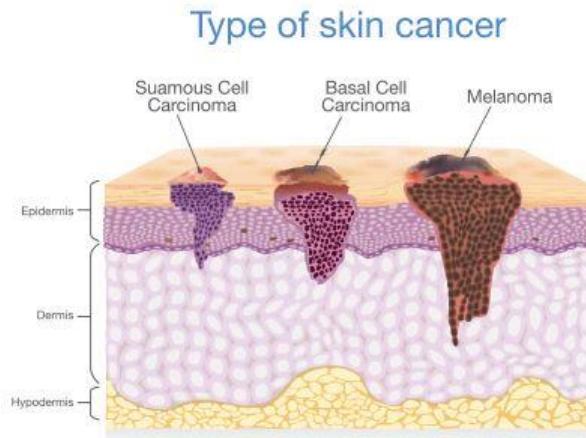
$$\text{recall} = \frac{TP}{TP + FN} \quad (15)$$

Accuracy merupakan *metrics* yang menilai berapa persen model mengklasifikasikan *true positive* dan *true negative* yang artinya seberapa persen model dapat mengklasifikasikan data yang bernilai benar secara tepat[67].

Precision merupakan *metrics* yang menilai berapa persen model mengklasifikasikan *true positive* dengan *false positive* sehingga akan dinilai seberapa persen model dapat mengklasifikasikan data yang bernilai negatif menjadi positif, sehingga akan diukur seberapa persen dari data yang seharusnya bernilai negatif menjadi positif[67].

Recall merupakan *metrics* yang menilai berapa persen model mengklasifikasikan *true positive* dengan *false negative* sehingga akan dinilai seberapa persen model dapat mengklasifikasikan data yang bernilai positif menjadi negatif, sehingga akan diukur seberapa persen dari data yang seharusnya bernilai positif menjadi negatif[67].

2.6 Kanker Kulit

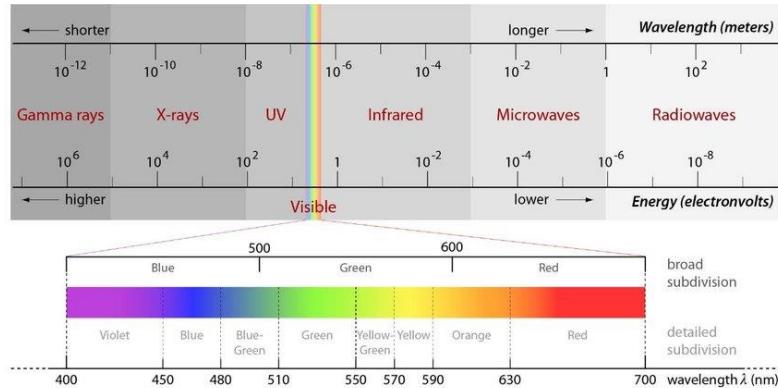


Gambar 2.23 Kanker Kulit[71]

Kanker merupakan penyakit yang disebabkan oleh pertumbuhan sel yang abnormal yang pertumbuhannya mempunyai kemampuan untuk menyebar ke bagian sel yang lain[72].

Salah satu penyakit kanker yang banyak diidap oleh orang yang tinggal di daerah yang setiap tahun dilewati oleh sinar matahari (seperti daerah khatulistiwa seperti Indonesia) salah satunya adalah kanker kulit. Kanker ini menyerang bagian

kulit yang terkena langsung oleh sinar *ultraviolet* oleh matahari. Karena lebih dari 90% kasus kanker kulit berasal dari terpaparnya kulit secara langsung oleh sinar matahari[73].



Gambar 2.24 Spektrum Gelombang Elektromagnetik[74]

Sinar *ultraviolet* adalah salah satu sinar yang diciptakan oleh radiasi elektromagnetik yang berasal dari matahari, panjang gelombang dari sinar ini adalah 400 nm sampai 10 nm dengan frekuensi 750 THz sampai 30 PHz[75].

Terdapat dua kategori kanker kulit berdasarkan tingkat keganasannya[76]. Pertama adalah kanker kulit yang bersifat *benign*, kanker kulit ini bersifat tidak ganas dan tidak mematikan sehingga dapat disembuhkan jika rutin konsultasi ke dokter. Kedua adalah kanker kulit yang bersifat *malignant*, kanker kulit ini bersifat ganas dan mematikan dan dapat dengan cepat menyebar ke bagian tubuh yang lain, kanker ini dapat disembuhkan tetapi dengan pengobatan khusus tentunya[76], [77].

2.7 Penelitian Sejenis

Tabel 2.1 Penelitian Sejenis

No	Author	Judul	Publish
1	Esteva, A., Kuprel, B., Novoa, R.	Dermatologist-level classification of skin cancer with deep neural networks	Januari 2017
2	K. M. Hosny, M. A. Kassem, M. M. Foaud	Skin Cancer Classification using Deep Learning and Transfer Learning	Desember 2018
3	A. Dascalu, E.O. David	Skin Cancer Classification using Deep Learning and Transfer Learning	April 2019
4	Jinnai S, Yamazaki N, Hirano Y, Sugawara Y, Ohe Y, Hamamoto R	The Development of a Skin Cancer Classification System for Pigmented Skin Lesions Using Deep Learning	Juli 2020
5	Teck Yan Tan, Li Zhang, Chee Peng Lim	Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models	November 2019
6	C. Barata and J. S. Marques	Deep Learning For Skin Cancer Diagnosis With Hierarchical Architectures	April 2019
7	Aditya Khamparia, Prakash Kumar Singh, Poonam Rani, Debabrata Samanta, Ashish Khanna, Bharat Bhushan	An internet of health things-driven deep learning framework for detection and classification of skin cancer using transfer learning	Mei 2020
8	Amirreza Rezvantalab, Habib Safigholi, Somayeh Karimijeshni	Dermatologist Level Dermoscopy Skin Cancer Classification Using Different Deep Learning Convolutional Neural Networks Algorithms	Oktober 2018
9	Ahmet Demir, Feyza Yilmaz, Onur Kose	Early detection of skin cancer using deep learning architectures: resnet-101 and inception-v3	Oktober 2019
10	Andre G. C. Pacheco, Renato A. Krohling	Recent advances in deep learning applied to skin cancer detection	Desember 2019

BAB III

METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

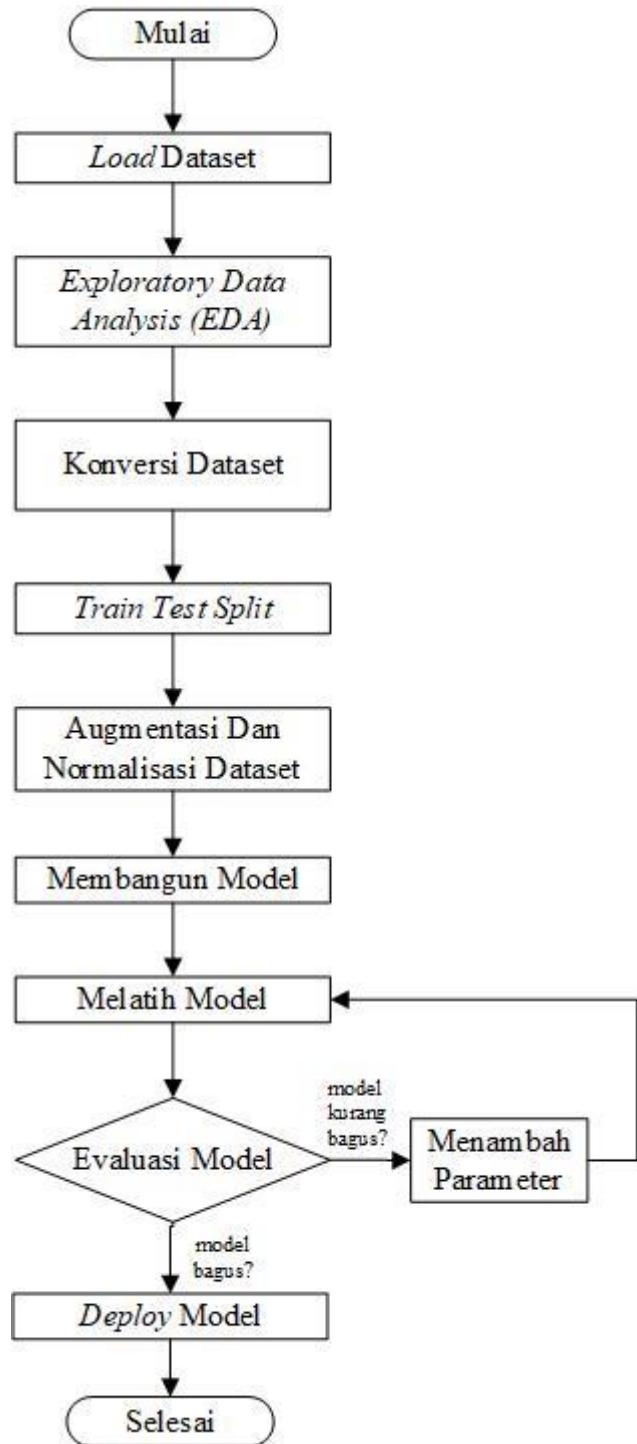
Penelitian ini dilakukan pada bulan Juli 2021 di Pusat Laboratorium Terpadu (PLT) Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta.

3.2 Peralatan

Alat yang digunakan dalam eksperimen ini adalah:

1. Laptop dengan sistem operasi *Windows 11*.
2. *Cloud Server Google Colab* dengan Spesifikasi:
 - a. *n1-highmem-2 instance*
 - b. *Hyper Threaded Xeon Processors @2.3Ghz (1x2 Threads)*
 - c. 13GB RAM
 - d. *Tesla K80 (2496 CUDA cores, 12GB GDDR5 VRAM)*
 - e. *Ubuntu 18.04 LTS*
3. *Python 3.8*
4. *Numpy 1.19.5*
5. *Tensorflow 2.5.0*
6. *Pandas 1.3.4*
7. *Matplotlib 3.4.3*
8. *Scikit-Learn 1.0*

3.3 Diagram Alur Penelitian



Gambar 3.1 Diagram Alur Penelitian

3.4 Pemrosesan Dataset

3.4.1 Exploratory Data Analysis (EDA)

Dataset yang digunakan disini adalah dataset dengan format file *CSV* dan gambar *JPEG* yang berasal dari *Kaggle* dengan nama repositori *Skin Cancer MNIST HAM10000* dan *K Scott Mader* sebagai *owner repository*[78].

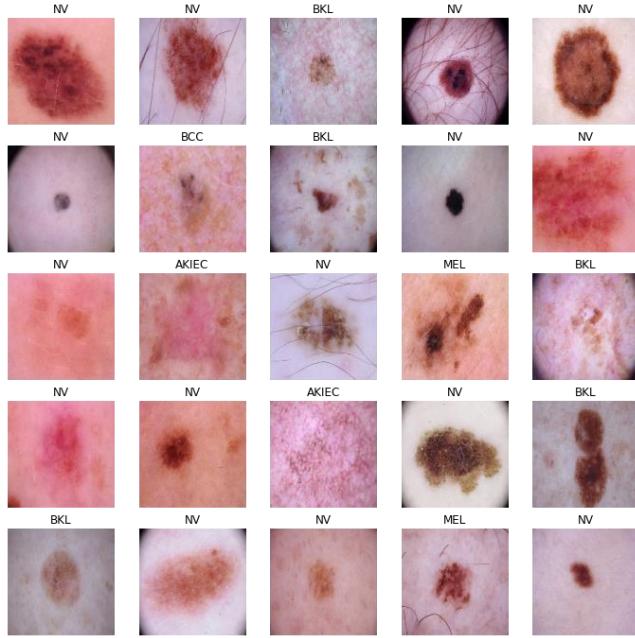
Dataset berisi 10.015 gambar dengan format *JPEG* dan beresolusi 1872x1053 piksel dengan 7 kategori kanker kulit yaitu:

Tabel 3.1 Jumlah Data dan Label

Nama	Jumlah Data
<i>Melanocytic Nevi</i>	6.705
<i>Melanoma</i>	1.113
<i>Benign Keratosis-Like Lesions</i>	1.099
<i>Basal Cell Carcinoma</i>	514
<i>Actinic Keratoses</i>	327
<i>Vascular Lesions</i>	142
<i>Dermatofibroma</i>	115
Total	10.015

Terlihat diatas bahwa pesebaran data tidak seimbang, tetapi untuk model *deep learning* dan *convolutional neural network* seperti ini penting untuk mengikutsertakan seluruh data, supaya model dapat belajar dengan baik tanpa kehilangan informasi apapun. Adapun untuk meningkatkan akurasi nantinya, maka dataset yang tidak seimbang tersebut harus ditambah dari dataset lain supaya seimbang.

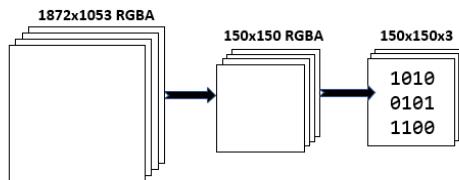
Kemudian data dieksplorasi dengan mengambil 25 sampel gambar, berikut adalah sampel gambar yang telah diambil secara acak:



Gambar 3.2 Sample 25 Gambar yang dipilih acak

Selain itu, file *Metadata CSV* tersebut berisi data-data pasien seperti jenis kelamin, umur, dan tempat dimana kanker tumbuh. Tetapi hal tersebut tidak begitu penting dikarenakan disini kita melakukan klasifikasi gambar bukan klasifikasi regresi.

3.4.2 Konversi Dataset

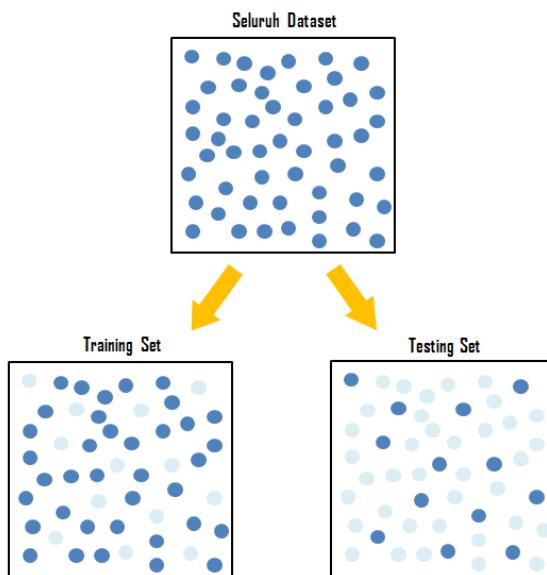


Gambar 3.3 Ilustrasi konversi dataset

Kemudian data dikonversi format dan resolusinya dari resolusi 1872x1053 *pixel* dengan format *RGBA* menjadi 150x150 *pixel* dengan format *RGB* supaya data kompatible dengan format *TensorFlow* dan data dapat diolah dengan cepat sehingga tidak akan membutuhkan *resource* yang besar seperti memori, kartu grafis dan *processor*, karena data akan diolah di *cloud* sehingga harus dibuat seefisien mungkin supaya tidak memakan biaya dan waktu *training* yang lama.

Kemudian data dikonversi ke dalam bentuk array. Penting untuk mengubah data menjadi bentuk array, dikarenakan bahasa pemrograman tidak mengenal tipe data gambar, oleh karena itu *metadata* dari gambar seperti warna gambar dan luas gambar harus diubah kedalam tipe data *float* dan diubah kedalam bentuk array[6], [45].

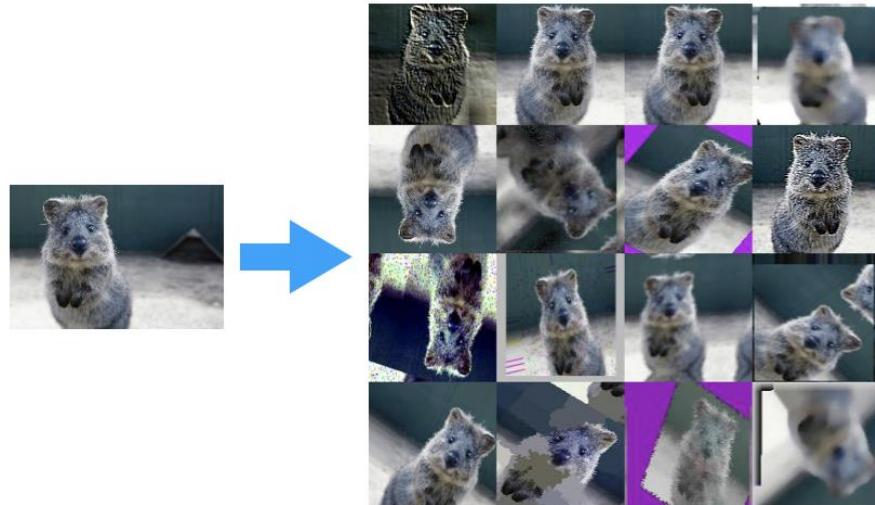
3.4.3 Pembagian Data Latih, Validasi dan Tes



Gambar 3.4 Illustrasi pembagian dataset[79]

Kemudian dataset dibagi kedalam data latih, data validasi dan data test. Dataset dibagi dengan 20% untuk data validasi, 20% untuk data test dan 80% untuk data latih. Tujuan dari pembagian data latih adalah untuk data acuan latih, artinya data tersebut akan dipakai untuk melatih model belajar memahami data. Tujuan dari pembagian data validasi untuk melihat performa model dalam mempelajari data dan melihat apakah data tersebut *overfitting* atau *underfitting*. Dan tujuan dari pembagian data test untuk tes model yang telah dilatih apakah bisa mempelajari data diluar data latih dan data validasi secara sempurna[6], [45].

3.4.4 Augmentasi dataset



Gambar 3.5 Illustrasi augmentasi dataset[80]

Tahap terakhir adalah augmentasi data, agar proses training nanti mendapatkan hasil yang optimal, data harus dibuat sedemikian rupa bentuk, ukuran, perbesaran, margin, warna dan rotasi dari gambar, supaya ketika proses pendekripsi data setelah data dilatih, data dapat belajar dari data yang bentuknya berbeda (tidak sesuai) dengan data yang ada pada proses latih, hal tersebut dapat dilakukan dengan menggunakan teknik augmentasi data[6], [45]. Berikut adalah parameter apa saja yang dipakai untuk augmentasi data[6], [45]:

```
ImageDataGenerator(rotation_range=30,  
                    zoom_range=0.15,  
                    width_shift_range=0.2,  
                    height_shift_range=0.2,  
                    shear_range=0.15,  
                    horizontal_flip=True,  
                    fill_mode="nearest")
```

1. *Rotation Range*, berfungsi untuk mengubah rotasi gambar secara acak dengan parameter sudut 30° .
2. *Zoom Range*, berfungsi untuk memperbesar gambar secara acak dengan parameter perbesaran sebesar 15%.
3. *Width Shift Range*, berfungsi untuk mengubah lebar dari gambar dengan parameter perubahan sebesar 20%.

4. *Height Shift Range*, berfungsi untuk mengubah tinggi dari gambar dengan parameter perubahan sebesar 20%.
5. *Shear Range*, berfungsi untuk mengeser gambar dalam arah berlawanan jarum jam dengan parameter pergeseran sekitar 15%.
6. *Horizontal Flip*, berfungsi untuk membalik gambar secara horizontal, parameter berbentuk *Boolean* yang berarti jika dia *True* maka fungsi *Horizontal Flip* akan diterapkan
7. *Fill Mode*, berfungsi untuk men-skalakan gambar yang resolusi nya diatas atau dibawah dari input yaitu 150x150 dan skala yang bukan 1:1 kedalam resolusi dan skala tersebut. parameter berupa string dengan nilai *nearest* yang berarti akan diskalakan ke bentuk-bentuk terdekat dari 150x150 yang tetap sama dengan skala sebelum diubah

3.5 Membangun Model

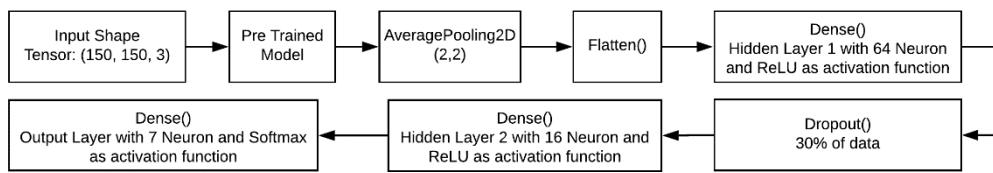
Selanjutnya adalah perancangan *Transfer Learning* model. Model dirancang sedemikian rupa sehingga model dapat menerapkan prinsip dari *Convolutional Neural Network (CNN)* untuk klasifikasi citra, sehingga model dapat mempelajari data gambar. Dengan menggunakan *library tensorflow* dan teknik *transfer learning* model telah dirancang.

```
params = (weights='imagenet',
           include_top=False,
           input_tensor=Input(150,150,3))

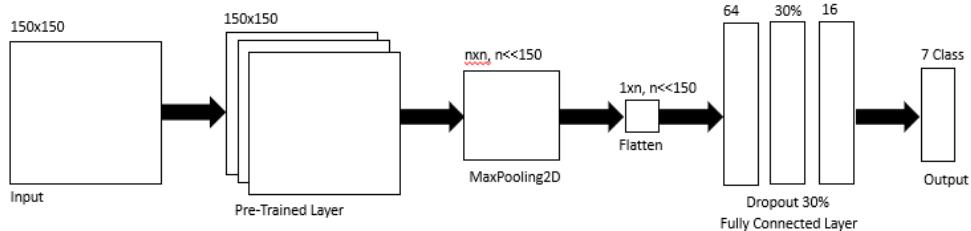
pre_trained_model = {'DenseNet201': DenseNet201(*params),
                     'InceptionV3': InceptionV3(*params),
                     'ResNet152V2': ResNet152V2(*params),
                     'Xception': Xception(*params)}

for key, value in pre_trained_model.items():
    model = Sequential([pre_trained_model[key],
                        AveragePooling2D(pool_size=(2,2)),
                        Flatten(),
                        Dense(filter=64, activation='relu'),
                        Dropout(0.3),
                        Dense(filter=16, activation='relu'),
                        Dense(filter=7, activation='softmax')])
```

Digunakan metode *transfer learning*. Dengan bantuan *pre-trained model* atau model yang sudah di latih sebelumnya dengan dataset yang berbeda tetapi mempunyai performa yang bagus sehingga akan optimal jika digunakan untuk dataset yang lain seperti pada dataset yang dipakai saat ini. Tetapi untuk kasus *transfer learning* ini, model dioptimalkan dengan menggunakan *layer* tambahan[6], [45] Untuk model, sudah dirancang seperti diagram dibawah ini:



Gambar 3.6 Diagram Alur Model Transfer Learning



Gambar 3.7 Visualisasi Model Transfer Learning

3.6 Alur Model

3.6.1 Inisialisasi Input

Pertama, diinisialisasi *input* terlebih dahulu, disini digunakan *input* dari data gambar yang sudah diubah menjadi *tensor*. *Tensor* untuk gambar tersebut akan berbentuk rank 3 array dengan properti ($n, x, y, depth$). Dimana nilai n merupakan jumlah data (data latih: 8012, data validasi dan data test: 2003), kemudian nilai x merupakan lebar gambar (150 *pixel*) dan y merupakan tinggi gambar (150 *pixel*) dan $depth$ adalah kedalaman warna (disini digunakan nilai 3 karena menggunakan RGB)[6], [45]

3.6.2 Inisialisasi Pre-Trained Model

Kemudian, digunakan *pre-trained* model. Ada 4 pre-trained model yang digunakan disini yaitu:

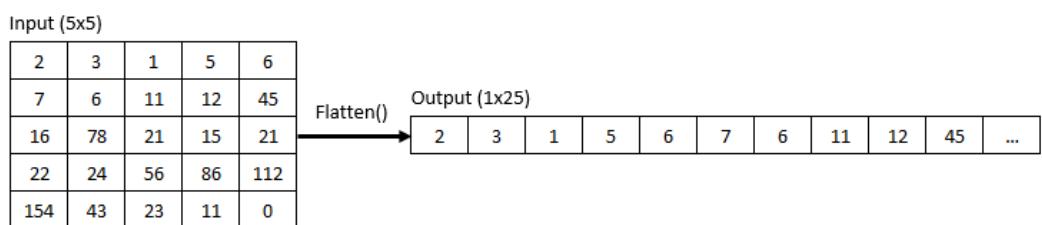
1. *DenseNet201*[81].
2. *InceptionV3*[82].
3. *ResNet50*[83].
4. *Xception*[84].

3.6.3 AveragePooling

Digunakan metode *AveragePooling* 2D karena bentuk matriks dari data *Image* adalah 2 dimensi 2x2.

3.6.4 Flatten Layer

Setelah itu, digunakan *Flatten layer*, dikarenakan input berupa gambar yang berbentuk tensor kemudian dari *tensor* tersebut harus diubah menjadi sebuah angka, sehingga harus digunakan metode *flatten* untuk mengubah bentuk *tensor nxn* tersebut menjadi *array* satu dimensi karena hasil dari *output* akan berupa probabilitas. Akan mustahil untuk mengetahui probabilitas dari hasil gambar tersebut dikarenakan *output* masih berupa *tensor nxn*[6][45]. Untuk ilustrasi *flatten layer* bisa dilihat pada gambar dibawah ini:



Gambar 3.8 Ilustrasi *Flatten Layer*

3.6.5 Dense Layer

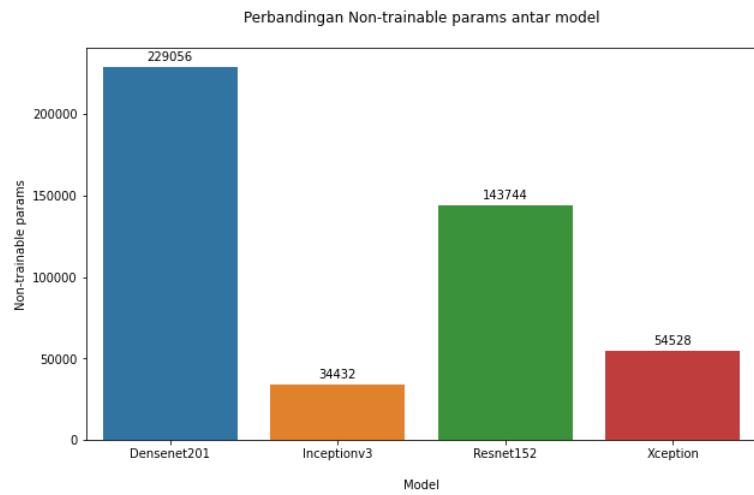
Setelah itu, digunakan *Dense* tambahan, yaitu *Hidden Layer* tambahan untuk mengoptimalkan *hidden layer* pada *pre-trained* model sebelumnya dengan:

1. *Hidden Layer 1* dengan *neuron* aktif berjumlah 64 dengan fungsi aktivasi *ReLU* dan *Dropout Layer* dengan *Layer* yang di *Dropout* sebesar 30%
2. *Hidden Layer 2* dengan *neuron* aktif berjumlah 16 dengan fungsi aktivasi *ReLU*.
3. *Output Layer* dengan *neuron* aktif (yang kemudian akan menghasilkan output berupa probabilitas) berjumlah 7 (sesuai dengan label) dengan fungsi aktivasi *Softmax*

Fungsi dari fungsi aktivasi disini adalah untuk menghasilkan *output* yang akan dilanjutkan ke *hidden layer* setelahnya atau akan menjadi sebuah *output* dengan hasil probabilitas[6], [45]

3.6.6 Dropout Layer

Parameter yang digunakan untuk dropout layer adalah sebesar 30% random *neuron* yang akan dibuang. Sehingga akan ada *neuron* yang tidak dilatih (*non-trainable neuron*). Setiap *pre-trained* model akan berbeda jumlah *neuron*-nya pada setiap masing-masing *pre-trained* model.

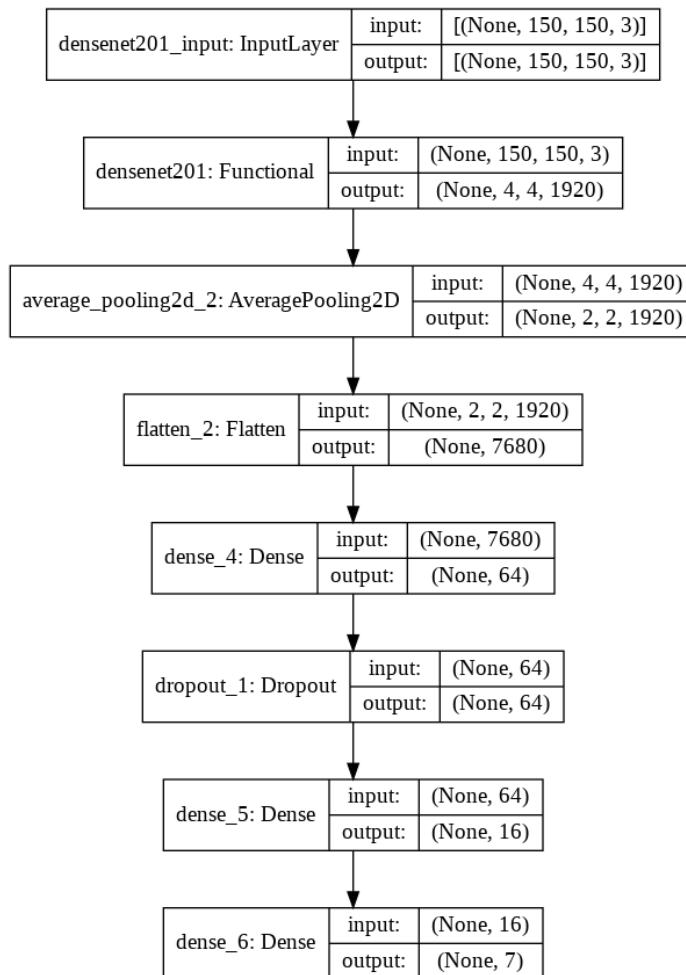


Gambar 3.9 Jumlah Parameter *Train* yang dihilangkan

Diatas merupakan penjelasan umum dari model secara keseluruhan, bagian ini akan menjelaskan sedikit tentang parameter dari masing-masing *pre-trained* model. Ada 4 *pre-trained* model yang digunakan disini,

3.6.7 Pre-Trained Model

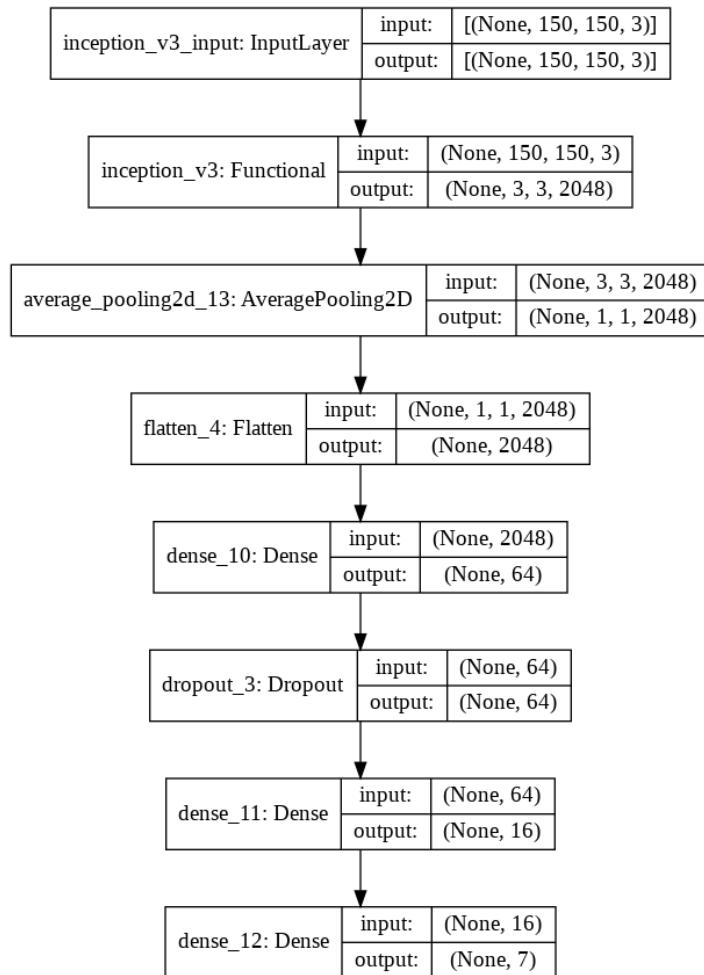
a. DenseNet201



Gambar 3.10 Diagram Model DenseNet201

Diatas merupakan diagram model yang telah dibangun menggunakan arsitektur *pre-trained model* yaitu *DenseNet201*. Dengan jumlah neuron yang di *train* sebesar 18.585.671 dan jumlah neuron yang tidak di *train* sebesar 229.056 (merupakan *neuron* yang di *dropout*).

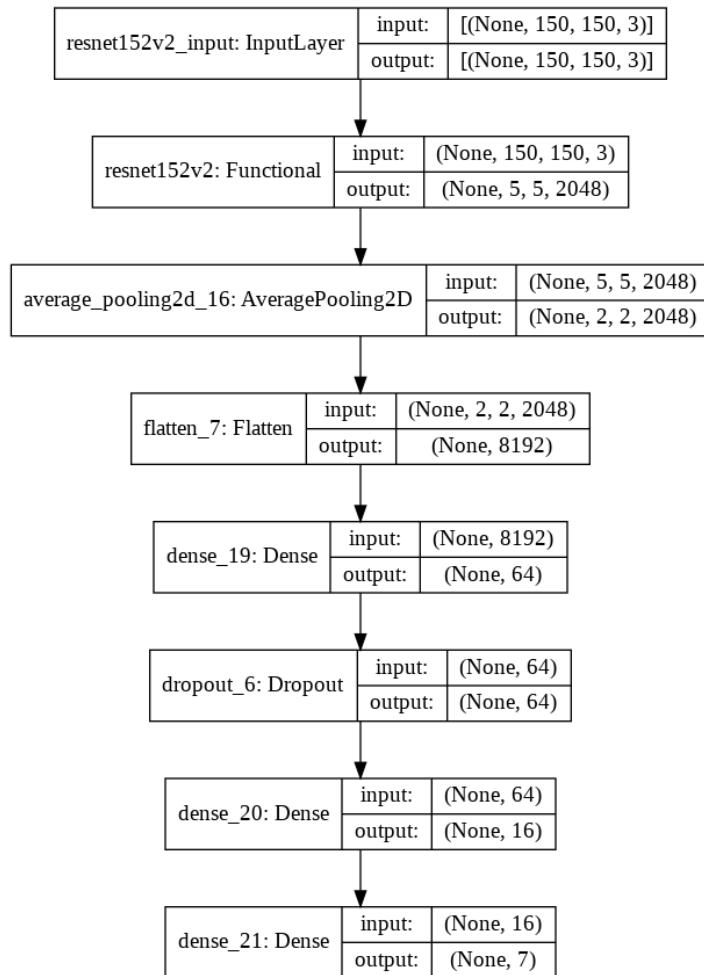
b. *InceptionV3*



Gambar 3.11 Diagram Model *InceptionV3*

Diatas merupakan diagram model yang telah dibangun menggunakan arsitektur *pre-trained model* yaitu *InceptionV3*. Dengan jumlah *neuron* yang di *train* sebesar 21.900.647 dan jumlah *neuron* yang tidak di *train* sebesar 34.432 (merupakan *neuron* yang di *dropout*).

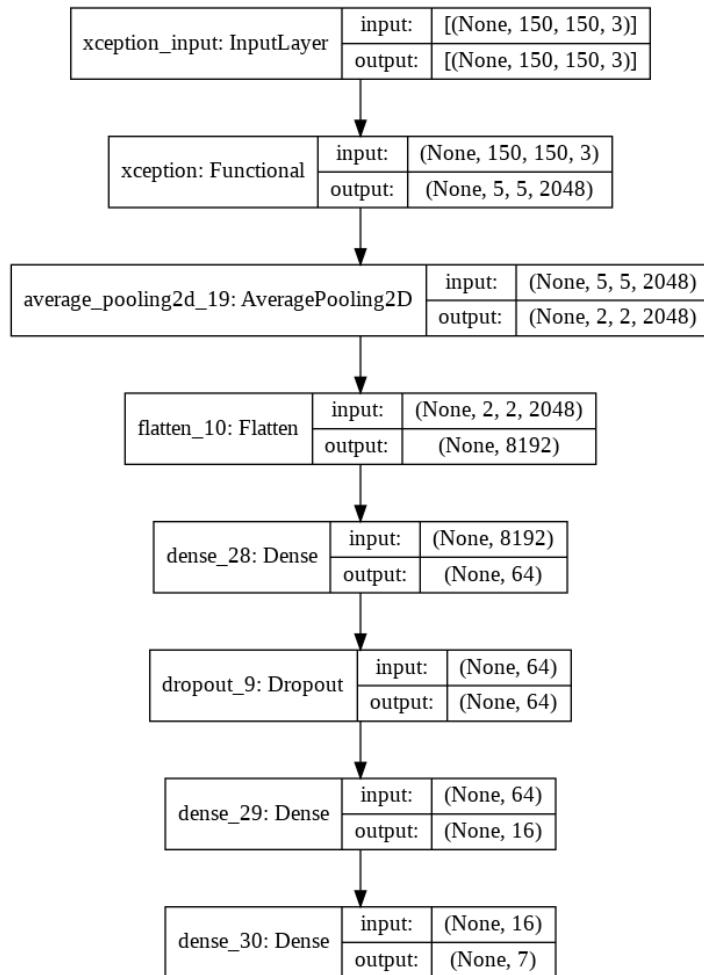
c. *ResNet152V2*



Gambar 3.12 Diagram Model *ResNet152V2*

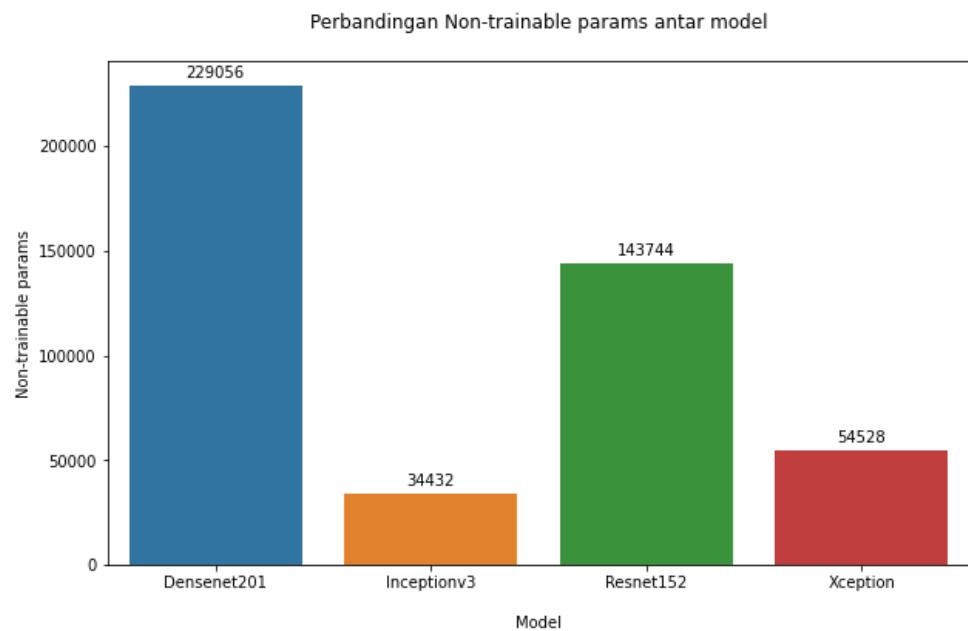
Diatas merupakan diagram model yang telah dibangun menggunakan arsitektur *pre-trained model* yaitu *ResNet152V2*. Dengan jumlah *neuron* yang di *train* sebesar 58.713.415 dan jumlah *neuron* yang tidak di *train* sebesar 143.744 (merupakan *neuron* yang di *dropout*).

d. *Xception*

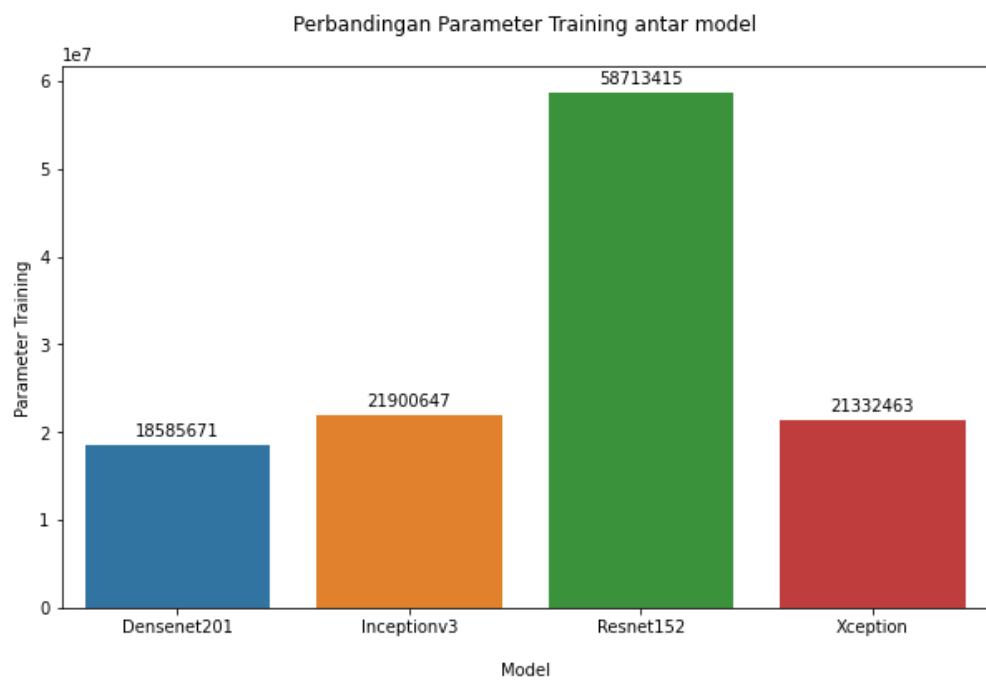


Gambar 3.13 Diagram Model *Xception*

Diatas merupakan diagram model yang telah dibangun menggunakan arsitektur *pre-trained model* yaitu *Xception*. Dengan jumlah *neuron* yang di *train* sebesar 21.332.463 dan jumlah *neuron* yang tidak di *train* sebesar 54.528 (merupakan *neuron* yang di *dropout*).



Gambar 3.14 Perbandingan *Trainable* Parameter antar *Pre-Trained* Model



Gambar 3.15 Perbandingan Non-Trainable Parameter antar Pre-Trained Model

3.7 Pelatihan Model

Setelah *transfer learning* model dirancang, kemudian model dilatih. Ketika model dilatih, ada 2 parameter yang harus diterapkan, yaitu parameter *compile* dan *fit*.

3.7.1 *Compile*

Sebelum model dilatih, model harus diterapkan parameter dari *loss function*, *accuracy metric* dan *optimizer*. Untuk menerapkan parameter ini, digunakan *methods* dari *class keras.model* yaitu *methods compile*. Berikut adalah parameter yang diterapkan untuk melatih model di *methods compile*:

```
model.compile(optimizer=Adam(learning_rate=1e-4),  
              loss=SparseCategoricalCrossentropy(),  
              metrics=['accuracy'])
```

1. Optimizer

Optimizer yang digunakan disini adalah *optimizer Adam* dengan *learning rate* 0.0001. *Optimizer Adam* digunakan karena optimal untuk model klasifikasi

2. Loss

Loss yang digunakan disini adalah *loss sparse categorical crossentropy*. *Loss* ini sangat optimal untuk data yang berbentuk *multiclass* (lebih dari satu *class*).

3. Metrics

Metrics yang digunakan adalah *accuracy* dikarenakan data berbentuk tensor dari gambar sehingga tidak cocok untuk menggunakan selain *accuracy*. Selain *accuracy* ada *metrics MSE (Mean Squared Error)* dan *MAE (Mean Average Error)*

3.7.2 Fit

Kemudian, masuk ke dalam proses pelatihan dengan menggunakan *methods fit* dari *class model*. Proses *fit* akan mem-*fitting* dataset ke dalam model, sehingga model akan belajar dari dataset. Proses ini membutuhkan sumber daya yang besar dan membutuhkan waktu yang lama, untuk rata-rata semua *transfer learning model*, pelatihan dilakukan dalam waktu 1 jam untuk setiap *transfer learning model*, sehingga diperlakukan waktu 4 jam untuk seluruh model, proses ini dapat disingkat dengan menyederhanakan model dan menghapus beberapa dataset, tetapi mustahil dilakukan karena akan merusak akurasi model itu sendiri. Berikut adalah parameter yang di-set ke dalam *methods fit*,

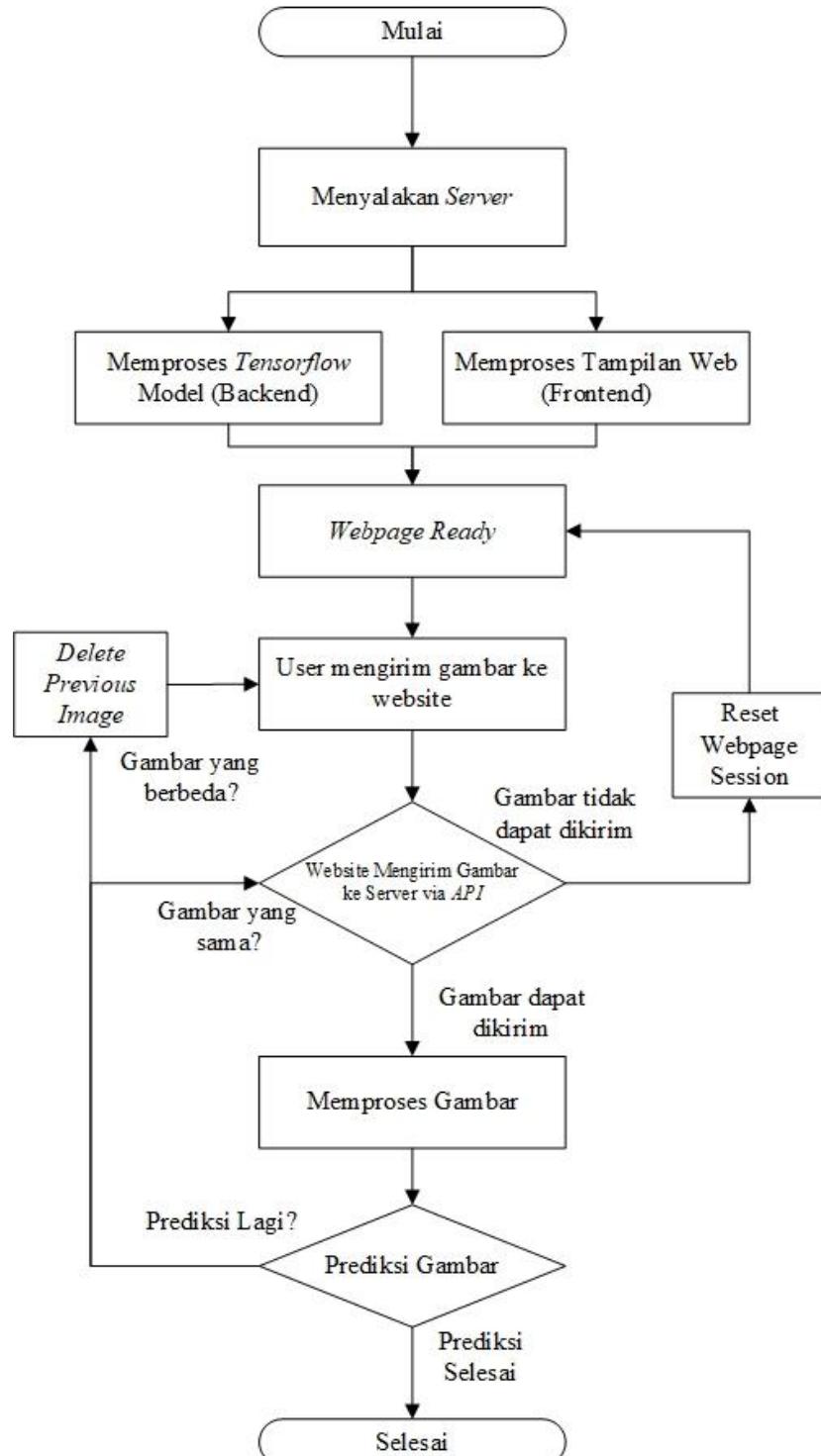
```
model.fit(train_dataset,  
          epochs=10,  
          validation_data=valid_dataset,  
          verbose=1)
```

Parameter yang di-set ke dalam *methods fit* adalah dataset latih yaitu *train_dataset* dan dataset validasi yaitu *valid_dataset*. Seperti yang sudah dijelaskan pada paragraf diatas, dataset validasi berfungsi untuk melihat performa apakah *overfitting* atau *underfitting*, kemudian mencegah model mempelajari hal yang sama berulang-ulang.

Kemudian ada *epochs* yang merupakan parameter untuk men-set berapa kali model ingin di iterasi, semakin banyak iterasi maka akan semakin bagus akurasi dan semakin baik model belajar, tetapi akan memakan banyak waktu, biaya dan sumber daya dikarenakan pelatihan ini dilakukan secara *online*, dan karena pelatihan berbasis *convolutional neural network* yang artinya akan menggunakan sumber daya *GPU* yang besar, batas dari penggunaan *GPU* di *Google Colab* hanya sampai 6 jam. Sehingga *epochs* hanya di-set sebesar 10 kali perulangan.

Kemudian *verbose*, *verbose* berfungsi untuk melihat progress dari pelatihan, jika di-set 1 maka program akan menampilkan progress pada *terminal*, tetapi jika di-set 0 maka program tidak akan menampilkan progress pada *terminal* (metode *silent*).

3.8 Deployment



Gambar 3.16 Diagram Alur Proses Deployment

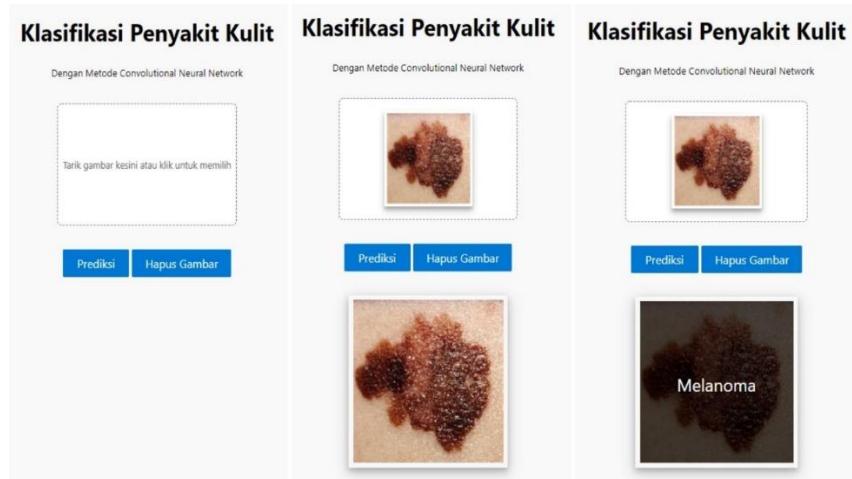
Selanjutnya adalah proses *deployment*, proses ini mengintegrasikan model yang telah dibangun kedalam sebuah aplikasi (dalam kasus ini, aplikasi sederhana) supaya dapat digunakan oleh masyarakat secara mudah tanpa harus menulis kode, dalam hal ini dibuat aplikasi *web*. Digunakan bahasa pemrograman *python* dengan *framework flask* untuk membuat sebuah *back-end* dan bahasa pemrograman *javascript native* dengan *html* sebagai *front-end* nya sehingga tercipta sebuah aplikasi *web* sederhana berbasis *python* dengan *framework flask*.

Untuk mengkoneksikan model dengan aplikasi *web* tersebut dibutuhkan sebuah *API (Application Program Interface)* *tensorflow load_model* dan *HDF5*. *HDF5* berfungsi untuk mengarsipkan bobot dan metadata model ke dalam format **.h5* (format yang biasa dipakai dalam *machine learning*, berisi *binary* dari model). Kemudian *tensorflow load_model* berfungsi untuk me-*load* model ke dalam bentuk *tensorflow* seperti sebelum di arsipkan (di dalam *tensorflow load_model* terdapat atribut yang sama seperti *framework tensorflow* seperti *model.predict*).

Setelah file **.h5* di-*load*, langkah selanjutnya *server* menerima file gambar dari *front-end* melalui metode *request.post*, kemudian gambar yang ingin di deteksi harus berformat *array*, berukuran 150x150 (seperti pada model) dan di normalisasi dengan membagi *array* dengan 255.0 sehingga gambar harus di konversi (proses konversi dilakukan otomatis oleh *python*). Setelah di *load*, maka model bisa di deteksi yaitu dengan atribut *model.predict*. Proses itu berada di dalam *API back-end* yang dibuat seperti *server* (dalam hal ini *server localhost*, artinya *server* itu hanya ada di komputer yang menjalankan *script server* tersebut) yang berbasis *framework flask*, sehingga proses mendeteksi tidak akan terlihat oleh *user*.

Kemudian, supaya hasil dapat dilihat oleh *user*, maka dilakukan pemanggilan *API* yang berasal dari *web server flask* tersebut dengan menggunakan metode *request.get*, maka *front-end* akan menerima file *JSON* dari *API* yang berisi hasil prediksi, kemudian memunculkannya pada program.

Berikut adalah demonstrasi dari aplikasi klasifikasi penyakit kulit berbasis aplikasi *web*,



Gambar 3.17 Demonstrasi Aplikasi Berbasis Web Klasifikasi Penyakit Kulit

BAB IV

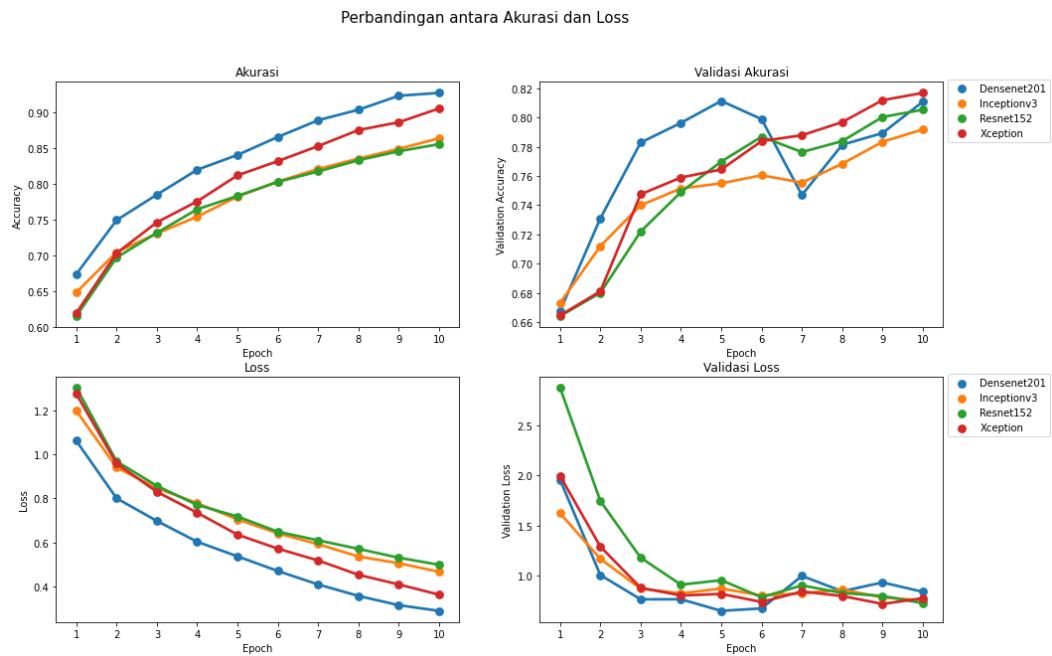
HASIL DAN PEMBAHASAN

4.1 Hasil dan Evaluasi Model

4.1.1 Evaluasi Hasil Pelatihan

Setelah model dilatih, model akan mengeluarkan *output* berupa *metrics accuracy* dan *loss*, seperti apa yang telah diterapkan pada sebelumnya. Output ini dikeluarkan oleh model sebagai hasil dari fungsi aktivasi dan learning rate yang sudah di terapkan sebelumnya[6], [45].

Selain mengeluarkan *output metrics accuracy* dan *loss*, model juga akan mengeluarkan *metrics val_accuracy* dan *val_loss* yang merupakan hasil dari data validasi. Berikut adalah hasil *metrics* dari pelatihan,



Gambar 4.1 Metrics Accuracy, Loss, Val_Accuracy, Val_Loss

1. Accuracy

Tabel 4.1 Perbandingan antara *pre-trained model* terhadap *Metrics Accuracy*

<i>Epoch</i>	<i>Densenet201</i>	<i>Inceptionv3</i>	<i>Resnet152</i>	<i>Xception</i>
1	0.6736	0.6482	0.6151	0.6194
2	0.749	0.7043	0.6968	0.7029
3	0.7846	0.7304	0.7314	0.746
4	0.8193	0.7539	0.7642	0.7752
5	0.84	0.7818	0.7828	0.8115
6	0.865	0.8028	0.8024	0.8313
7	0.8884	0.8203	0.8169	0.8523
8	0.903	0.8349	0.8326	0.8747
9	0.9225	0.8484	0.8451	0.8855
10	0.9265	0.8632	0.855	0.9048
Rata-Rata	0.84	0.78	0.77	0.80

Berdasarkan tabel diatas, dapat diambil kesimpulan bahwa dari ke-4 *pre-trained* model tersebut, model dapat mempelajari dengan baik, ditandai dengan *metrics accuracy* yang nilainya stabil konvergen menuju 1 dengan *accuracy* paling tinggi ada pada model *Densenet201*.

2. Loss

Tabel 4.2 Perbandingan antara *pre-trained* model terhadap *Metrics Loss*

<i>Epoch</i>	<i>Densenet201</i>	<i>Inceptionv3</i>	<i>Resnet152</i>	<i>Xception</i>
1	1.0616	1.1982	1.3034	1.2745
2	0.8003	0.9404	0.9683	0.9586
3	0.6973	0.8466	0.8566	0.8297
4	0.6027	0.7781	0.7702	0.7348
5	0.5364	0.7043	0.7174	0.635
6	0.4697	0.641	0.6485	0.5714
7	0.4087	0.5916	0.6092	0.5176
8	0.3565	0.5356	0.5703	0.4526
9	0.3145	0.5051	0.5302	0.4087
10	0.2886	0.4664	0.4981	0.3623
Rata-Rata	0.55	0.72	0.75	0.67

Berdasarkan tabel diatas, dapat diambil kesimpulan bahwa dari ke-4 *pre-trained* model tersebut, model dapat mempelajari dengan baik terhadap iterasi, ditandai dengan *metrics loss* yang nilainya stabil konvergen menuju 0 dengan *loss* paling kecil ada pada model *Xception*.

3. *Val Accuracy*

Tabel 4.3 Perbandingan antara *pre-trained model* terhadap *Metrics Val_Accuracy*

<i>Epoch</i>	<i>Densenet201</i>	<i>Inceptionv3</i>	<i>Resnet152</i>	<i>Xception</i>
1	0.6675	0.673	0.664	0.6645
2	0.7309	0.7119	0.68	0.681
3	0.7828	0.7399	0.7219	0.7474
4	0.7963	0.7514	0.7489	0.7589
5	0.8113	0.7549	0.7698	0.7644
6	0.7988	0.7604	0.7868	0.7838
7	0.7469	0.7554	0.7763	0.7878
8	0.7813	0.7683	0.7838	0.7968
9	0.7893	0.7833	0.8003	0.8118
10	0.8108	0.7918	0.8053	0.8168
Rata-Rata	0.77	0.75	0.75	0.76

Berdasarkan tabel diatas, dapat diambil kesimpulan bahwa dari ke-4 *pre-trained* model tersebut, model dapat mempelajari data validasi (data diluar data latih) dengan baik, ditandai dengan *metrics val_accuracy* yang nilainya stabil konvergen menuju 1 pada model *Xception*, sedangkan untuk model yang lain terdapat *noise* yang menyebabkan model *val_accuracy* nya naik dan turun, tetapi masih kategori model yang bagus. *Val_accuracy* paling baik ada pada model *Xception* dikarenakan memiliki *val_accuracy* paling tinggi dan model paling stabil.

4. Val Loss

Tabel 4.4 Perbandingan antara *pre-trained model* terhadap *Metrics Val_Loss*

Epoch	Densenet201	Inceptionv3	Resnet152	Xception
1	1.9549	1.6207	2.874	1.9935
2	1.0041	1.166	1.7431	1.2885
3	0.7622	0.8685	1.1773	0.8774
4	0.7628	0.8212	0.9082	0.8004
5	0.647	0.8702	0.9529	0.8161
6	0.6715	0.8	0.7833	0.7357
7	0.9951	0.8203	0.9006	0.8405
8	0.8429	0.86	0.8253	0.7952
9	0.9306	0.7808	0.796	0.7158
10	0.8371	0.7523	0.7235	0.7735
Rata-Rata	0.94	0.94	1.17	0.96

Berdasarkan tabel diatas, dapat diambil kesimpulan bahwa dari ke-4 *pre-trained* model tersebut, model dapat mempelajari dan memprediksi data validasi (data diluar data latih) terhadap iterasi dengan cukup walaupun tidak begitu bagus, ditandai dengan *metrics val_loss* yang nilainya tidak terlalu konvergen menuju 0 masih banyak yang *val_loss* nya naik turun dan banyak noise nya, tetapi yang paling stabil disini ada di *ResNet152*.

Kesimpulan dari seluruh *metrics* diatas, model dengan hasil yang paling tinggi dan stabil dalam 10 epochs adalah *Xception*, dengan beberapa parameter seperti *accuracy*, *loss*, dan *val_accuracy* semuanya stabil dan konvergen ke nilai 1 dan 0. *Accuracy* menghasilkan 90%. *Loss* menghasilkan 36%. *Validation accuracy* menghasilkan 81%. *Validation loss* menghasilkan 77%.

Karena *metrics* diatas merupakan hasil dari pelatihan yang berarti hanya menghitung dataset yang ada di data latih dan validasi[6], sehingga dibutuhkan metode lain yang dapat menilai performa dari data test, kemudian karena dataset yang ada tidak *balanced*, sehingga dibutuhkan juga metode lain dalam mengevaluasi yaitu dengan menggunakan *confusion metrics*[85], [86].

4.1.2 Confusion Metrics

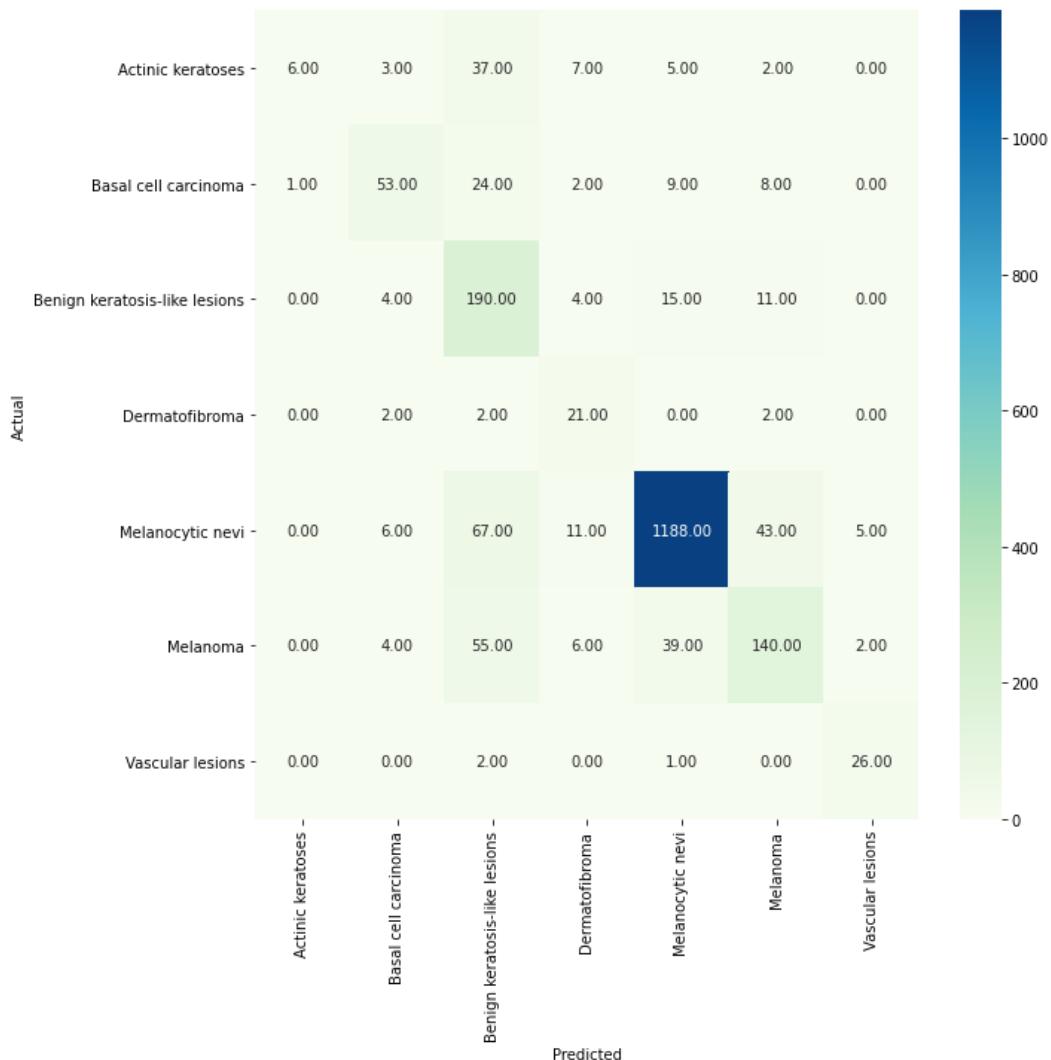
Selain menggunakan metode *metrics* diatas, evaluasi juga dapat dilakukan dengan menggunakan *confusion metrics*. *Dataset* yang digunakan untuk mengukur performa dari model adalah dataset yang berbeda dari data latih dan data validasi, yaitu data *test*. Supaya dengan digunakan data *test*, maka akan mudah untuk mengetahui performa dari model, karena model tidak pernah belajar data tersebut sebelumnya. Dalam data *test*, terdapat 2003 data dengan 7 label dengan jumlah data adalah sebagai berikut,

Tabel 4.5 Jumlah Data dan Label *Test*

Nama	Jumlah Data
<i>Melanocytic Nevi</i>	60
<i>Melanoma</i>	97
<i>Benign Keratosis-Like Lesions</i>	224
<i>Basal Cell Carcinoma</i>	27
<i>Actinic Keratoses</i>	1320
<i>Vascular Lesions</i>	246
<i>Dermatofibroma</i>	29
Total	2003

Dengan menggunakan data *test* yang sudah dipisah dari data latih dan data validasi, model diberikan dataset untuk memprediksi data yang tidak pernah ada di data latih dan data validasi sebelumnya sehingga model kita dapat melihat apakah model dapat mempelajari data dengan baik, jika tidak maka akan dibutuhkan penambahan parameter untuk model dilatih kembali. Untuk setiap *pre-trained model*, hasil yang dikeluarkan berbeda-beda karena setiap *pre-trained model* mempunyai parameter dan jumlah *neuron* yang berbeda, yang tentu saja akan mempengaruhi hasil. Berikut adalah *confusion metrics* dari tiap-tiap *pre-trained model*.

1. DenseNet201



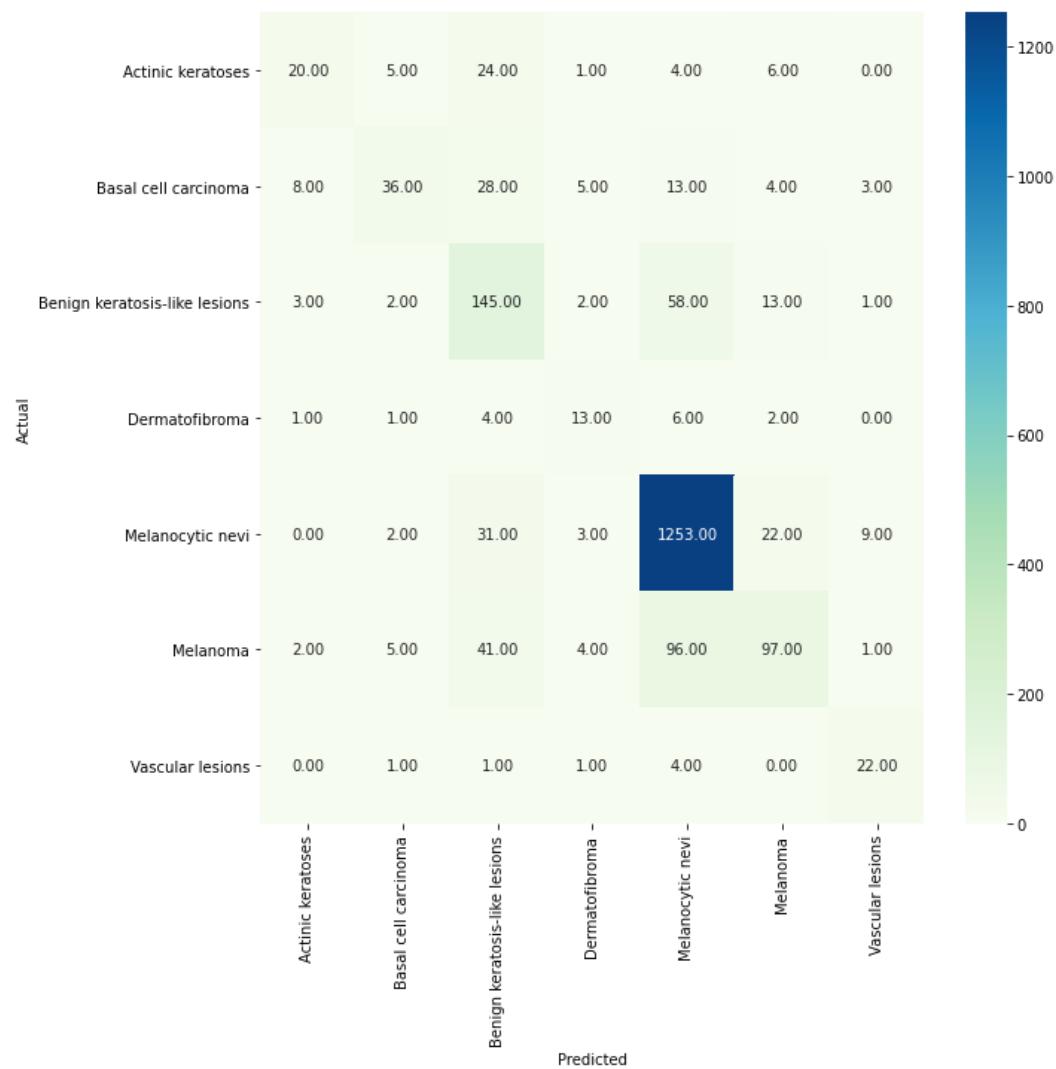
Gambar 4.2 DenseNet201 Confusion Metrics

Tabel 4.6 Precision, Recall dan Accuracy dari DenseNet201

Jenis Kanker Kulit	Precision	Recall	Jumlah Data
Actinic keratoses	0.86	0.10	60
Basal cell carcinoma	0.74	0.55	97
Benign keratosis-like lesions	0.50	0.85	224
Dermatofibroma	0.41	0.78	27
Melanocytic nevi	0.95	0.90	1320
Melanoma	0.68	0.57	246
Vascular lesions	0.79	0.90	29
Rata-Rata	0.7	0.6	-
Akurasi		0.81	

Model *DenseNet201* mendapat nilai akurasi 0.81 atau 81% yang berarti model 81% akurat dalam mendekripsi 7 label. Rata-rata model *precision* berada di atas 70% seperti *Actinic Keratoses*, *Basal Cell Carcinoma*, *Melanocytic Nevi* dan *Vascular Lesions*. Jenis kanker kulit tersebut dapat mencapai precision diatas 70 % kemungkinan karena gambar dari jenis kanker tersebut berbeda dari rata-rata jenis kanker kulit seperti *Actinic Keratoses*, *Basal Cell Carcinoma* dan *Vascular Lesion* sehingga *false positive* pun sedikit, sehingga *precision* pun naik walaupun data yang tersedia sedikit. Kemudian kemungkinan jumlah dari jenis kanker kulit tersebut terlalu jauh banyak seperti *Melanocytic Nevi* sehingga data terlalu banyak mempelajari jenis kanker kulit tersebut sehingga *false positive* pun sedikit. Untuk *Benign Keratosis-like Lesion*, *Dermatofibroma* dan *Melanoma* kemungkinan karena gambar jenis kanker kulit banyak yang sama dari rata-rata jenis kanker kulit dan jumlah nya yang terpaut jauh dari *Melanocytic Nevi* sehingga akan banyak *false positive*-nya. Kemudian untuk recall, sama seperti *precision*, bergantung dari jumlah data yang ada, sehingga model tersebut akan banyak memprediksi *false negative*. Sehingga semakin kecil nilai recall, maka akan semakin banyak juga data yang di prediksi *false negative*. Seperti *Actinic keratoses*, *Basal cell carcinoma* dan *Melanoma*.

2. InceptionV3



Gambar 4.3 InceptionV3 Confusion Metrics

Tabel 4.7 Precision, Recall dan Accuracy dari InceptionV3

Jenis Kanker Kulit	Precision	Recall	Jumlah Data
Actinic keratoses	0.59	0.33	60
Basal cell carcinoma	0.69	0.37	97
Benign keratosis-like lesions	0.53	0.65	224
Dermatofibroma	0.45	0.48	27
Melanocytic nevi	0.87	0.95	1320
Melanoma	0.67	0.39	246
Vascular lesions	0.61	0.76	29
Rata-Rata	0.63	0.56	-
Akurasi		0.79	

Model *InceptionV3* mendapat nilai akurasi 0.79 atau 79% yang berarti model 79% akurat dalam mendekripsi 7 label. Rata-rata model *precision* berada di atas 60% seperti *Basal Cell Carcinoma*, *Melanocytic Nevi*, *Melanoma* dan *Vascular Lesions*. Jenis kanker kulit tersebut dapat mencapai precision diatas 60 % kemungkinan karena gambar dari jenis kanker tersebut berbeda dari rata-rata jenis kanker kulit seperti *Melanoma*, *Basal Cell Carcinoma* dan *Vascular Lesion* sehingga *false positive* pun sedikit, sehingga *precision* pun naik walaupun data yang tersedia sedikit. Kemudian kemungkinan jumlah dari jenis kanker kulit tersebut terlalu jauh banyak seperti *Melanocytic Nevi* sehingga data terlalu banyak mempelajari jenis kanker kulit tersebut sehingga *false positive* pun sedikit. Untuk *Actinic Keratoses*, *Benign Keratosis-like Lesion*, dan *Dermatofibroma* kemungkinan karena gambar jenis kanker kulit banyak yang sama dari rata-rata jenis kanker kulit dan jumlah nya yang terpaut jauh dari *Melanocytic Nevi* sehingga akan banyak *false positive*-nya. Kemudian untuk *recall*, sama seperti *precision*, bergantung dari jumlah data yang ada, sehingga model tersebut akan banyak memprediksi *false negative*. Sehingga semakin kecil nilai *recall*, maka akan semakin banyak juga data yang di prediksi *false negative*. Seperti *Actinic keratoses*, *Basal cell carcinoma*, *Dermatofibroma* dan *Melanoma*.

3. ResNet152



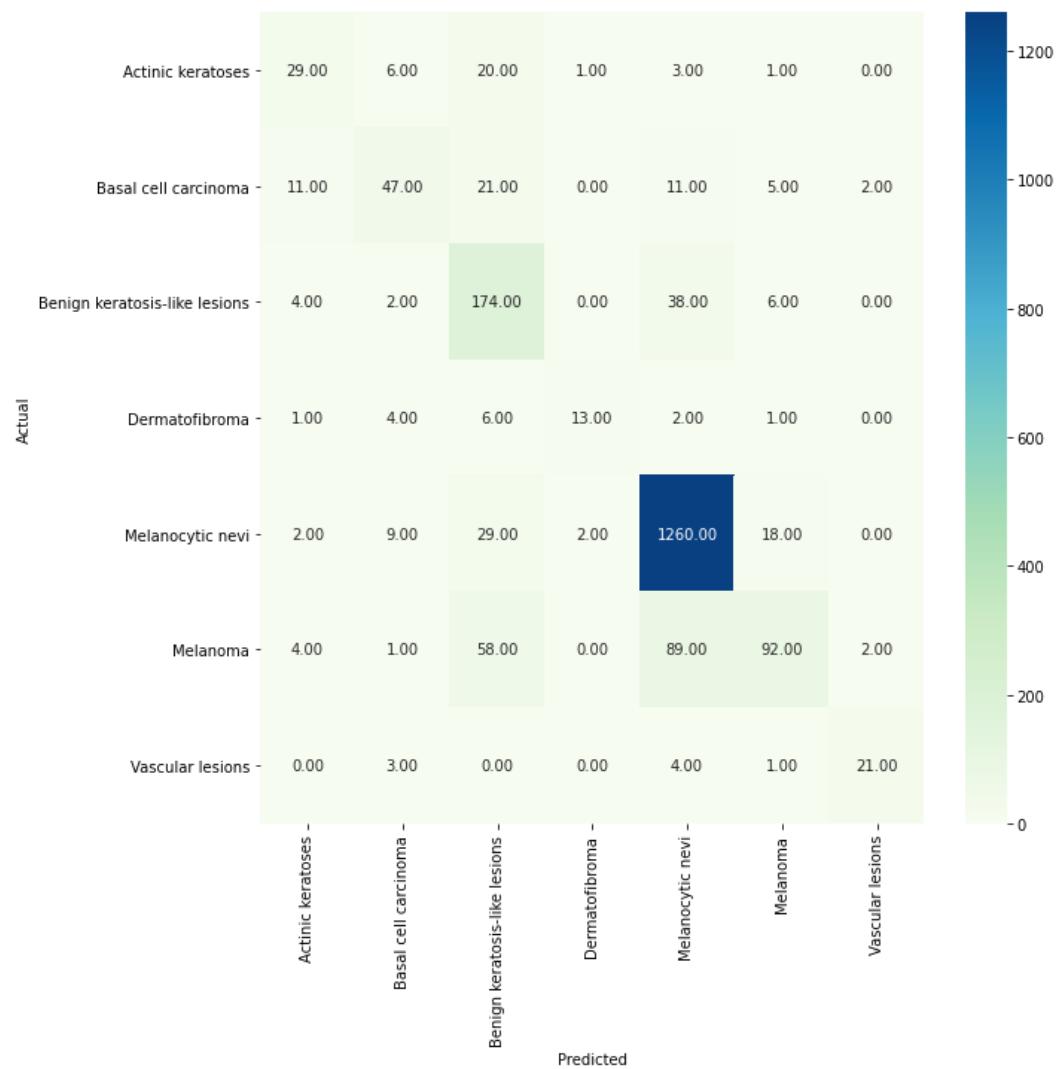
Gambar 4.4 ResNet152 Confusion Metrics

Tabel 4.8 Precision, Recall dan Accuracy dari ResNet152

Jenis Kanker Kulit	Precision	Recall	Jumlah Data
Actinic keratoses	0.57	0.22	60
Basal cell carcinoma	0.53	0.69	97
Benign keratosis-like lesions	0.63	0.61	224
Dermatofibroma	0.59	0.59	27
Melanocytic nevi	0.88	0.95	1320
Melanoma	0.64	0.44	246
Vascular lesions	1.00	0.79	29
Rata-Rata	0.69	0.61	-
Akurasi		0.81	

Model *ResNet152* mendapat nilai akurasi 0.81 atau 81% yang berarti model 81% akurat dalam mendekripsi 7 label. Rata-rata model *precision* berada di atas 60% seperti *Benign keratosis-like lesions*, *Melanocytic nevi*, *Melanoma* dan *Vascular Lesions*. Jenis kanker kulit tersebut dapat mencapai precision diatas 60 % kemungkinan karena gambar dari jenis kanker tersebut berbeda dari rata-rata jenis kanker kulit seperti *Benign keratosis-like lesions*, *Melanonam* dan *Vascular Lesion* sehingga *false positive* pun sedikit, sehingga *precision* pun naik walaupun data yang tersedia sedikit. Kemudian kemungkinan jumlah dari jenis kanker kulit tersebut terlampaui jauh banyak seperti *Melanocytic Nevi* sehingga data terlalu banyak mempelajari jenis kanker kulit tersebut sehingga *false positive* pun sedikit. Untuk *Actinic keratoses*, *Basal cell carcinoma* dan *Dermatofibroma* kemungkinan karena gambar jenis kanker kulit banyak yang sama dari rata-rata jenis kanker kulit dan jumlah nya yang terpaut jauh dari *Melanocytic Nevi* sehingga akan banyak *false positive*-nya. Kemudian untuk *recall*, sama seperti *precision*, bergantung dari jumlah data yang ada, sehingga model tersebut akan banyak memprediksi *false negative*. Sehingga semakin kecil nilai *recall*, maka akan semakin banyak juga data yang di prediksi *false negative*. Seperti *Actinic keratoses*, *Dermatofibroma* dan *Melanoma*.

4. Xception



Gambar 4.5 Xception Confusion Metrics

Tabel 4.9 Precision, Recall dan Accuracy dari Xception

Jenis Kanker Kulit	Precision	Recall	Jumlah Data
Actinic keratoses	0.57	0.48	60
Basal cell carcinoma	0.65	0.48	97
Benign keratosis-like lesions	0.56	0.78	224
Dermatofibroma	0.81	0.48	27
Melanocytic nevi	0.90	0.95	1320
Melanoma	0.74	0.37	246
Vascular lesions	0.84	0.72	29
Rata-Rata	0.72	0.60	-
Accuracy		0.82	

Model *Xception* mendapat nilai akurasi 0.82 atau 82% yang berarti model 81% akurat dalam mendekripsi 7 label. Rata-rata model *precision* berada di atas 70% seperti *Dermatofibroma*, *Melanocytic nevi*, *Melanoma* dan *Vascular lesions*. Jenis kanker kulit tersebut dapat mencapai *precision* diatas 70 % kemungkinan karena gambar dari jenis kanker tersebut berbeda dari rata-rata jenis kanker kulit seperti *Dermatofibroma*, *Melanoma* dan *Vascular Lesion* sehingga *false positive* pun sedikit, sehingga *precision* pun naik walaupun data yang tersedia sedikit. Kemudian kemungkinan jumlah dari jenis kanker kulit tersebut terlampaui jauh banyak seperti *Melanocytic Nevi* sehingga data terlalu banyak mempelajari jenis kanker kulit tersebut sehingga *false positive* pun sedikit. Untuk *Actinic Keratoses*, *Basal Cell Carcinoma*, *Benign Keratosis-like Lesion* kemungkinan karena gambar jenis kanker kulit banyak yang sama dari rata-rata jenis kanker kulit dan jumlahnya yang terpaut sedikit dari *Melanocytic Nevi* sehingga akan banyak *false positive*-nya. Kemudian untuk *recall*, sama seperti *precision*, bergantung dari jumlah data yang ada, sehingga model tersebut akan banyak memprediksi *false negative*. Sehingga semakin kecil nilai *recall*, maka akan semakin banyak juga data yang di prediksi *false negative*. Seperti *Actinic keratoses*, *Basal cell carcinoma*, *Dermatofibroma* dan *Melanoma*

Untuk kesimpulan, diatas merupakan *confusion metrics*, dimana *metrics* tersebut akan sangat berguna jika data berbentuk *imbalanced* (tidak seimbang) seperti dataset diatas[85], [86]

Model dapat dikatakan baik jika model tersebut dapat memprediksi data yang bukan hanya berasal dari data latih dan data validasi[87], sehingga confusion metrics harus mempunyai nilai akurasi yang tinggi, kemudian mempunyai nilai precision dan recall yang tinggi pula, karena precision dan recall bergantung dari false positive dan false negative, semakin sedikit false positive dan false negative maka akan semakin tinggi nilai precision dan recall, dan nilai akurasi juga bergantung dari banyaknya data yang diprediksi benar[88], [89].

Dari semua model yang ada, model yang mempunyai hasil tertinggi yang dapat memprediksi semua jenis kanker kulit berdasarkan *confusion metrics* (dengan memperhatikan nilai *precision* dan *recall*) dengan iterasi (*epochs*) sebesar 10 kali adalah *xception* dengan nilai *precision* rata-rata dari setiap label berada di 72% dan *recall* yang rata-rata berada di 60% dengan akurasi yang berada di 82%.

Dan dari semua model yang ada, rata-rata model dapat mendekripsi jenis kanker *Melanocytic Nevi* secara benar dengan jumlah yang paling banyak dengan rata-rata dapat mendekripsi 1220 benar dari 1320 jumlah gambar *Melanocytic Nevi* kemudian dengan presentase mendekripsi jenis tersebut benar adalah 80% hingga 90% akurasi.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Telah dilakukan penelitian berupa pembuatan model deep learning berbasis transfer learning dalam mendeteksi penyakit kanker kulit, sehingga dapat diambil kesimpulan:

1. Untuk 10 iterasi, dapat diambil kesimpulan bahwa *xception* merupakan *pre-trained* model dengan performa paling baik karena akurasi dari model tersebut tinggi, validasi akurasi yang tidak terlalu jauh dari akurasi dan juga stabil terhadap iterasi.
2. Parameter yang digunakan adalah *averagepooling2D* dengan *pool size* = 2x2, *layer dropout* sebesar 30%, menggunakan 2 *hidden layer* tambahan, fungsi aktivasi *softmax*, *optimizer adam* dan *learning rate* sebesar 0.0001
3. Untuk metrik evaluasi yang dihasilkan oleh *xception* pada 10 kali pelatihan (*epochs*) adalah, *Accuracy* menghasilkan 90%. *Loss* menghasilkan 36%. *Validation accuracy* menghasilkan 81%. *Validation loss* menghasilkan 77%. Untuk *confusion metrics*, *precision* rata-rata berada di 72% dan *recall* rata-rata berada di 60% dengan akurasi 82%.
4. Kanker kulit yang paling banyak dideteksi dan dapat dibedakan secara baik oleh model adalah kanker kulit dengan jenis Melanocytic Nevi dengan presentasi benar ada di 80% - 90% akurasi dengan rata-rata jumlah deteksi benar adalah 1220 gambar Melanocytic Nevi dari 1320 total gambar Melanocytic Nevi.

5.2 Saran

Untuk penelitian lebih lanjut, disarankan untuk:

1. Menggunakan model yang dibuat sendiri tanpa *pre-trained* model atau menggunakan *pre-trained* model yang lain (diluar 4 *pre-trained* model diatas)
2. Menggunakan *dataset* lain yang digabung dengan *dataset* yang digunakan pada penelitian ini.

3. Meningkatkan akurasi model dengan menambahkan, atau memodifikasi parameter yang ada
4. Membuat aplikasi yang lebih *portable* dan *mobile* lagi, sehingga *user* tidak perlu repot untuk membuka *laptop* atau *PC*.

DAFTAR PUSTAKA

- [1] P. David, A. K. Mackworth, and R. Goebel, “Computational Intelligence and Knowledge 1.1 What Is Computational Intelligence?,” *Comput. Intell. A Log. Approach*, no. Ci, pp. 1–22, 1998, Accessed: Aug. 13, 2021. [Online]. Available: <http://people.cs.ubc.ca/~poole/ci/ch1.pdf>.
- [2] A. Zell *et al.*, “SNNS (Stuttgart Neural Network Simulator),” Springer, Boston, MA, 1994, pp. 165–186.
- [3] P. H. Winston, “Artificial intelligence,” p. 737, 1993.
- [4] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.
- [5] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,” *Neural Networks*, vol. 61, pp. 85–117, Apr. 2014, doi: 10.1016/j.neunet.2014.09.003.
- [6] F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.
- [7] “Artificial Intelligence: Definition, Types, Examples, Technologies | by Chethan Kumar GN | Medium.” <https://chethankumargn.medium.com/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b> (accessed Aug. 13, 2021).
- [8] “The Tumultuous History of the Search for Artificial Intelligence, Daniel Crevier. 1993. Basic Books, New York, NY. 432 pages. ISBN: 0-465-02997-3. \$27.50,” *Bull. Sci. Technol. Soc.*, vol. 14, no. 4, pp. 224–224, Jul. 1994, doi: 10.1177/027046769401400414.
- [9] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. *Artificial Intelligence*. <https://doi.org/10.1017/S0269888900007724>. 2010.
- [10] M. Robins, “The Difference Between Artificial Intelligence, Machine Learning and Deep Learning,” *Intel*, 2020. <https://www.intel.co.id/content/dam/www/public/us/en/ai/images/ai-machine-learning-deep-learning-rwd.png.rendition.intel.web.1648.927.png>

- (accessed Aug. 13, 2021).
- [11] R. Hierons, “Machine learning. Tom M. Mitchell. Published by McGraw-Hill, Maidenhead, U.K., International Student Edition, 1997. ISBN: 0-07-115467-1, 414 pages. Price: U.K. £22.99, soft cover.,” *Softw. Testing, Verif. Reliab.*, vol. 9, no. 3, 1999, doi: 10.1002/(sici)1099-1689(199909)9:3<191::aid-stvr184>3.0.co;2-e.
 - [12] “Machine Learning Model.” <https://devopedia.org/machine-learning-model> (accessed Aug. 13, 2021).
 - [13] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, “Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming,” *Artif. Intell. Des. '96*, pp. 151–170, 1996, doi: 10.1007/978-94-009-0279-4_9.
 - [14] C. Snijders, U. Matzat, and U.-D. Reips, “‘Big Data’: Big Gaps of Knowledge in the Field of Internet Science,” *Int. J. Internet Sci.*, vol. 2012, no. 1, pp. 1–5, 2012, Accessed: Aug. 14, 2021. [Online]. Available: <http://wikipedia-miner.cms.waikato.ac.nz/>.
 - [15] J. Zytkow and J. Rauch, *Principles of Data Mining and Knowledge Discovery: Third European Conference, PKDD'99 Prague, Czech Republic, September 15-18, 1999 Proceedings*. Springer Berlin Heidelberg, 1999.
 - [16] IBM Cloud Education, “Structured vs. Unstructured Data: What’s the Difference? | IBM.” <https://www.ibm.com/cloud/blog/structured-vs-unstructured-data> (accessed Sep. 21, 2021).
 - [17] “Supervised Machine learning - Javatpoint.” <https://www.javatpoint.com/supervised-machine-learning> (accessed Aug. 13, 2021).
 - [18] “Machine learning (ML) is expanding the possibilities | LogPoint.” <https://www.logpoint.com/en/blog/explained-siemply-machine-learning/> (accessed Aug. 13, 2021).
 - [19] F. Lotte, “Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain-computer interfaces,” *Proc. IEEE*,

- vol. 103, no. 6, pp. 871–890, Jun. 2015, doi: 10.1109/JPROC.2015.2404941.
- [20] W. Xing and D. Du, “Dropout Prediction in MOOCs: Using Deep Learning for Personalized Intervention,” *J. Educ. Comput. Res.*, vol. 57, no. 3, pp. 547–570, Jun. 2019, doi: 10.1177/0735633118757015.
- [21] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nat. 2015* 5217553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [22] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Jun. 2012, Accessed: Aug. 13, 2021. [Online]. Available: <https://arxiv.org/abs/1206.5538v3>.
- [23] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Nov. 1958, doi: 10.1037/H0042519.
- [24] D. Nasution, T. H. F. Harumy, E. Haryanto, F. Fachrizal, Julham, and A. Turnip, “A classification method for prediction of qualitative properties of multivariate EEG-P300 signals,” *Proc. 2015 Int. Conf. Autom. Cogn. Sci. Opt. Micro Electro-Mechanical Syst. Inf. Technol. ICACOMIT 2015*, pp. 82–86, Mar. 2016, doi: 10.1109/ICACOMIT.2015.7440180.
- [25] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989, doi: 10.1016/0893-6080(89)90020-8.
- [26] “File:Neuron3.png - Wikimedia Commons.” <https://commons.wikimedia.org/wiki/File:Neuron3.png> (accessed Aug. 13, 2021).
- [27] F. Chen, G. Chen, Q. He, G. He, and X. Xu, “Universal perceptron and DNA-like learning algorithm for binary neural networks: Non-LSBF implementation,” *IEEE Trans. Neural Networks*, vol. 20, no. 8, pp. 1293–1301, 2009, doi: 10.1109/TNN.2009.2023122.
- [28] M. G. C. A. Cimino, W. Pedrycz, B. Lazzerini, and F. Marcelloni, “Using multilayer perceptrons as receptive fields in the design of neural networks,”

- Neurocomputing*, vol. 72, no. 10–12, pp. 2536–2548, Jun. 2009, doi: 10.1016/J.NEUCOM.2008.10.014.
- [29] “File:ArtificialNeuronModel english.png - Wikimedia Commons.” https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png (accessed Aug. 13, 2021).
 - [30] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, 2020.
 - [31] Hyunjulie, “Activation Functions: A Short Summary | by 심현주 | Hyunjulie | Medium.” <https://medium.com/hyunjulie/activation-functions-a-short-summary-8450c1b1d426> (accessed Aug. 14, 2021).
 - [32] M. Nielsen, “Neural Networks and Deep Learning,” Accessed: Aug. 14, 2021. [Online]. Available: <http://neuralnetworksanddeeplearning.com>.
 - [33] L. C. J., “Activation Functions for Artificial Neural Networks,” 2020. <https://www.linkedin.com/pulse/activation-functions-neural-networks-leonardo-calderon-j-> (accessed Aug. 14, 2021).
 - [34] “Activation functions sigmoid, tanh, relu | Develop Paper.” <https://developpaper.com/activation-functions-sigmoid-tanh-relu/> (accessed Aug. 13, 2021).
 - [35] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” 2011.
 - [36] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” *6th Int. Conf. Learn. Represent. ICLR 2018 - Work. Track Proc.*, Oct. 2017, Accessed: Aug. 14, 2021. [Online]. Available: <https://arxiv.org/abs/1710.05941v2>.
 - [37] “Activation Functions in Deep Neural Network | by Barış Ayten | DataDrivenInvestor.” <https://medium.datadriveninvestor.com/activation-functions-in-deep-neural-network-4d8849b70046> (accessed Aug. 13, 2021).
 - [38] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

- [39] “Softmax Activation Function Explained | by Dario Radečić | Towards Data Science.” <https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60> (accessed Aug. 13, 2021).
- [40] Y. M. Mohmad Hassim and R. Ghazali, “Training a Functional Link Neural Network Using an Artificial Bee Colony for Solving a Classification Problems,” vol. 4, 2012.
- [41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning (Adaptive Computation and Machine Learning series)*. MIT Press, 2016.
- [42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nat.* 1986 3236088, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.
- [43] M. Learning *et al.*, “Optimization for Machine Learning,” vol. 2, p. 149, 2012.
- [44] K. P. Murphy, “Machine Learning: A Probabilistic Perspective,” *Mach. Learn. A Probabilistic Perspect.*, pp. 73-78,216-244, 1991, Accessed: Aug. 14, 2021. [Online]. Available: http://link.springer.com/chapter/10.1007/978-94-011-3532-0_2.
- [45] Aurélien Géron, “Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems,” *O'Reilly Media*, p. 851, 2019, Accessed: Sep. 13, 2021. [Online]. Available: <https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>.
- [46] Phung and Rhee, “A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets,” *Appl. Sci.*, vol. 9, p. 4500, 2019, doi: 10.3390/app9214500.
- [47] S. Sarangi, M. Sahidullah, and G. Saha, “Optimization of data-driven filterbank for automatic speaker verification,” *Digit. Signal Process.*, vol. 104, p. 102795, Sep. 2020, doi: 10.1016/J.DSP.2020.102795.
- [48] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2014.
- [49] “How ReLU and Dropout Layers Work in CNNs | Baeldung on Computer

- Science.” <https://www.baeldung.com/cs/ml-relu-dropout-layers> (accessed Aug. 14, 2021).
- [50] L. Perez and J. Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning,” Dec. 2017, Accessed: Aug. 14, 2021. [Online]. Available: <https://arxiv.org/abs/1712.04621v1>.
- [51] K. Hasan Mahmud and S. Al Faraby, “Klasifikasi Citra Multi-Kelas Menggunakan Convolutional Neural Network.”
- [52] N. Vaidya, “How Artificial Neural Networks Unlock Insights from Unstructured Data.” <https://blog.aureusanalytics.com/blog/how-artificial-neural-networks-unlock-insights-from-unstructured-data> (accessed Sep. 21, 2021).
- [53] “python/cpython: The Python programming language.” <https://github.com/python/cpython> (accessed Aug. 14, 2021).
- [54] D. Kuhlman, *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. Platypus Global Media, 2011.
- [55] “About Python™ | Python.org.” <https://www.python.org/about/> (accessed Aug. 14, 2021).
- [56] “numpy/numpy: The fundamental package for scientific computing with Python.” <https://github.com/numpy/numpy> (accessed Aug. 14, 2021).
- [57] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, p. 357, 2020, doi: 10.1038/s41586-020-2649-2.
- [58] “keras-team/keras: Deep Learning for humans.” <https://github.com/keras-team/keras> (accessed Aug. 14, 2021).
- [59] “tensorflow/tensorflow: An Open Source Machine Learning Framework for Everyone.” <https://github.com/tensorflow/tensorflow> (accessed Aug. 14, 2021).
- [60] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” *Proc. 12th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2016*, pp. 265–283, May 2016, Accessed: Aug. 14, 2021. [Online]. Available: <https://arxiv.org/abs/1605.08695v2>.
- [61] W. McKinney, “Data Structures for Statistical Computing in Python,”

- Proc. 9th Python Sci. Conf.*, pp. 56–61, 2010, doi: 10.25080/MAJORA-92BF1922-00A.
- [62] “pandas-dev/pandas: Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.frame objects, statistical functions, and much more.” <https://github.com/pandas-dev/pandas> (accessed Nov. 19, 2021).
 - [63] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.
 - [64] “matplotlib/matplotlib: matplotlib: plotting with Python.” <https://github.com/matplotlib/matplotlib> (accessed Nov. 19, 2021).
 - [65] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011, Accessed: Nov. 19, 2021. [Online]. Available: <http://jmlr.org/papers/v12/pedregosa11a.html>.
 - [66] “scikit-learn/scikit-learn: scikit-learn: machine learning in Python.” <https://github.com/scikit-learn/scikit-learn> (accessed Nov. 19, 2021).
 - [67] B. S. Everitt and A. Skrondal, *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010.
 - [68] “Techniques for handling underfitting and overfitting in Machine Learning | by Manpreet Singh Minhas | Towards Data Science.” <https://towardsdatascience.com/techniques-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9> (accessed Aug. 14, 2021).
 - [69] S. V. Stehman, “Selecting and interpreting measures of thematic classification accuracy,” *Remote Sens. Environ.*, vol. 62, no. 1, pp. 77–89, Oct. 1997, doi: 10.1016/S0034-4257(97)00083-7.
 - [70] “Confusion Matrix untuk Evaluasi Model pada Supervised Learning | by Kuncahyo Setyo Nugroho | Medium.” <https://medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f> (accessed Aug. 14, 2021).
 - [71] “Skin Cancer Facts, Signs & Symptoms & Treatment | Summa Health.”

- <https://www.summahealth.org/medicalservices/cancer/cancer-care/skin-cancer> (accessed Aug. 16, 2021).
- [72] “What Is Cancer? - National Cancer Institute.”
<https://www.cancer.gov/about-cancer/understanding/what-is-cancer> (accessed Aug. 16, 2021).
- [73] U. Leiter and C. Garbe, “Epidemiology of Melanoma and Nonmelanoma Skin Cancer—The Role of Sunlight,” *Adv. Exp. Med. Biol.*, vol. 624, pp. 89–103, 2008, doi: 10.1007/978-0-387-77574-6_8.
- [74] G. Verhoeven, “The reflection of two fields – Electromagnetic radiation and its role in (aerial) imaging,” vol. 55, pp. 13–18, 2017, doi: 10.5281/zenodo.3534245.
- [75] E. Maverakis, Y. Miyamura, M. P. Bowen, G. Correa, Y. Ono, and H. Goodarzi, “Light, including ultraviolet,” *J. Autoimmun.*, vol. 34, no. 3, pp. J247–J257, May 2010, doi: 10.1016/J.JAUT.2009.11.011.
- [76] PDQ Adult Treatment Editorial Board, “Skin Cancer Treatment (PDQ®): Health Professional Version,” *PDQ Cancer Inf. Summ.*, 2002, Accessed: Aug. 16, 2021. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/26389366>.
- [77] B. Ö. Cakir, P. Adamson, and C. Cingi, “Epidemiology and Economic Burden of Nonmelanoma Skin Cancer,” *Facial Plast. Surg. Clin. North Am.*, vol. 20, no. 4, pp. 419–422, Nov. 2012, doi: 10.1016/J.FSC.2012.07.004.
- [78] P. Tschandl, C. Rosendahl, and H. Kittler, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Sci. Data*, vol. 5, no. 1, p. 180161, 2018, doi: 10.1038/sdata.2018.161.
- [79] “Evaluasi Model Machine Learning: Train/Test Split - IlmudataPy.”
<https://ilmudatapy.com/evaluasi-model-machine-learning-dengan-train-test-split/> (accessed Sep. 21, 2021).
- [80] “aleju/imgaug: Image augmentation for machine learning experiments.”
<https://github.com/aleju/imgaug> (accessed Sep. 21, 2021).

- [81] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Aug. 2016, Accessed: Aug. 16, 2021. [Online]. Available: <https://arxiv.org/abs/1608.06993v5>.
- [82] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 2818–2826, Dec. 2015, Accessed: Aug. 16, 2021. [Online]. Available: <https://arxiv.org/abs/1512.00567v3>.
- [83] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2015, Accessed: Aug. 16, 2021. [Online]. Available: <https://arxiv.org/abs/1512.03385v1>.
- [84] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, Oct. 2016, Accessed: Aug. 16, 2021. [Online]. Available: <https://arxiv.org/abs/1610.02357v3>.
- [85] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.
- [86] P. Branco, L. Torgo, and R. Ribeiro, “A Survey of Predictive Modelling under Imbalanced Distributions,” May 2015, Accessed: Sep. 13, 2021. [Online]. Available: <https://arxiv.org/abs/1505.01658v2>.
- [87] Y. Xu and R. Goodacre, “On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning,” *J. Anal. Test.* 2018 23, vol. 2, no. 3, pp. 249–262, Oct. 2018, doi: 10.1007/S41664-018-0068-2.

- [88] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics* 2019 211, vol. 21, no. 1, pp. 1–13, Jan. 2020, doi: 10.1186/S12864-019-6413-7.
- [89] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation,” *AAAI Work. - Tech. Rep.*, vol. WS-06-06, pp. 24–29, 2006, doi: 10.1007/11941439_114.

LAMPIRAN

a. *Script code untuk Training*

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd

from tqdm import tqdm

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

from tensorflow.keras import Sequential
from tensorflow.keras.applications import *
from tensorflow.keras.regularizers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.losses import *
from tensorflow.keras.layers import Input, AveragePooling2D,
Dense, Dropout, Flatten
from tensorflow.keras.preprocessing.image import
ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.utils import get_file, plot_model
from tensorflow.keras.models import load_model

from imblearn.over_sampling import RandomOverSampler

# Load the saved models/checkpoints models from Google Drive

def load_models(url, summary=False, filenames='tmp'):
    model_download = download_file(url, unzip=False,
remove_after_unzip=True, google_drive=True, ext="h5",
filename=filenames)
    model = load_model(model_download)
    if summary:
        model.summary()
    return model

class_names_short = ['AKIEC', 'BCC', 'BKL', 'DF', 'NV', 'MEL',
'VASC']
class_names = ['Actinic keratoses', 'Basal cell carcinoma',
'Benign keratosis-like lesions', 'Dermatofibroma',
'Melanocytic nevi', 'Melanoma', 'Vascular
lesions']
```

```

def define_dataset(csv):
    df = pd.read_csv(csv) # Reading the csv files from pandas
    img = df['image_id'] # Define the Feature
    target = df['cell_type_idx'].values # Define the Labels

    return img, target

# Defining dataset into labels and features
img, target =
define_dataset(f'{Dataset().tmp}/{filename}/metadata.csv')

def split_data(x, y,
               test_size=0.2,
               validation_size=0.05,
               random_state=1234):
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=test_size, random_state=random_state)
    x_train, x_val, y_train, y_val = train_test_split(x, y,
test_size=validation_size, random_state=random_state)

    return x_train, x_val, x_test, y_train, y_val, y_test

# Splitting into train, valid and test with balancing all of the
data
x_train, x_val, x_test, y_train, y_val, y_test = split_data(img,
target, test_size=0.2, validation_size=0.2)

def to_tensor(image_paths,
              size=64):
    imgs = []

    for i in tqdm(image_paths):
        img =
load_img(f'{Dataset().tmp}/{filename}/HAM10000/{i}.jpg',
target_size=(size, size)) # Load the
Image then resized them
        img = img_to_array(img) # Convert the Image to arrays
        img = img.astype(np.float32) / 255 # Rescale the Images
        imgs.append(img) # Load all of Images to datasets
    return np.stack(imgs, axis=0) # Join a sequence of arrays
along a new axis in other hands, transpose the axis

x_train = to_tensor(x_train, size=150)

```

```

x_val = to_tensor(x_val, size=150)
x_test = to_tensor(x_test, size=150)

# Scoring saved models/checkpoint models

def scoring(model, x_test, y_test, class_names):
    def plot_confusionmat(confusion_mat, class_names,
cmap='GnBu'):
        fig, ax = plt.subplots(figsize=(10, 10))

        sns.heatmap(confusion_mat, annot=True, fmt='.2f',
                    xticklabels=[f"{c}" for c in class_names],
                    yticklabels=[f"{c}" for c in class_names],
                    cmap=cmap)

        plt.ylabel('Actual')
        plt.xlabel('Predicted')
        plt.show()

    score = model.evaluate(x_test, y_test, verbose=0)
    predicting = model.predict(x_test)
    pred = np.argmax(predicting, axis=1)
    report = classification_report(y_true=y_test, y_pred=pred,
target_names=class_names)
    conf = confusion_matrix(y_test, pred)

    plot_confusionmat(conf, class_names)
    print("\n%s: %.2f%%\n" % (model.metrics_names[1], score[1] *
100))
    print("\nAccuracy: {:.2f}%\n".format(accuracy_score(y_test,
pred) * 100))
    print(report)

# Defining the models

def trainable_model(x_train, y_train, x_val, y_val,
batch_size=64, dropout=0.25, base_model='MobileNet',
                    regularizer=0.01, learning_rate=1e-4,
epochs=15, verbose=1,
                    metrics='accuracy', pool_size=2,
rotation_range=30,
                    zoom_range=0.15, width_shift_range=0.2,
shear_range=0.15,
                    horizontal_flip=True, fill_mode="nearest",
height_shift_range=0.2,
                    weights="imagenet", include_top=False,
activation='relu',

```

```

                activation_final='softmax'):

dataset = (x_train,
           y_train,
           x_val,
           y_val)

BATCH_SIZE = batch_size
INPUT_SHAPE = dataset[0][0].shape
trainX = dataset[0]
trainY = dataset[1]
valX = dataset[2]
valY = dataset[3]

train_gen = ImageDataGenerator(rotation_range=rotation_range,
                               zoom_range=zoom_range,
                               width_shift_range=width_shift_range,
                               height_shift_range=height_shift_range,
                               shear_range=shear_range,
                               horizontal_flip=horizontal_flip,
                               fill_mode=fill_mode)

train_dataset = train_gen.flow(trainX,
                               trainY,
                               batch_size=BATCH_SIZE)

valid_dataset = train_gen.flow(valX,
                               valY,
                               batch_size=BATCH_SIZE)

if base_model is 'InceptionV3':
    base_model = InceptionV3(weights=weights,
                             include_top=include_top,
                             input_tensor=Input(shape=INPUT_SHAPE)) # Doesn't support by this
                                             # dataset

elif base_model is 'Xception':
    base_model = Xception(weights=weights,
                           include_top=include_top,
                           input_tensor=Input(shape=INPUT_SHAPE)) # Xception: 95%, Epochs:
                                             # 9

elif base_model is 'DenseNet201':
    base_model = DenseNet201(weights=weights,
                             include_top=include_top,

```

```

input_tensor=Input(shape=INPUT_SHAPE)) # DenseNet201: 82%
Epochs: 6

    elif base_model is 'ResNet152V2':
        base_model = ResNet152V2(weights=weights,
                                include_top=include_top,

    input_tensor=Input(shape=INPUT_SHAPE)) # ResNet152V2: 75%,
Epochs: 6

    else:
        raise ValueError(f"base_model is {base_model} which is
not compatible for this model")

    base_model.trainable = False

    model = Sequential([base_model,
                        AveragePooling2D(pool_size=(pool_size,
pool_size)),
                        Flatten(),
                        Dense(64, activation=activation),
                        Dropout(dropout),
                        Dense(16, activation=activation),
                        Dense(7, activation=activation_final,
kernel_regularizer=l2(regularizer))])

    model.compile(optimizer=Adam(learning_rate=learning_rate),
                  loss=SparseCategoricalCrossentropy(),
                  metrics=[metrics])

    model.summary()

    history = model.fit(train_dataset,
                         epochs=epochs,
                         validation_data=valid_dataset,
                         verbose=verbose
                         )

    return history, model

list_model = ['DenseNet201', 'InceptionV3', 'ResNet152V2',
'Xception']

for model_name in list_model:
    base_model = model_name

    history, model = trainable_model(x_train, y_train, x_val,
y_val, x_test, y_test,

```

```
        fine_tuning=True, epochs=10,
base_model=base_model,
                    dropout=0.3,
regularizer=0.01, batch_size=128,
                                summary=True)

    scoring(model, x_test, y_test, class_names) # Scoring the
models

    plot_model(model, to_file=f"{base_model}.png",
show_shapes=True)
```

b. Dokumentasi proses penelitian

```
# We only balanced the Train and Validation data
# So, if we set oversample to True for x_test
# There will be errors
x_train = to_tensor(x_train, size=150) #size=299
x_val = to_tensor(x_val, size=150) #size=299
# x_test = to_tensor(x_test)

# Unbalanced datasets
#x_train = to_tensor(x_train, oversample=False)
#x_val = to_tensor(x_val, oversample=False)
x_test = to_tensor(x_test, oversample=False, size=150) # Since this is only the test dataset, no need to balanced this #size=299

100% | 2390/2390 [00:04<00:00, 510.30it/s]
100% | 124/124 [00:00<00:00, 514.74it/s]
100% | 474/474 [00:01<00:00, 364.89it/s]
```

Gambar Lampiran 1. Proses Konversi Gambar ke Tensor

```
[ ]: class_names = ['Benign', 'Malignant']

def show_batch(x, y, figsize=12, subfigsize=5, n=25, no_axis=True):
    plt.figure(figsize=(figsize, figsize)) # Defining figure of the datasets

    for i in range(n): # Iterating through n datasets randomly
        ax = plt.subplot(subfigsize, subfigsize, i+1) # Defining frontend of figure
        plt.imshow(x[i]) # Showing the picture from datasets
        plt.title(class_names[y[i]]) # Showing class names

    # Remove axis from the figure
    if no_axis:
        plt.axis('off')

[ ]: # Showing images from the datasets
show_batch(x_test, y_test)
```



Gambar Lampiran 2. Proses Menampilkan Beberapa Gambar Kanker Kulit

```

callbacks = EarlyStopping(monitor='val_loss', mode='min', patience=5, verbose=1)
checkpoints = ModelCheckpoint('HAM10000_Xception_dropout0045' + "{accuracy:.2f}acc.h5", verbose=1) # Create Keras SavedModel as checkpoint
history, model = trainable_model(x_train, y_train, x_val, y_val, x_test, y_test,
                                 fine_tuning=True, epochs=20, base_model='Xception',
                                 dropout=0.05, regularizer=0.1, batch_size=32,
                                 callbacks=callbacks, summary=True, checkpoint=checkpoints)

Model: "sequential_6"
-----  

Layer (type)          Output Shape       Param #  

=====-----  

xception (Functional)    (None, 5, 5, 2048)      20861480  

-----  

average_pooling2d_6 (Average) (None, 2, 2, 2048)      0  

-----  

flatten_6 (Flatten)     (None, 8192)        0  

-----  

dense_18 (Dense)        (None, 64)         524352  

-----  

dropout_6 (Dropout)     (None, 64)         0  

-----  

dense_19 (Dense)        (None, 16)         1040  

-----  

dense_20 (Dense)        (None, 2)          34  

=====-----  

Total params: 21,386,966  

Trainable params: 21,332,378  

Non-trainable params: 54,528

```

Gambar Lampiran 3. Proses Pelatihan Salah Satu Pre-Trained Model (Xception)

```

-----  

Epoch 1/20  

75/75 [=====] - 24s 269ms/step - loss: 0.9867 - accuracy: 0.6380 - val_loss: 1.0038 - val_accuracy: 0.5833  

Epoch 00001: saving model to HAM10000_Xception_dropout0045_0.70acc.h5  

Epoch 2/20  

75/75 [=====] - 20s 263ms/step - loss: 0.7626 - accuracy: 0.8143 - val_loss: 0.9263 - val_accuracy: 0.7500  

Epoch 00002: saving model to HAM10000_Xception_dropout0045_0.83acc.h5  

Epoch 3/20  

75/75 [=====] - 20s 265ms/step - loss: 0.6865 - accuracy: 0.8427 - val_loss: 0.9140 - val_accuracy: 0.7667  

Epoch 00003: saving model to HAM10000_Xception_dropout0045_0.85acc.h5  

Epoch 4/20  

75/75 [=====] - 20s 266ms/step - loss: 0.6083 - accuracy: 0.8781 - val_loss: 0.7828 - val_accuracy: 0.8417  

Epoch 00004: saving model to HAM10000_Xception_dropout0045_0.89acc.h5  

Epoch 5/20  

75/75 [=====] - 20s 267ms/step - loss: 0.5716 - accuracy: 0.8917 - val_loss: 0.5673 - val_accuracy: 0.8750  

Epoch 00005: saving model to HAM10000_Xception_dropout0045_0.90acc.h5  

Epoch 6/20  

75/75 [=====] - 20s 269ms/step - loss: 0.5021 - accuracy: 0.9204 - val_loss: 0.6137 - val_accuracy: 0.9000  

Epoch 00006: saving model to HAM10000_Xception_dropout0045_0.93acc.h5  

Epoch 7/20  

75/75 [=====] - 20s 270ms/step - loss: 0.4688 - accuracy: 0.9367 - val_loss: 0.5320 - val_accuracy: 0.9000  

Epoch 00007: saving model to HAM10000_Xception_dropout0045_0.94acc.h5  

Epoch 8/20  

75/75 [=====] - 20s 272ms/step - loss: 0.4385 - accuracy: 0.9431 - val_loss: 0.5299 - val_accuracy: 0.8750

```

Gambar Lampiran 4. Proses Pelatihan Salah Satu Pre-Trained Model (Xception)

```

# Save the models to Google Drive
google_drive_mount(mounting=True)
copy("/content/HAM10000_Xception_dropout0045_0.97acc.h5", "/content/drive/MyDrive/fix-datasets-new")
# Saved it to google drive: https://drive.google.com/file/d/1-00DyEWBJcERmvXXM5Ejk_VoxV7gI6SW/view?usp=sharing

Mounted at /content/drive
'/content/drive/MyDrive/fix-datasets-new/HAM10000_Xception_dropout0045_0.97acc1.h5'

We successfully created the model with 97% of accuracy

```

Gambar Lampiran 5. Menyimpan Model Untuk Kemudian di Deploy

```

# Change directory to Google Drive
if not os.path.exists(f"{os.getcwd()}/fix-datasets"):
    os.mkdir(f"{os.getcwd()}/fix-datasets")

os.chdir(f"{os.getcwd()}/fix-datasets")

if not os.path.exists(f'{os.getcwd()}/skin-cancer-mnist-ham10000'):
    os.mkdir(f'{os.getcwd()}/skin-cancer-mnist-ham10000') # Make directory for MNIST HAM 10000

os.chdir(f'{os.getcwd()}/skin-cancer-mnist-ham10000') # Change directory to MNIST HAM 10000

# Downloading the datasets using Kaggle API
!kaggle datasets download -d kmader/skin-cancer-mnist-ham10000

Downloading skin-cancer-mnist-ham10000.zip to /content/fix-datasets/skin-cancer-mnist-ham10000
100% 5.20G/5.20G [01:53<00:00, 62.8MB/s]
100% 5.20G/5.20G [01:53<00:00, 49.1MB/s]

```

Gambar Lampiran 6. Proses Download dari Kaggle

```

# Extracting the datasets
local_zip = f'{os.getcwd()}/skin-cancer-mnist-ham10000.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall(f'{os.getcwd()}/')
zip_ref.close()

# Removing the old 'huge' zip
os.remove(f'{os.getcwd()}/skin-cancer-mnist-ham10000.zip')

# Resizing and moving function
def resize_img(SOURCE, DEST, SIZE=150, remove_after_resize=False):
    files = []

    if not os.path.exists(DEST):
        os.mkdir(DEST)

    for filename in os.listdir(SOURCE):
        file = SOURCE + filename
        if os.path.getsize(file) > 0:
            files.append(filename)
        else:
            print(filename + " is zero length, so ignoring.")

    print(f"{SOURCE}: {len(files)}")

    for filename in files:
        if '.jpg' in filename:
            img = cv2.imread(f'{SOURCE}/{filename}')
            resize_img = cv2.resize(img, (SIZE,SIZE))
            cv2.imwrite(f'{SOURCE}/{filename}', resize_img)
            move(f'{SOURCE}/{filename}', f'{DEST}/{filename}')

    print(f"Successfully moving {SOURCE} to {DEST}")

    print(f"After Moving {DEST}: {len(os.listdir(DEST))}\n") # Checking how much file in HAM10000 to make sure there is no file lost

```

Gambar Lampiran 7. Proses Ekstraksi Dataset dan Resizing Dataset

```

# Resizing the image
resize_img(f'{os.getcwd()}/HAM10000_images_part_1/', f'{os.getcwd()}/HAM10000/', remove_after_resize=True)
resize_img(f'{os.getcwd()}/HAM10000_images_part_2/', f'{os.getcwd()}/HAM10000/', remove_after_resize=True)

# Remove some dir, to make directory more clean
rmtree(f'{os.getcwd()}/ham10000_images_part_1')
rmtree(f'{os.getcwd()}/ham10000_images_part_2')

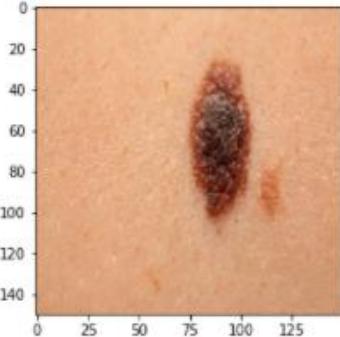
/content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000_images_part_1/: 5000
Successfully moving /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ to /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000/
After Moving /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000/: 5000

/content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000_images_part_2/: 5015
Successfully moving /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ to /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000/
After Moving /content/fix-datasets/skin-cancer-mnist-ham10000/HAM10000/: 10015

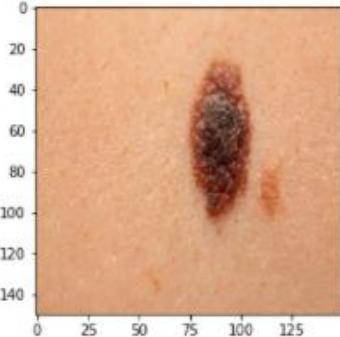
```

Gambar Lampiran 8. Proses Ekstraksi Dataset dan Resizing Dataset

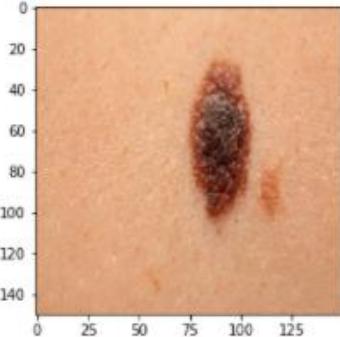
```




```



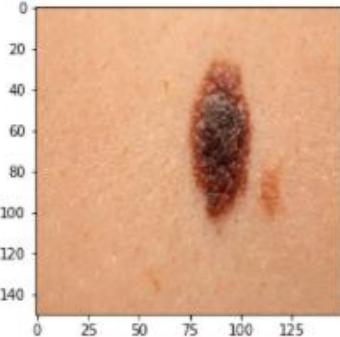

```



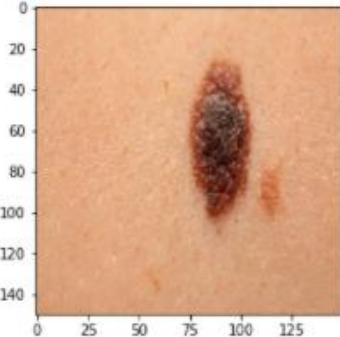

```



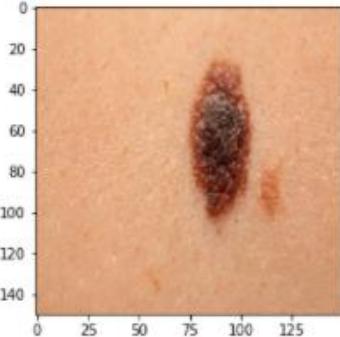

```



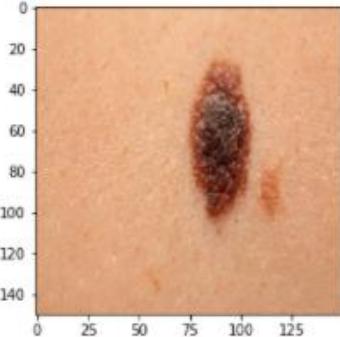

```



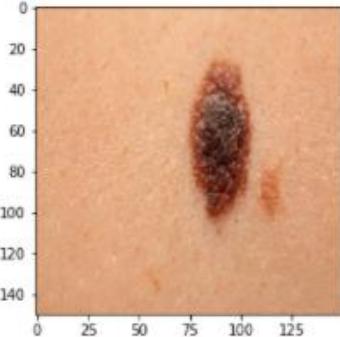

```



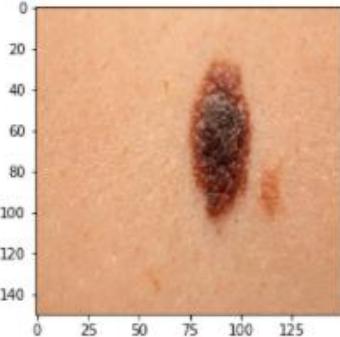

```



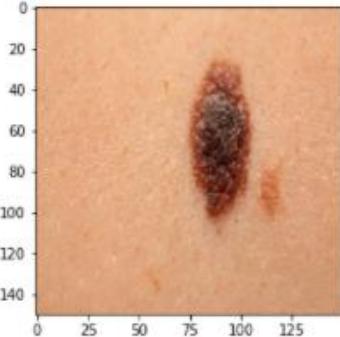

```



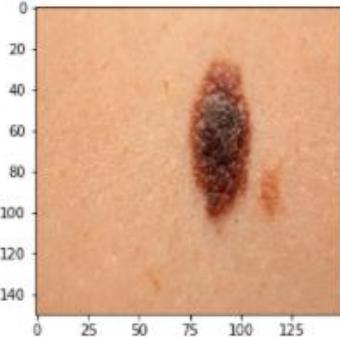

```



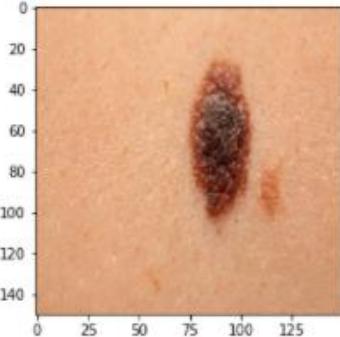

```



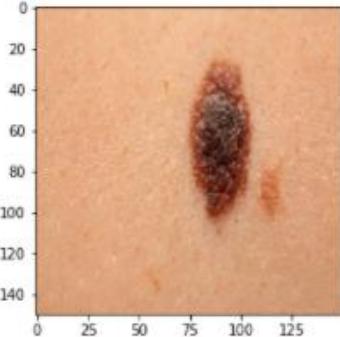

```



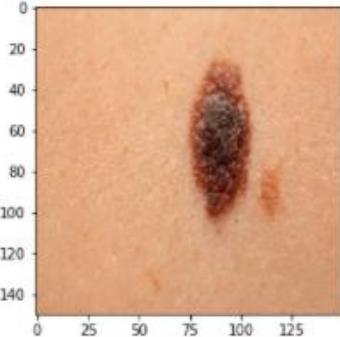

```



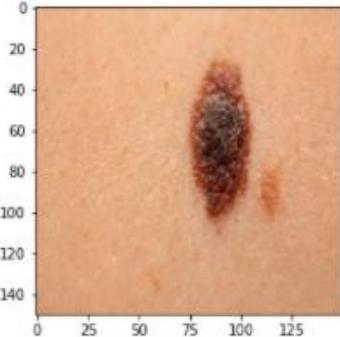

```



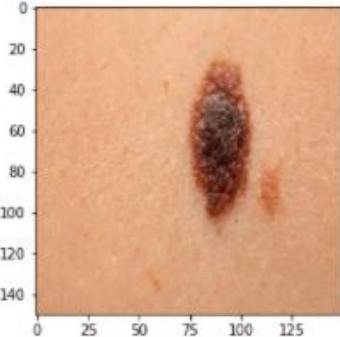

```



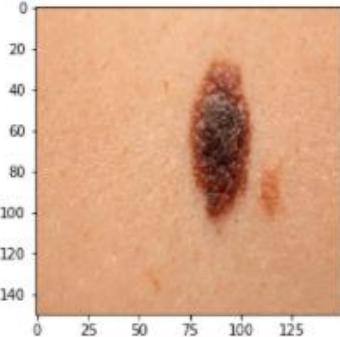

```



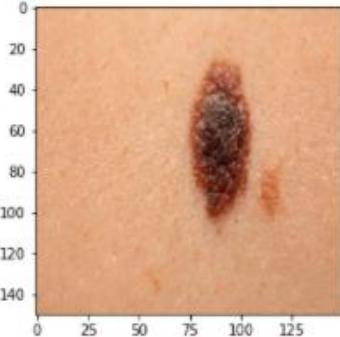

```



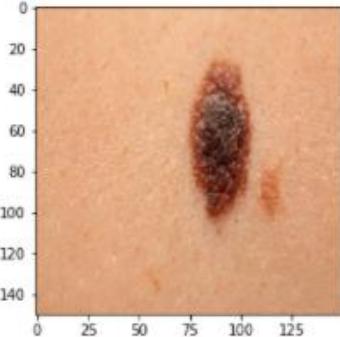

```



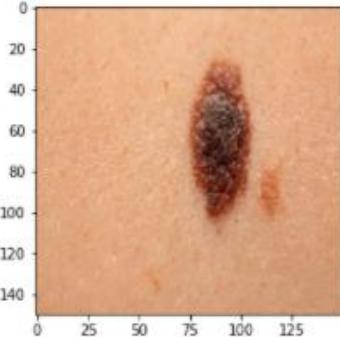

```



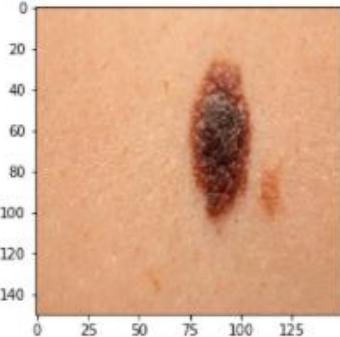

```



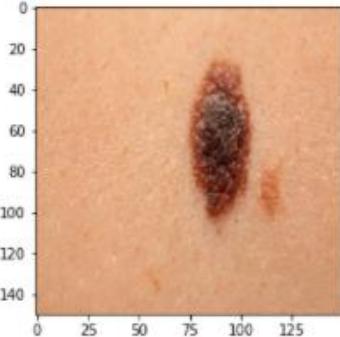

```



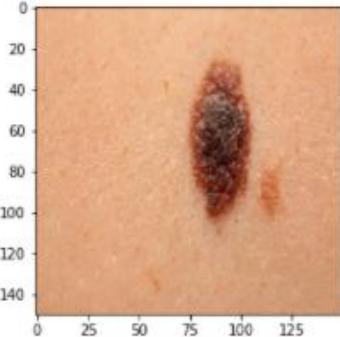

```



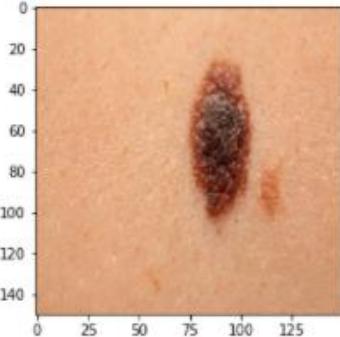

```



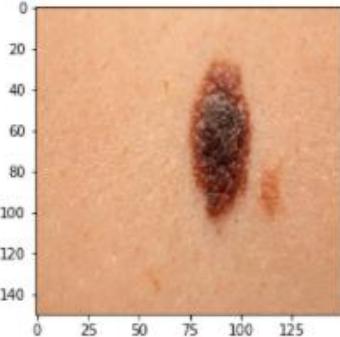

```



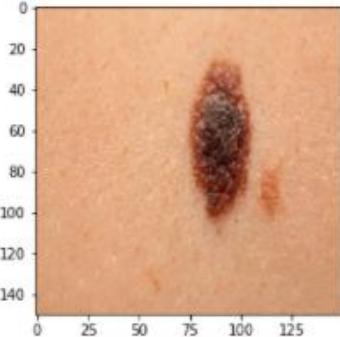

```



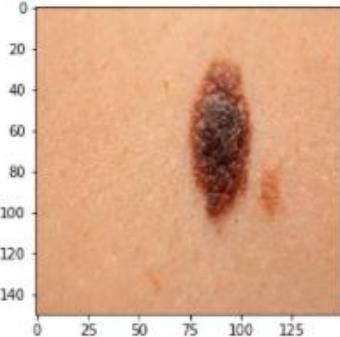

```



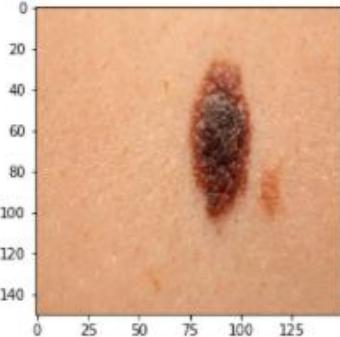

```



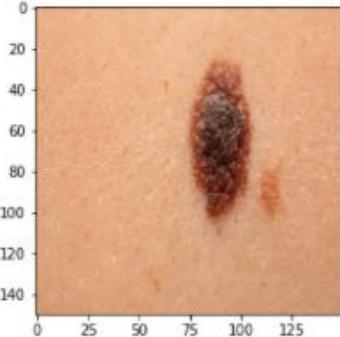

```



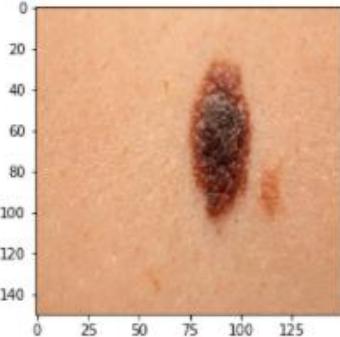

```



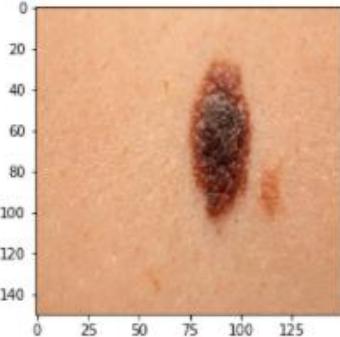

```



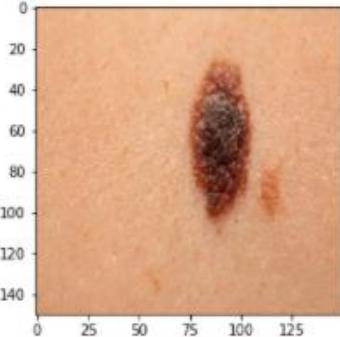

```



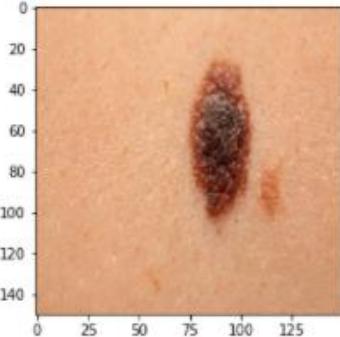

```



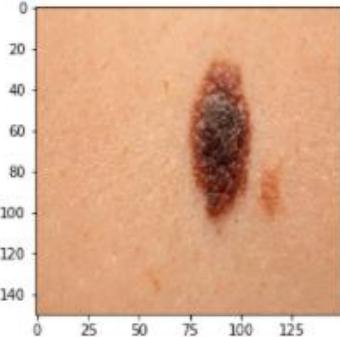

```



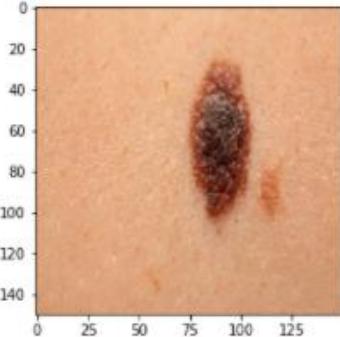

```



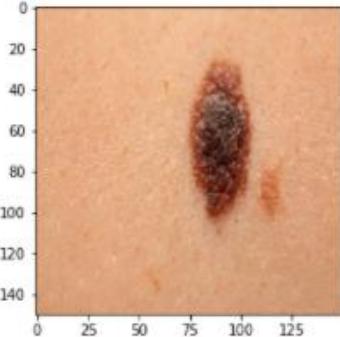

```



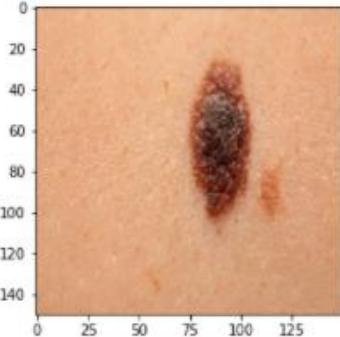

```



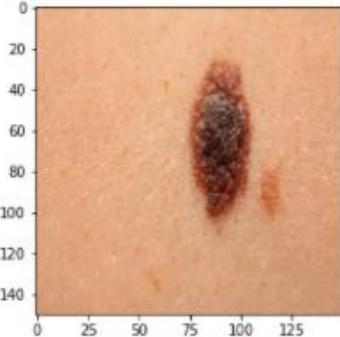

```



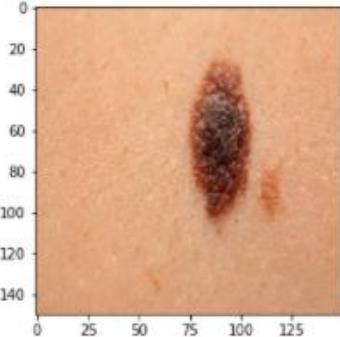

```



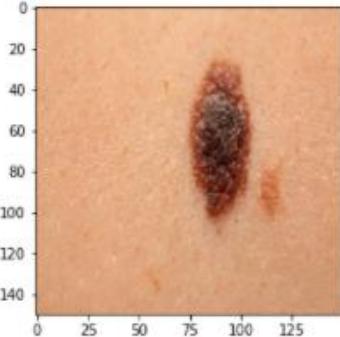

```



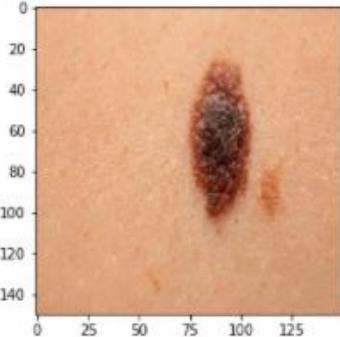

```



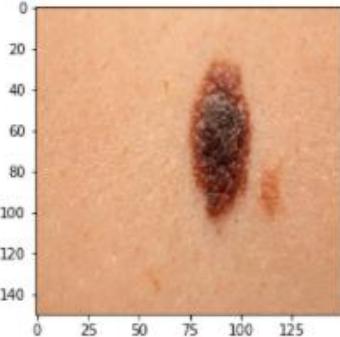

```



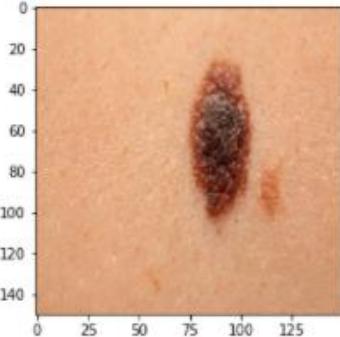

```



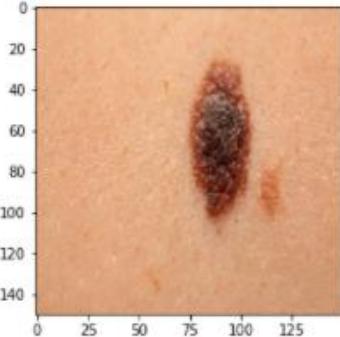

```



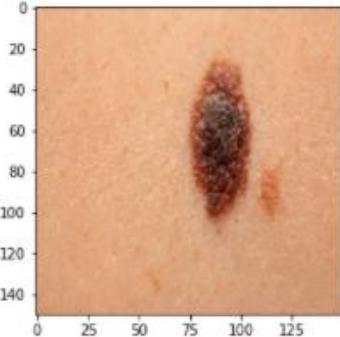

```



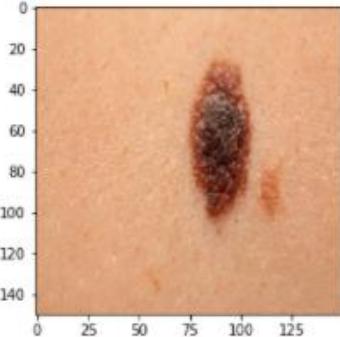

```



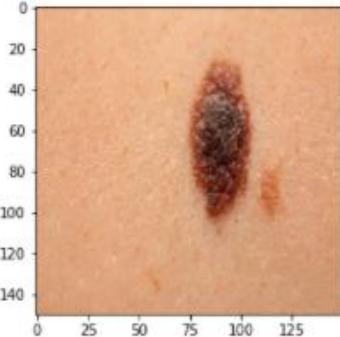

```



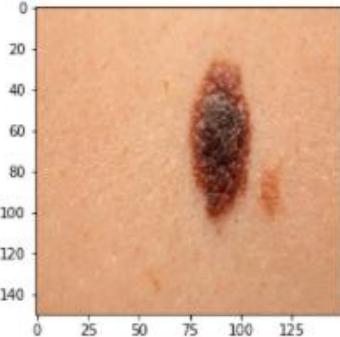

```



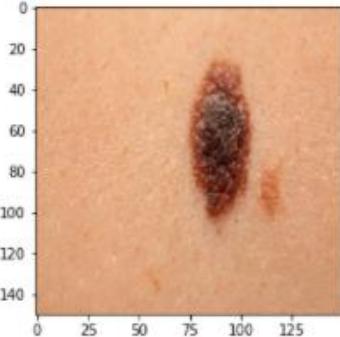

```



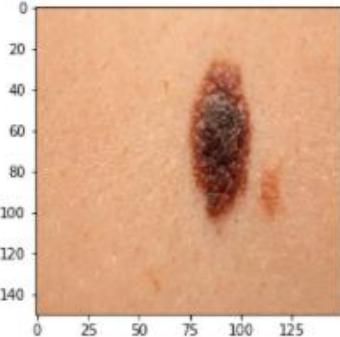

```



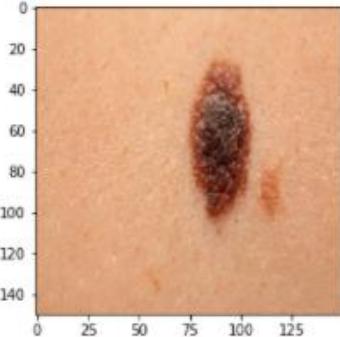

```



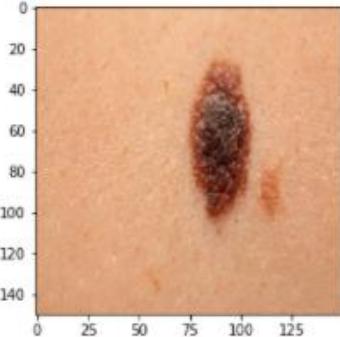

```



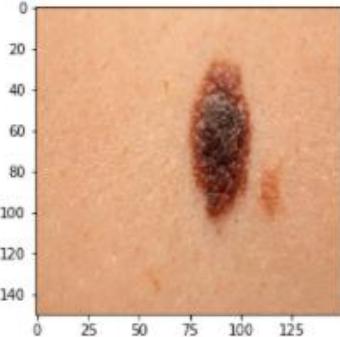

```



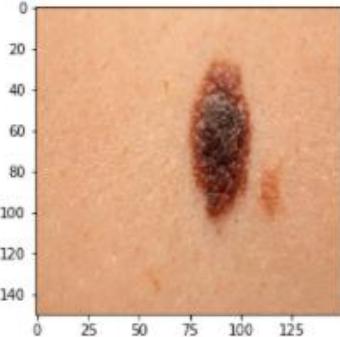

```



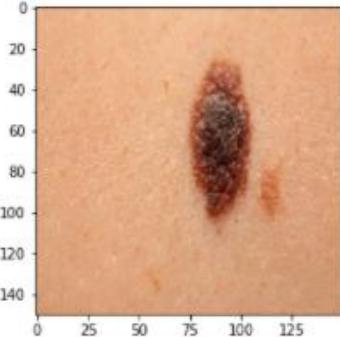

```



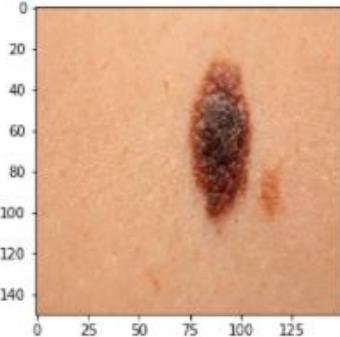

```



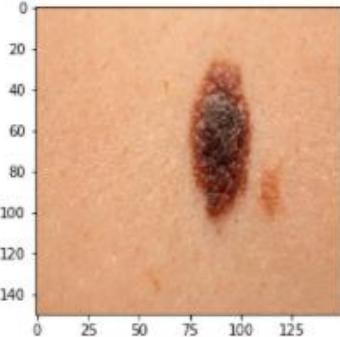

```



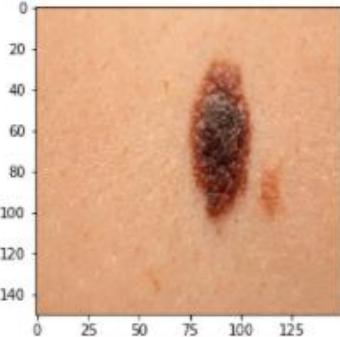

```



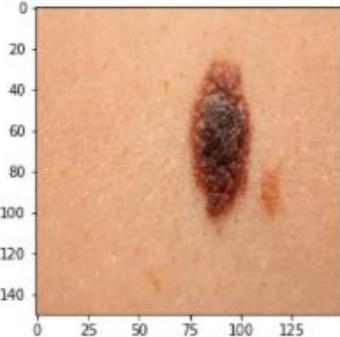

```



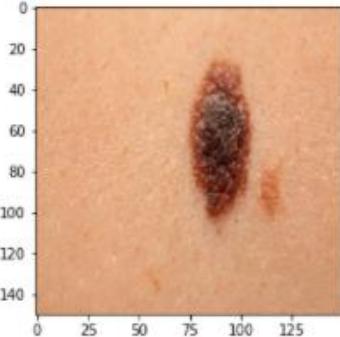

```



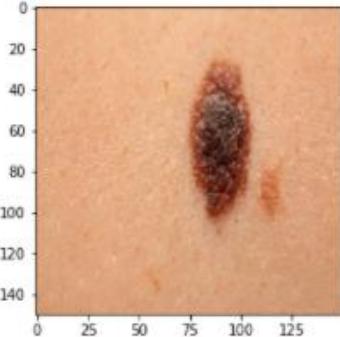

```



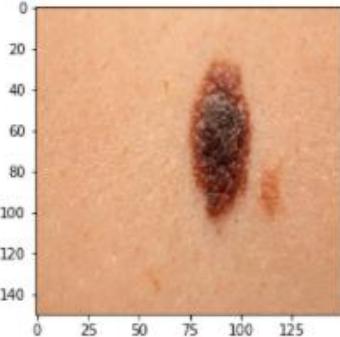

```



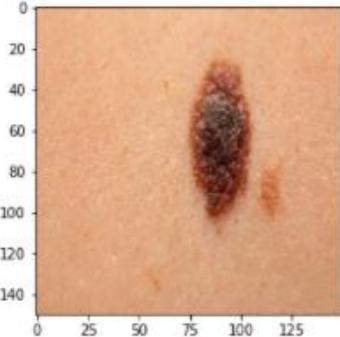

```



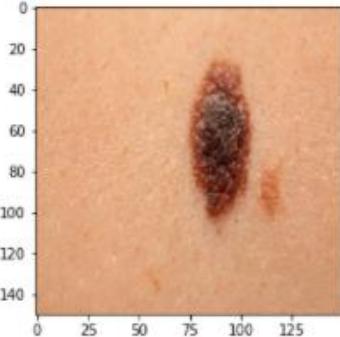

```



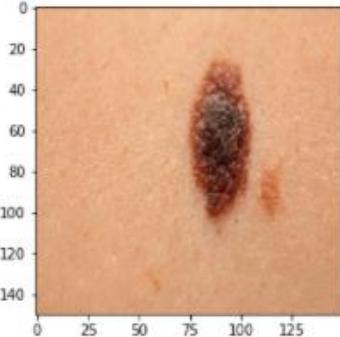

```



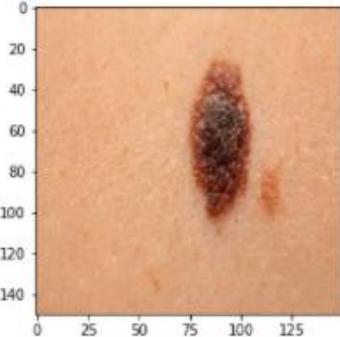

```



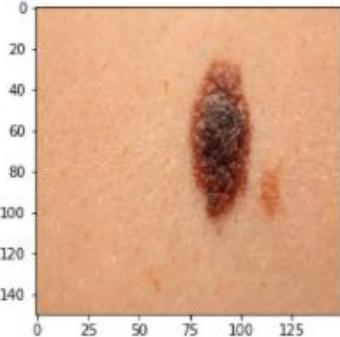

```



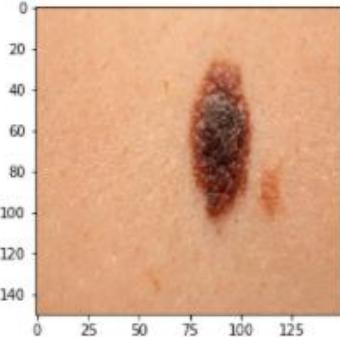

```



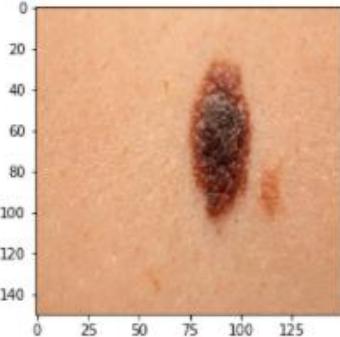

```



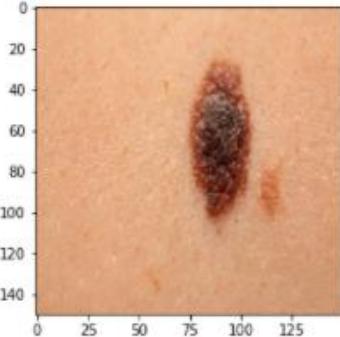

```



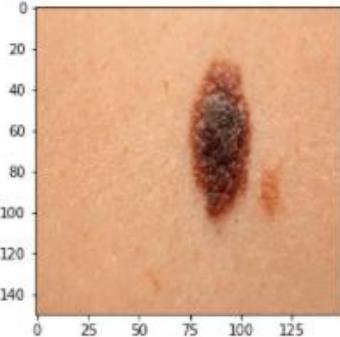

```



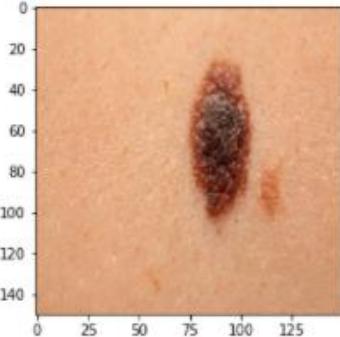

```



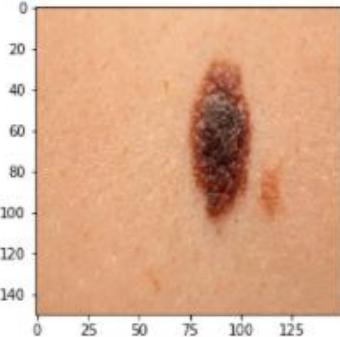

```



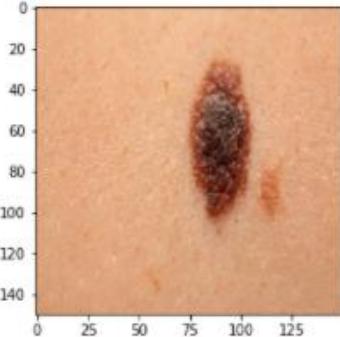

```



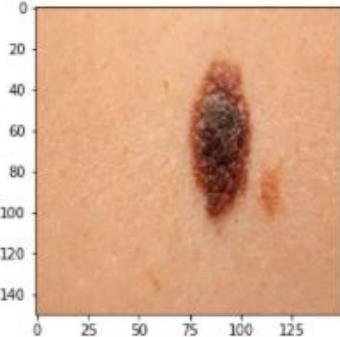

```



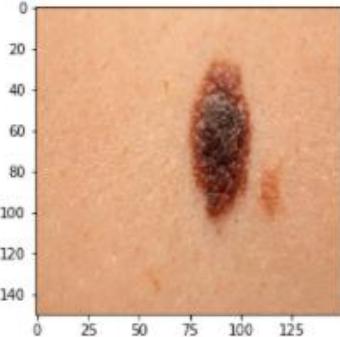

```



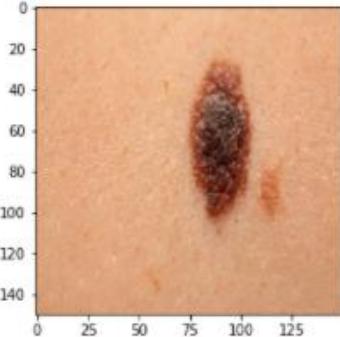

```



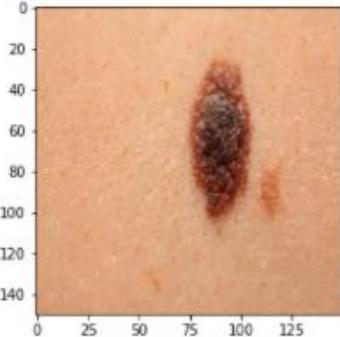

```



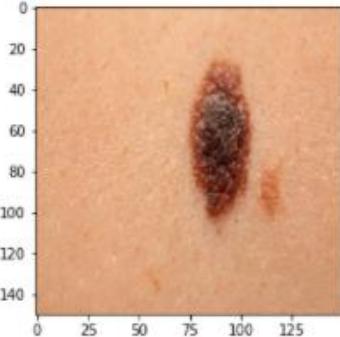

```



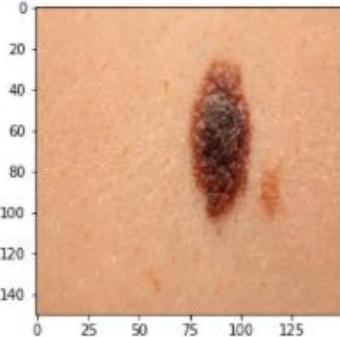

```



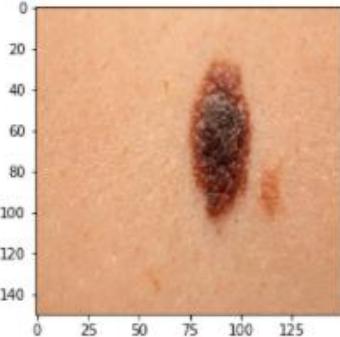

```



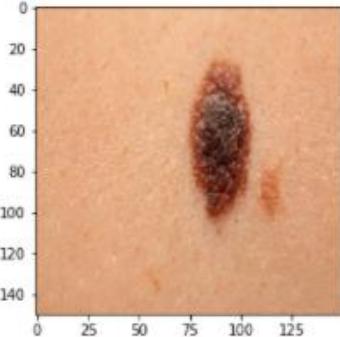

```



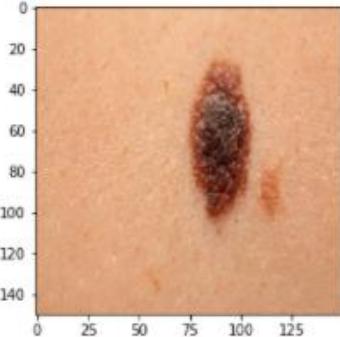

```



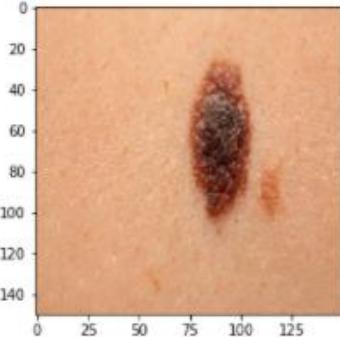

```



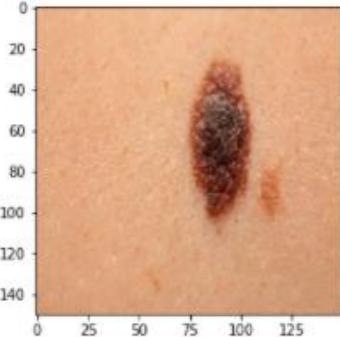

```



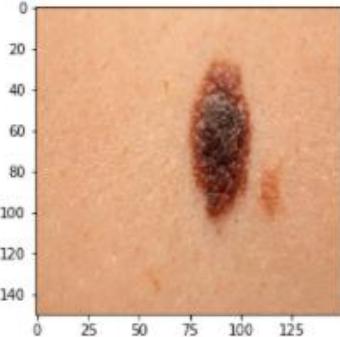

```



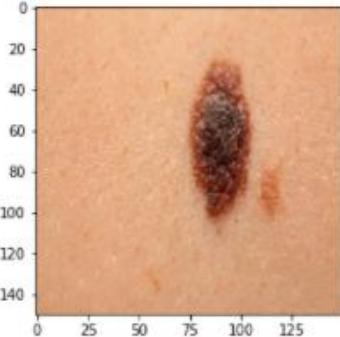

```



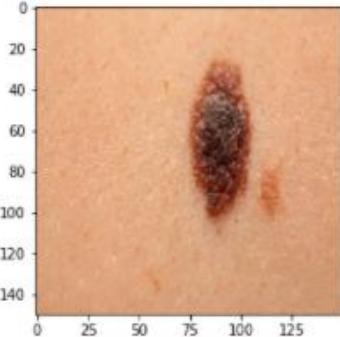

```



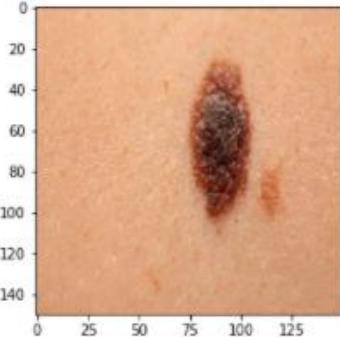

```



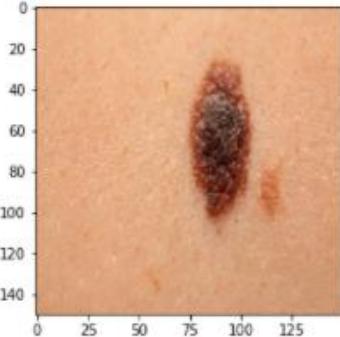

```



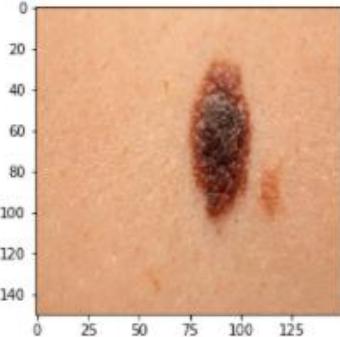

```




```




```


<img alt="A screenshot of a
```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```


```

```

app.py > ...
app.py > ...
munaufal, 2 months ago | 1 author (munaufal)
1  from flask import Flask, request, render_template      munaufal, 2 months ago + initial commit
2  from gevent.pywsgi import WSGIServer
3  from models.predict import get_model_predict, model_predict, \
4      model_path, byte_array_predict,\ 
5      get_predict, image_predict, get_labels
6
7  app = Flask(__name__)
8
9  LABEL_PATH = "models/labels.txt"
10 class_names_long = get_labels(LABEL_PATH)
11
12 MODEL_PATH = 'models/model.h5'
13 model = model_path(MODEL_PATH)
14
15
16 @app.route('/', methods=['GET'])
17 def index():
18     return render_template('index.html')
19
20
21 @app.route('/predict', methods=['GET', 'POST'])
22 def predict():
23     if request.method == 'POST':
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
[Upgrade now, or check out the release page at:
https://aka.ms/PowerShell-Release?tag=v7.2.0]
WARNING: The prediction 'ListView' is temporarily disabled because the current window size of the console is too small. To use the 'ListView', please make sure the 'WindowWidth' is not less than '54' and the 'WindowHeight' is not less than '15'.
Loading personal and system profiles took 5437ms.
~ > .. > deployment > master > 3.9.7
9:47
16:4

```

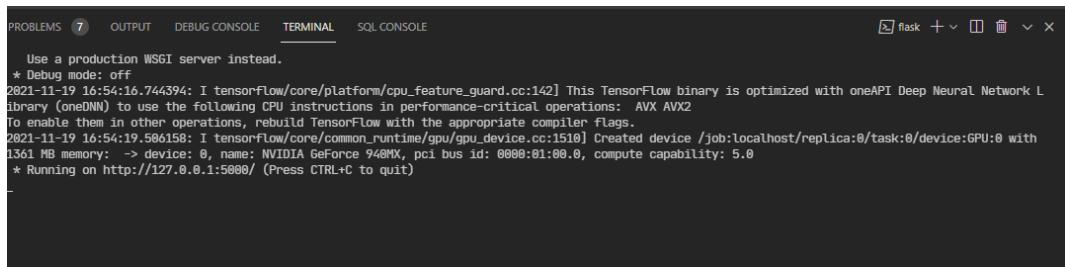
Gambar Lampiran 10. Proses Pembuatan API

```

index.html > ...
templates > index.html > ...
munaufal, 2 months ago | 1 author (munaufal)
1  {% extends "base.html" %} {% block content %}      munaufal, 2 months ago + initial commit
2
3  <div class="main">
4      <div class="title">
5          <h1>Klasifikasi Penyakit Kulit</h1>
6          <p align="center">
7              <small>Dengan Metode Convolutional Neural Network</small>
8          </p>
9      </div>
10
11     <div class="panel">
12         <input id="file-upload" class="hidden" type="file" accept="image/x-png,image/gif,image/jpeg" />
13         <label for="file-upload" id="file-drag" class="upload-box">
14             <div id="upload-caption">Tarik gambar kesini atau Klik untuk memilih</div>
15             <img id="image-preview" class="hidden" />
16         </label>
17     </div>
18     <div style="margin-bottom: 2rem;">
19         <input type="button" value="Prediksi" class="button" onclick="submitImage();"/>
20         <input type="button" value="Hapus Gambar" class="button" onclick="clearImage();"/>
21     </div>
22
23     <div id="image-box">

```

Gambar Lampiran 11. Proses Pembuatan Aplikasi Web



A screenshot of a terminal window titled "flask". The window shows log messages from a Flask application. The messages include:

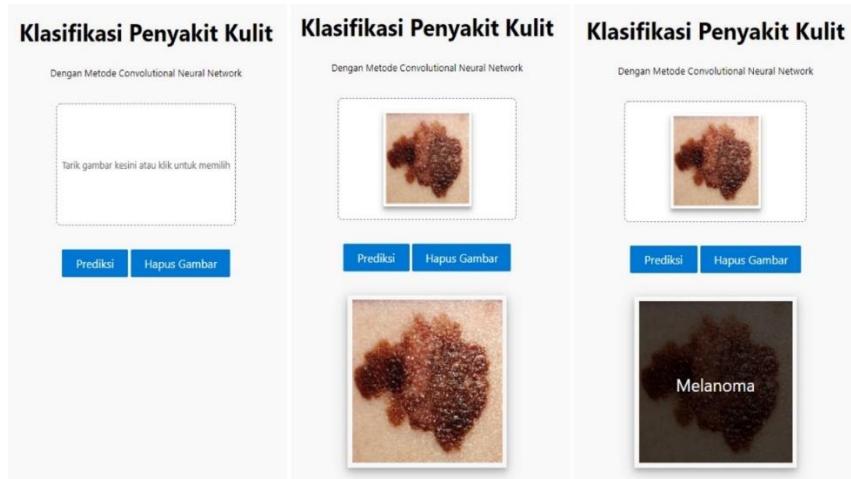
```
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
flask + - ×

Use a production WSGI server instead.
* Debug mode: off
[2021-11-19 16:54:16.744394: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2 To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[2021-11-19 16:54:19.506158: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 1361 MB memory; -> device: 0, name: NVIDIA GeForce 940MX, pci bus id: 0000:01:00.0, compute capability: 5.0
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar Lampiran 12. Proses Running Server Flask



Gambar Lampiran 13. Aplikasi Web



Gambar Lampiran 14. Aplikasi Web