

**PENINGKATAN AKURASI PADA KLASIFIKASI LESI KULIT MANUSIA  
DENGAN *DATA BALANCING* MENGGUNAKAN PENDEKATAN  
*CONVOLUTIONAL NEURAL NETWORK* (CNN)**

**SKRIPSI**

**PUAN ABIDAH NITISARA  
201401009**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

**PENINGKATAN AKURASI PADA KLASIFIKASI LESI KULIT MANUSIA  
DENGAN *DATA BALANCING* MENGGUNAKAN PENDEKATAN  
*CONVOLUTIONAL NEURAL NETWORK* (CNN)**

**SKRIPSI**

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah  
Sarjana Ilmu Komputer**

**PUAN ABIDAH NITISARA  
201401009**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

## PERSETUJUAN

Judul : PENINGKATAN AKURASI PADA  
KLASIFIKASI LESI KULIT MANUSIA  
DENGAN *DATA BALANCING*  
MENGUNAKAN PENDEKATAN  
*CONVOLUTIONAL NEURAL NETWORK (CNN)*

Kategori : SKRIPSI

Nama : PUAN ABIDAH NITISARA

Nomor Induk Mahasiswa : 201401009

Program Studi : SARJANA (S1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI  
INFORMASI UNIVERSITAS SUMATERA  
UTARA

Telah diuji dan dinyatakan lulus di Medan, 15 Oktober 2024

Pembimbing I



Dr. Amalia ST., M.T.

NIP. 197812212014042001

Pembimbing II

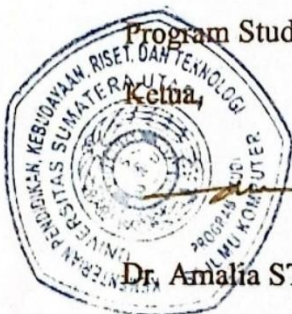


Fanindia Purnamasari S.TI., M.IT

NIP. 198908172019032023

Diketahui/disetujui oleh

Program Studi S1 Ilmu Komputer



Dr. Amalia ST., M.T.

NIP. 197812212014042001

**PERNYATAAN**

**PENINGKATAN AKURASI PADA KLASIFIKASI LESI KULIT MANUSIA  
DENGAN *DATA BALANCING* MENGGUNAKAN PENDEKATAN  
*CONVOLUTIONAL NEURAL NETWORK* (CNN)**

**SKRIPSI**

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 6 Agustus 2024



Puan Abidah Nitisara

201401009

## **PENGHARGAAN**

Segala puji bagi Allah Azza Wa Jalla yang telah melimpahkan keberkahan, rahmat, taufik, inayah, dan hidayahNya kepada penulis sehingga mampu menyelesaikan penyusunan skripsi ini sebagai syarat untuk memperoleh gelar Sarjana Komputer, pada Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. Shalawat berangkaikan salam kepada baginda Rasulullah, Nabi Muhammad Shallallahu ‘alaihi Wasallam, karena perantara beliau-lah kita bisa merasakan nikmat ilmu pengetahuan yang diridho Allah.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada:

1. Bapak Dr. Muryanto Amin, S.Sos, M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia ST., M.T. selaku Ketua Program Studi S1 Ilmu Komputer Universitas Sumatera Utara serta selaku dosen pembimbing I yang telah memberikan bimbingan, kritik, motivasi, dan saran serta arahan menuju kebaikan kepada penulis dalam menyelesaikan skripsi ini.
4. Ibu Fanindia Purnamasari S.TI., M.IT selaku dosen pembimbing II yang telah memberikan bimbingan, kritik, motivasi, dan saran kepada penulis dalam menyelesaikan skripsi ini.
5. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer yang telah memberikan waktu dan tenaga untuk mengajar dan membimbing sehingga penulis dapat sampai kepada tahap penyusunan skripsi ini.
6. Seluruh Staf Pegawai Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara yang telah banyak memberikan bantuan kepada penulis selama masa perkuliahan sampai kepada tahap penyusunan skripsi ini.

7. Penulis berterima kasih pada diri sendiri karena telah bertahan, tekun, dan tidak menyerah dalam mengerjakan skripsi. Tanpa semangat dari dalam diri, skripsi ini mungkin tidak akan selesai. Sekali lagi, terima kasih kepada diriku tersayang.
8. Orang tua penulis yang telah memberikan motivasi, dukungan, doa, dan kasih sayang penuh kepada penulis dalam menyelesaikan pendidikan.
9. Kakak kandung tercinta Galuh Atika Nabila, S.Kom. yang selalu memberikan motivasi, dukungan, saran, kasih sayang, serta doa kepada penulis dalam menjalankan aktivitas perkuliahan hingga akhirnya menyelesaikan tugas akhir.
10. Teman-teman perkuliahan Zelda, Rizky, Rezha, Ayu, Ainun, Sonia, Dhila, Zulfa, Lia, Sitti, Sawaliyah, Syaripa, dan Siti Jubaidah yang telah menemani dan membantu penulis selama belajar di perkuliahan serta memberikan semangat untuk menyelesaikan tugas akhir.
11. Teman-teman seluruh keluarga besar angkatan 2020 serta Abang dan Kakak senior Ilmu Komputer Universitas Sumatera Utara yang telah banyak memberi motivasi kepada penulis.
12. Dan semua pihak yang telah membantu dan menyemangati yang tidak dapat disebutkan satu per satu.

Semoga rahmat Allah selalu menemani langkah serta diberikan keberkahan kepada semua pihak yang telah memberikan dukungan berupa motivasi, arahan dan tindakan serta dukungan kepada penulis dalam menyelesaikan skripsi ini. Semoga skripsi ini bermanfaat bagi pribadi, keluarga, masyarakat, organisasi dan negara.

Medan, 6 Agustus 2024



Puan Abidah Nitisara

201401009

## ABSTRAK

Lesi kulit adalah area pada kulit yang memiliki penampilan atau pertumbuhan abnormal, yang dapat dikategorikan sebagai jinak atau ganas. Diagnosa awal lesi kulit sangat penting untuk menentukan perawatan yang tepat, namun klasifikasi lesi kulit secara visual memiliki tantangan karena kemiripan visual antar lesi. Meskipun dermatologis menggunakan dermatoskop untuk memeriksa lesi, diagnosis sering kali bersifat subjektif dan dipengaruhi oleh berbagai faktor, seperti kualitas gambar dan kondisi fisik pemeriksa. Oleh karena itu, pengembangan sistem diagnosis otomatis berbasis komputer diperlukan untuk meningkatkan konsistensi dan akurasi hasil. Penelitian ini mengembangkan sistem klasifikasi lesi kulit menggunakan metode *deep learning* dengan arsitektur DenseNet121, yang diterapkan pada dataset HAM10000. Masalah utama dalam penelitian ini adalah ketidakseimbangan jumlah data pada masing-masing kelas dalam dataset. Untuk mengatasi masalah tersebut, penelitian dilakukan pada dataset asli maupun pada dataset yang diterapkan teknik *data balancing* berupa SMOTE (*Synthetic Minority Oversampling Technique*) dan Augmentasi Data, guna mengatasi ketidakseimbangan data. Hasil penelitian menunjukkan akurasi terbaik pada dataset yang diterapkan teknik SMOTE dengan akurasi pelatihan sebesar 96.48% dan *loss* pelatihan sebesar 0.101, dengan akurasi validasi sebesar 94.01% dan *loss* validasi sebesar 0.2463, serta akurasi pengujian sebesar 93.20% dengan *loss* pengujian sebesar 0.2341. Model ini kemudian diintegrasikan ke dalam aplikasi berbasis *website* untuk memudahkan pengguna mendeteksi lesi kulit jenis apa berdasarkan gambar *dermoscopy* yang dimasukkan.

**Kata Kunci:** Lesi kulit, *DenseNet121*, *Data Balancing*, *SMOTE*, Augmentasi Data, Aplikasi Berbasis Website.

## ABSTRACT

Skin lesion is a part of the skin with an abnormal appearance or growth, which can be categorized as benign or malignant. Early diagnosis of skin lesions is essential to determine appropriate treatment, but visual classification of skin lesions is challenging due to visual similarities between lesions. Although dermatologists use a dermatoscope to examine lesions, diagnosis is often subjective and influenced by various factors, such as image quality and the physical condition of the dermatologist. Therefore, the development of a computer-based automated diagnosis system is necessary to improve the consistency and accuracy of results. This research develops a skin lesion classification system using deep learning method with DenseNet121 architecture, which is applied to HAM10000 dataset. The main problem in this research is the imbalance in the amount of data in each class in the dataset. To address this issue, research was conducted on the original dataset as well as on datasets where data balancing techniques, such as SMOTE (Synthetic Minority Oversampling Technique) and Data Augmentation, were applied to mitigate data imbalance. The results indicated the highest accuracy on the dataset that applied the SMOTE technique, with a training accuracy of 96.48% and a training loss of 0.101, a validation accuracy of 94.01% and a validation loss of 0.2463, and a testing accuracy of 93.20% with a testing loss of 0.2341. This model is then integrated into a web-based application to facilitate user detection of the type of skin lesion based on the entered dermoscopy image.

**Keywords:** Skin lesion, DenseNet121, Data Balancing, SMOTE, Data Augmentation, Web-based Application.



## DAFTAR ISI

<b>PERSETUJUAN .....</b>	<b>iii</b>
<b>PERNYATAAN.....</b>	<b>iv</b>
<b>PENGHARGAAN.....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vii</b>
<b>ABSTRACT .....</b>	<b>viii</b>
<b>DAFTAR ISI.....</b>	<b>ix</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>BAB 1 .....</b>	<b>1</b>
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	5
1.3. Batasan Masalah.....	6
1.4. Tujuan Penelitian.....	7
1.5. Manfaat Penelitian.....	7
1.6. Metodologi Penelitian .....	7
1.7. Sistematika Penulisan.....	8
<b>BAB 2 .....</b>	<b>10</b>
2.1. Lesi Kulit ( <i>Skin Lesion</i> ).....	10
2.2. <i>Deep Learning</i> .....	13
2.2.1. <i>Convolutional Neural Network (CNN)</i> .....	14
2.3. <i>Data Balancing</i> .....	16
2.3.1. <i>Synthetic Minority Oversampling Technique (SMOTE)</i> .....	17
2.3.2. Augmentasi Data .....	18
2.4. Metrik Evaluasi .....	19
2.4.1. <i>Confusion Matrix</i> .....	19

2.5. Penelitian yang Relevan .....	21
<b>BAB 3 .....</b>	<b>23</b>
3.1. Gambaran Umum Sistem .....	23
3.2. Pengambilan Dataset .....	25
3.3. Pra-pemrosesan Data .....	25
3.3.1. <i>Data Balancing</i> .....	25
3.4. Membangun Model .....	28
3.5. Pelatihan Model.....	29
3.6. Evaluasi dan Pengujian Model .....	29
3.7. Perancangan Antarmuka Sistem Berbasis <i>Website</i> .....	30
<b>BAB 4 .....</b>	<b>33</b>
4.1. Implementasi Sistem .....	33
4.1.1. Perangkat Keras .....	33
4.1.2. Perangkat Lunak .....	33
4.2. Implementasi Tahap Pengambilan Dataset .....	34
4.3. Implementasi Tahap Pra-pemrosesan Data .....	34
4.3.1. Transformasi Gambar ke Array NumPy, Resize dan Normalization .....	34
4.3.2. Label Encoding .....	35
4.3.3. Data Splitting .....	36
4.3.4. Data Balancing.....	36
4.3.5. <i>Train-Val-Test Splitting</i> .....	37
4.4. Implementasi Model.....	38
4.5. Implementasi Tahap Pelatihan Model.....	39
4.6. Implementasi Tahap Evaluasi dan Pengujian Model .....	41
4.7. Implementasi Model ke Aplikasi Berbasis Website.....	49
4.8. Pengujian Sistem .....	52
<b>BAB 5 .....</b>	<b>56</b>
5.1. Kesimpulan.....	56
5.2. Saran.....	57
<b>DAFTAR PUSTAKA.....</b>	<b>58</b>
<b>LAMPIRAN.....</b>	<b>A-1</b>

## DAFTAR TABEL

<b>Tabel 1.1</b> Dataset HAM10000 .....	3
<b>Tabel 2.1</b> Arsitektur DenseNet-121 .....	15
<b>Tabel 4.1</b> Hasil Teknik <i>Label Encoding</i> .....	35
<b>Tabel 4.2</b> Perbandingan Data Asli dengan Data yang Diterapkan <i>Data Balancing</i> ....	36
<b>Tabel 4.3</b> Distribusi Data untuk <i>Train-Val-Test Split</i> .....	38
<b>Tabel 4.4</b> Hasil Pelatihan Model DenseNet121 dengan Dataset Asli .....	40
<b>Tabel 4.5</b> Hasil Pelatihan Model DenseNet121 + SMOTE .....	40
<b>Tabel 4.6</b> Hasil Pelatihan Model DenseNet121 + Augmentasi Data .....	41
<b>Tabel 4.7</b> Perbandingan Metrik Akurasi dan <i>Loss</i> pada <i>Epoch</i> Terakhir Pelatihan....	42
<b>Tabel 4.8</b> Perbandingan Metrik Evaluasi menggunakan Data Pengujian .....	49
<b>Tabel 4.9</b> <i>Data Testing</i> yang Menjadi Input Ketika Pengujian Sistem .....	52

## DAFTAR GAMBAR

<b>Gambar 2.1</b> <i>Dermoscopy</i> Basal Cell Carcinoma .....	10
<b>Gambar 2.2</b> <i>Dermoscopy</i> Actinic keratoses .....	11
<b>Gambar 2.3</b> <i>Dermoscopy</i> Melanoma .....	11
<b>Gambar 2.4</b> <i>Dermoscopy</i> Benign keratoses-like lesions .....	12
<b>Gambar 2.5</b> <i>Dermoscopy</i> Melanocytic nevi .....	12
<b>Gambar 2.6</b> <i>Dermoscopy</i> Dermatofibroma .....	13
<b>Gambar 2.7</b> <i>Dermoscopy</i> Vascular lesions .....	13
<b>Gambar 2.8</b> Arsitektur CNN .....	14
<b>Gambar 2.9</b> Teknik <i>Resampling</i> .....	17
<b>Gambar 2.10</b> Ilustrasi dari SMOTE <i>Oversampling</i> .....	18
<b>Gambar 2.11</b> <i>Confusion Matrices: (a) Binary Classification Confusion Matrix, (b) Multiclass Classification Confusion Matrix</i> .....	20
<b>Gambar 3.1</b> Gambaran Umum Sistem .....	24
<b>Gambar 3.2</b> Arsitektur Umum Sistem .....	24
<b>Gambar 3.3</b> Gambar Desain <i>Mockup</i> Halaman Website .....	31
<b>Gambar 4.1</b> Memuat Dataset Gambar Lesi Kulit .....	34
<b>Gambar 4.2</b> Transformasi Gambar ke Array NumPy, Resize, dan Normalization ....	35
<b>Gambar 4.3</b> Arsitektur DenseNet121 yang Digunakan dalam Penelitian .....	39
<b>Gambar 4.4</b> Grafik Akurasi dan Loss Function pada Pelatihan dan Validasi Model.	43
<b>Gambar 4.5</b> Confusion Matrix Model DenseNet121 dengan Dataset Asli .....	44
<b>Gambar 4.6</b> Classification Report DenseNet121 dengan Dataset Asli .....	45
<b>Gambar 4.7</b> Confusion Matrix Model DenseNet121 + Augmentasi Data .....	46
<b>Gambar 4.8</b> Classification Report Model DenseNet121 + Augmentasi Data .....	46
<b>Gambar 4.9</b> Confusion Matrix Model DenseNet121 + SMOTE .....	47
<b>Gambar 4.10</b> Classification Report Model DenseNet121 + SMOTE .....	48
<b>Gambar 4.11</b> Tampilan Beranda .....	50
<b>Gambar 4.12</b> Tampilan About Skin Lesions .....	50

<b>Gambar 4.13</b> Tampilan Tipe Skin Lesions.....	51
<b>Gambar 4.14</b> Tampilan Deteksi Skin Lesion .....	51
<b>Gambar 4.15</b> Tampilan Hasil Deteksi Skin Lesion .....	52
<b>Gambar 4.16</b> Pengujian gambar <i>dermoscopy</i> dari RS Sanglah, Bali pada sistem klasifikasi berbasis <i>website</i> .....	55

# **BAB 1**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Lesi kulit (*skin lesion*) adalah bagian dari kulit yang memiliki pertumbuhan atau penampilan yang tidak normal dibandingkan dengan kulit di sekitarnya. Ini dapat berupa ruam, tahi lalat, kutil, kista, lepuh, atau area kulit lain yang tidak normal. Lesi kulit dapat bersifat jinak (non-kanker) atau ganas (kanker). Diagnosa awal terhadap lesi kulit ini cukup penting agar dapat mengetahui perawatan yang tepat bagi pasien. Mengklasifikasikan dan mengidentifikasi lesi kulit menjadi tugas yang sulit bahkan bagi dermatologis yang sudah berpengalaman, karena adanya kemiripan visual antar lesi kulit (Ghalejoogh et al., 2020).

Dermatologis atau pakar kulit menggunakan metode pemeriksaan dermatoskopis yang terkomputerisasi untuk memeriksa gambar lesi kulit pada tubuh manusia dan memutuskan apakah lesi tersebut mencurigakan dengan meneliti gambar-gambar tersebut. Dermatoskop memungkinkan pemeriksaan lebih dekat pada warna, pola, dan struktur permukaan kulit, membantu dalam identifikasi karakteristik khas dari berbagai jenis lesi kulit. Maka dari itu, citra yang dihasilkan oleh dermatoskopi dapat membantu dermatologis dalam melakukan diagnosis dan pengambilan keputusan (Hatem, 2022).

Dalam praktiknya, diagnosis oleh dermatologis dapat bersifat subjektif. Berbagai faktor, seperti resolusi gambar yang rendah, pengalaman, kelelahan, dan gangguan visual dari pemeriksa, dapat mempengaruhi hal ini, sehingga menyebabkan variasi dalam diagnosis dan hasil yang tidak konsisten di antara para dermatologis (Alhudhaif et al., 2023). Dengan demikian, pengembangan sistem diagnosis otomatis berbasis komputer dapat membantu mengatasi kerentanan dan ketidakpastian yang mungkin terjadi dalam diagnosis lesi kulit, sehingga meningkatkan konsistensi dan akurasi hasil (Wang et al., 2022).

Oleh karena itu, diperlukan sebuah sistem yang mampu mengklasifikasikan lesi kulit sebagai langkah awal untuk mendeteksi dini lesi kulit yang mencurigakan. Dengan memberikan diagnosis yang cepat, aplikasi ini dapat memberikan kesadaran bagi pengguna untuk segera mencari perawatan yang tepat kepada dermatologis. Selain itu, sistem juga dapat memberikan informasi terkait lesi kulit dan penanganan yang tepat sebagai sarana edukasi bagi pengguna. Sistem tersebut menggunakan teknologi *machine learning* dalam melakukan klasifikasi.

Dengan kemajuan teknologi kecerdasan buatan (*Artificial Intelligence*) dan pembelajaran mesin (*Machine Learning*), analisis citra medis dan diagnosis penyakit menjadi lebih mungkin dilakukan. Dalam beberapa tahun terakhir, masalah klasifikasi lesi kulit telah menjadi fokus penelitian, terutama dengan menggunakan metode deep learning (Alhudhaif et al., 2023). Penggunaan data berkualitas tinggi pun menjadi penting untuk mendapatkan klasifikasi lesi kulit yang akurat (Rahman et al., 2021).

Di sisi lain, masalah utama dalam penelitian ini yakni ketidakseimbangan atau *imbalance* data pada masing-masing kelas dalam dataset penelitian. Ketika dataset tidak seimbang, satu kelas memiliki jumlah *instance* yang mayoritas, sementara kelas lainnya memiliki jumlah *instance* yang minoritas (Swana et al., 2022). Berdasarkan beberapa penelitian yang dilaksanakan oleh (Eftekheri et al., 2017; Qian et al., 2014; Yakshit et al., 2022) mendapati bahwa model yang menggunakan *imbalance* data dapat menurunkan akurasi klasifikasi.

Berdasarkan permasalahan tersebut, perlu dilakukan penyeimbangan data atau *data balancing* agar dapat meningkatkan akurasi dalam pengklasifikasian. Penelitian ini membuat tiga model *machine learning* pada dataset yang sama. Model pertama tidak menggunakan teknik *data balancing* pada dataset sedangkan model kedua dan ketiga menggunakan teknik *data balancing* dengan metode yang berbeda pada dataset. Setelah itu, dilakukan evaluasi performa model untuk melihat peningkatan akurasi model *machine learning* yang didapatkan dari ketiga model tersebut.

Pada studi ini, algoritma *Convolutional Neural Network* (CNN), khususnya arsitektur *DenseNet121*, digunakan pada sistem pendeteksi lesi kulit sebagai solusi untuk klasifikasi lesi kulit. *Data balancing* diterapkan menggunakan metode

*Synthetic Minority Oversampling Technique* (SMOTE) dan Augmentasi Data guna menangani permasalahan ketidakseimbangan data. Sebagai hasil akhir, sistem akan diimplementasikan pada aplikasi berbasis *website*. Selain itu, dalam studi ini menggunakan dataset HAM10000 ("*Human Against Machine with 10000 training images*") yang teridentifikasi sebagai *imbalance* data. Dataset dapat diakses secara publik melalui repositori Harvard Dataverse berikut:

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>.

Dataset tersebut berisi 10015 *dermatoscopic/dermoscopy images*, yang merupakan gambar kulit yang diambil dengan menggunakan alat dermatoskop, yaitu alat yang dapat melihat struktur permukaan kulit dengan optimal, seperti pola, warna, dan batas lesi. Alat ini memiliki lensa pembesar yang digabungkan dengan sumber cahaya, yang memungkinkan visualisasi struktur kulit yang lebih dalam yang tidak terlihat dengan mata telanjang. Dermatoskop biasanya tidak mengubah warna alami kulit. Warna kulit tetap sama seperti aslinya di gambar yang dihasilkan, tetapi pencahayaan dari dermatoskop mungkin membuat detail kulit terlihat lebih kontras, terutama dalam hal perbedaan warna pada pigmen atau struktur kulit yang tidak biasa. Pada beberapa kasus, pencahayaan bisa membuat bagian kulit terlihat sedikit lebih cerah atau lebih gelap tergantung pada pengaturan cahaya dan teknik yang digunakan. Namun, secara umum, warna dasar kulit tetap konsisten. Untuk dataset penelitian terdiri dari 7 jenis lesi kulit yang berbeda. Detail data per kelas tercantum pada Tabel 1.1 berikut:

**Tabel 1.1** Dataset HAM10000

<b>Nama Kelas</b>	<b>Jumlah Sampel dalam Kelas</b>
Actinic keratoses (akiec)	327
Basal cell carcinoma (bcc)	514
Benign keratoses-like lesions (bkl)	1099
Dermatofibroma (df)	115
Melanoma (mel)	1113



Melanocytic nevi (nv)	6705
Vascular lesions (vasc)	142

---

Dapat dilihat pada Tabel 1.1, bahwa distribusi setiap kelas dalam dataset tersebut memiliki jumlah data yang tidak merata atau *imbalance*, sehingga dikhawatirkan klasifikasi kelas menjadi tidak akurat. Dengan adanya distribusi kelas yang tidak merata, sistem dapat menghasilkan kesalahan klasifikasi pada sejumlah *instance* minoritas karena sistem klasifikasi cenderung bersifat bias dan lebih mendukung *instance* mayoritas. Hal tersebut membuat sistem klasifikasi cenderung mengabaikan kelas minoritas dan mendeteksinya sebagai *noise* (Napierala & Stefanowski, 2016).

Dalam penyeimbangan data, *resampling* adalah salah satu metode yang dapat diterapkan. *Resampling* bertujuan agar pemerataan jumlah data per kelas dapat dicapai, yaitu dengan mengurangi data kelas mayoritas, yang dikenal sebagai *undersampling*, maupun meningkatkan data kelas minoritas, yang dikenal sebagai *oversampling*. Teknik *undersampling* dilakukan dengan mengeliminasi secara acak data kelas mayoritas (Hasib et al., 2020). Namun, *undersampling* dapat menghilangkan data yang penting atau signifikan.

Di sisi lain, teknik *oversampling* dilakukan dengan menggandakan secara acak data pada kelas minoritas hingga menciptakan distribusi kelas yang seimbang (Hasib et al., 2020). Keuntungan dari teknik tersebut adalah tidak ada data yang dihilangkan, dan memberikan performa yang lebih efektif dibandingkan dengan teknik *undersampling*. Namun, ada potensi risiko *overfitting* karena data yang direplikasi.

Guna mengatasi permasalahan tersebut, *Synthetic Minority Oversampling Technique* (SMOTE), ialah teknik *oversampling* sintetis yang dikembangkan oleh (Chawla et al., 2011) sebagai perbaikan terhadap *oversampling* konvensional, dapat diterapkan. Pendekatan *k-nearest neighbor* dimanfaatkan demi menciptakan sampel buatan yang baru berdasarkan proporsi tertentu pada kelas minoritas dalam ruang fitur. SMOTE mampu menciptakan data sintetis baru tanpa

menggandakannya dan menangani kemungkinan *overfitting* yang mungkin muncul dalam *oversampling* biasa.

Penelitian oleh (Alhudhaif et al., 2023) telah mengaplikasikan *Synthetic Minority Oversampling Technique* (SMOTE), berhasil meningkatkan akurasi hingga 95%, yang dibandingkan dengan akurasi sebelum melakukan SMOTE didapatkan sebesar 70%. Adapun penelitian dari (Varalakshmi et al., 2021) berhasil mendapatkan akurasi yang lebih baik sebesar 96% dengan menggunakan SMOTE dibandingkan dengan model yang lain.

Selain itu, teknik *data balancing* lainnya adalah Augmentasi Data. Augmentasi Data adalah proses mengaplikasikan transformasi atau modifikasi pada data asli untuk menghasilkan versi baru dari data tersebut. Teknik ini membantu dalam meningkatkan performa model pembelajaran mesin dengan memperluas dan memperkaya dataset pelatihan (Yang et al., 2023). Teknik ini sangat berguna dalam pembelajaran mesin dan deep learning, terutama ketika bekerja dengan dataset gambar. Dengan teknik augmentasi berupa rotasi, translasi, skalasi, pemotongan acak, pembalikan (*flipping*), perubahan kecerahan, kontras, dan warna (*brightness, contrast, and color adjustment*) dapat menyediakan variasi yang lebih luas pada data pelatihan serta mengatasi ketidakseimbangan data (Yang et al., 2023).

Dalam riset yang dilakukan oleh (Gu et al., 2019) berhasil meningkatkan tes akurasi hingga 92% pada dataset CIFAR-10, yang dibandingkan dengan tes akurasi sebelum melakukan Augmentasi Data didapatkan sebesar 86%. Pada penelitian yang dilakukan oleh (O'gara & Mcguinness, 2019) dapat meningkatkan akurasi sebesar +2.83% menjadi 95.85% menggunakan model ResNet pada dataset CIFAR-10. Di samping itu, riset yang dilaksanakan oleh (Liu et al., 2021) mendapatkan hasil terbaik dalam mengklasifikasikan *portrait paintings* dengan ResNet50 dan Augmentasi Data sebesar 90%, yang dibandingkan dengan akurasi sebelum melakukan Augmentasi Data sebesar 80%.

## 1.2. Rumusan Masalah

Peningkatan akurasi pada model *machine learning* untuk klasifikasi dapat diperoleh dengan menjamin bahwa tiap kelas dalam dataset mempunyai jumlah data yang terdistribusi secara merata atau *balance*. Pada studi ini memanfaatkan

pendekatan *Convolutional Neural Network* (CNN), khususnya arsitektur *DenseNet121* pada dataset berikut:

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T> menunjukkan ketidakseimbangan data atau *imbalance data*. Karenanya, dalam studi ini perlu diterapkan penyeimbangan data menggunakan metode *Synthetic Minority Oversampling Technique* (SMOTE) dan Augmentasi Data. Sebagai hasil akhir, sistem akan diimplementasikan pada aplikasi berbasis *website* sebagai sarana pendeteksi dini lesi kulit dan edukasi.

### 1.3. Batasan Masalah

Pembatasan masalah pada studi ini mencakup:

1. Model dibangun dengan mengaplikasikan pendekatan *Convolutional Neural Network* (CNN) yaitu arsitektur *DenseNet121*.
2. Pelatihan model yang pertama dilakukan pada dataset orisinal yang tidak seimbang, lalu pelatihan model kedua dan ketiga dilakukan pada dataset yang akan diseimbangkan dengan *data balancing* menggunakan *Synthetic Minority Oversampling Technique* (SMOTE) dan Augmentasi Data. Setelah itu, dilakukan evaluasi untuk melihat peningkatan akurasi yang didapatkan dari ketiga model tersebut.
3. Menggunakan dataset HAM10000, yang berisi 10015 *dermoscopy images*, di mana sampel tersebut didapatkan dari departemen dermatologi di Medical University of Vienna, Austria dan Skin Cancer Practice of Cliff Rosendahl in Queensland, Australia. Dataset tersebut dapat diakses secara publik melalui repositori Harvard Dataverse berikut:  
<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>
4. Sampel *dermoscopy image* yang digunakan dataset HAM10000 berasal dari berbagai populasi yang berbeda di Eropa dan Australia. Untuk kebutuhan visual, pakar dermatologis melakukan koreksi histogram manual untuk meningkatkan kontras visual dan reproduksi warna pada beberapa gambar yang terlalu gelap (*underexposed*) dan beberapa gambar yang terlihat memiliki rona (*hue*) kuning atau hijau.

5. Klasifikasi lesi kulit terdiri dari 7 kelas yaitu *Actinic keratoses*, *Basal cell carcinoma*, *Benign keratoses-like lesions*, *Dermatofibroma*, *Melanoma*, *Melanocytic nevi*, dan *Vascular lesions*.
6. Sistem akan diimplementasikan pada aplikasi berbasis *website*. Aplikasi *website* ini menggunakan *framework* ReactJS untuk *frontend* dan FastAPI untuk *backend*.

#### 1.4. Tujuan Penelitian

Studi ini memiliki beberapa tujuan meliputi pengembangan aplikasi berbasis *website* yang efektif dan akurat sehingga dapat mengklasifikasikan lesi kulit yang mencurigakan sebagai bentuk deteksi dini yang mudah diakses, serta dapat memberikan edukasi yang bermanfaat terkait lesi kulit. Selain itu, fokus utama penelitian mencakup peningkatan akurasi model *deep learning* dengan menerapkan teknik *data balancing* yang menjadi esensial dalam mengatasi ketidakseimbangan jumlah sampel antar kelas dalam dataset.

#### 1.5. Manfaat Penelitian

Aplikasi yang dikembangkan diharapkan dapat berperan sebagai pendeteksi lesi kulit serta sebagai sarana edukasi untuk pengguna. Dengan demikian, aplikasi ini diharapkan dapat memberikan kesadaran kepada penderita untuk segera mencari perawatan yang tepat kepada dermatologis. Selain daripada itu, studi ini memberikan gambaran tentang pengaplikasian *Convolutional Neural Network* (CNN), khususnya arsitektur *DenseNet121*, serta teknik *data balancing* menggunakan *Synthetic Minority Oversampling Technique* (SMOTE) dan Augmentasi Data.

#### 1.6. Metodologi Penelitian

Metode penelitian dilaksanakan pada studi ini mencakup:

##### 1. Studi Pustaka

Tahap ini melibatkan pencarian literatur dari sumber kredibel serta penelaahan *e-book*, studi yang relevan, dan berbagai penelitian yang telah dipublikasikan dalam jurnal terkait lesi kulit, *Deep Learning*, *Convolutional Neural Network*

(CNN), *DenseNet121*, *data balancing*, *Synthetic Minority Oversampling Technique* (SMOTE), Augmentasi Data, dan metrik evaluasi.

## 2. Analisis dan Perancangan

Tahap ini meliputi analisis kebutuhan penelitian, baik dari aspek teori maupun desain sistem, termasuk arsitektur sistem secara umum dan diagram alir (*flowchart*).

## 3. Implementasi Sistem

Tahap ini mencakup pengembangan model yang dikembangkan dengan bahasa *Python*. Model diimplementasikan ke dalam aplikasi berbasis *website* dengan menggunakan *framework* ReactJS untuk *frontend* dan FastAPI untuk *backend*.

## 4. Pengujian Sistem

Pada langkah ini, sistem diuji untuk memastikan bahwa program berfungsi sebagaimana yang diharapkan dan sesuai dengan kebutuhan.

## 5. Dokumentasi Sistem

Pada fase ini, semua tahap dari analisis hingga pengujian sistem didokumentasikan dalam skripsi untuk menunjukkan pencapaian yang diperoleh.

### 1.7. Sistematika Penulisan

Sistematika penulisan dari skripsi ini terdiri dari lima bab, yakni:

#### **BAB I      PENDAHULUAN**

Semua aspek relevan terkait latar belakang permasalahan, seperti rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metode penelitian, dan sistematika penulisan, dibahas dalam bab ini.

#### **BAB II     LANDASAN TEORI**

Tinjauan teoritis yang berkaitan dengan lesi kulit, Deep Learning, *Convolutional Neural Network* (CNN), *DenseNet121*, *data balancing*, *Synthetic Minority Oversampling Technique* (SMOTE), Augmentasi Data, dan metrik evaluasi dibahas pada bab ini.

### **BAB III ANALISIS DAN PERANCANGAN**

Pembahasan mengenai masalah yang dianalisis serta pengembangan sistem, diikuti dengan tahap perancangan menggunakan *Convolutional Neural Network* (CNN), khususnya arsitektur *DenseNet121*, serta penerapan teknik *data balancing* melalui metode *Synthetic Minority Oversampling Technique* (SMOTE) dan Augmentasi Data, tercakup pada bab ini.

### **BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM**

Proses implementasi dan pengujian sistem berdasarkan tahapan analisis dan perancangan, diuraikan dalam bab ini.

### **BAB V KESIMPULAN DAN SARAN**

Kesimpulan dan saran dari hasil studi yang sudah dilaksanakan tercakup dalam bagian ini.

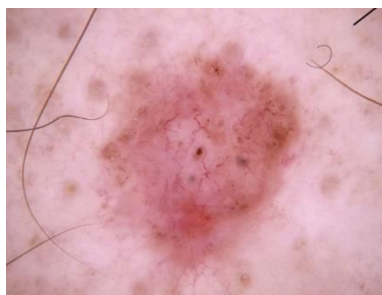
## BAB 2

### LANDASAN TEORI

#### 2.1. Lesi Kulit (*Skin Lesion*)

Lesi kulit (*skin lesion*) merupakan area pada kulit yang menunjukkan pertumbuhan atau tampilan yang tidak normal dari kulit di sekitarnya. Lesi kulit dapat bersifat jinak (non-kanker) atau ganas (kanker). *Skin cancer* atau kanker kulit muncul akibat pertumbuhan sel kulit yang tidak normal. Faktor utama yang menyebabkan kanker kulit ialah paparan sinar matahari berbahaya, yang dapat meningkatkan risiko seseorang mengalami kanker kulit (Alhudhaif et al., 2023). Ini dapat muncul di berbagai bagian tubuh, tetapi lebih umum terjadi di bagian kepala, leher, wajah, dan tangan yang paling terpengaruh oleh sinar matahari yang merugikan. Beberapa jenis yang termasuk lesi kulit ganas (kanker) adalah sebagai berikut:

- *Basal Cell Carcinoma* (bcc) merupakan jenis kanker kulit non-melanoma yang paling sering ditemukan, juga dikenal sebagai *keratinocyte cancer*, dan bersifat lokal invasif, yang berarti bahwa biasanya tumbuh secara perlahan dan terbatas pada area kulit di sekitarnya (Bichakjian et al., 2018).



**Gambar 2.1** *Dermoscopy Basal Cell Carcinoma*

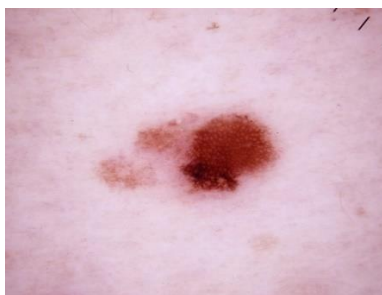
- *Actinic keratoses* (akiec) adalah bercak bersisik pra-kanker yang muncul pada kulit yang mengalami kerusakan akibat terpapar sinar matahari, dikenal juga sebagai solar keratosis, serta potensi dari lesi ini untuk berkembang menjadi kanker ganas yang invasif dianggap rendah (De

Berker et al., 2007). Ini mungkin dianggap sebagai bentuk awal dari *squamous cell carcinoma (keratinocyte cancer)*. *Actinic keratoses* menyerang orang yang sering tinggal di daerah tropis atau subtropis yang sering terpapar sinar matahari tanpa perlindungan dalam jangka waktu yang lama, ataupun yang memiliki kekebalan tubuh yang kurang baik.



**Gambar 2.2** *Dermoscopy Actinic keratoses*

- *Melanoma* (mel), disebut juga sebagai melanoma ganas, adalah kanker kulit yang berpotensi sangat serius di mana terjadi pertumbuhan tak terkendali dari sel melanosit (sel pigmen) (Marsden et al., 2010). Melanosit yang biasa ditemukan di lapisan basal dari epidermis (lapisan luar kulit), memproduksi protein yang disebut melanin, yang melindungi sel kulit dengan menyerap radiasi ultraviolet (UV). Pertumbuhan non-kanker dari melanosit menghasilkan tahi lalat (*melanocytic nevi* jinak) dan *freckles (ephelides dan lentigines)*. Sebaliknya, pertumbuhan kanker dari melanosit menghasilkan melanoma.



**Gambar 2.3** *Dermoscopy Melanoma*

Lesi kulit yang bersifat jinak (non-kanker) ini merupakan pertumbuhan abnormal pada kulit. Sebagian besar lesi kulit yang bersifat jinak tidak berbahaya dan tidak memerlukan pengobatan kecuali jika lesi tersebut menyebabkan ketidaknyamanan ataupun mengurangi estetika dari penampilan seseorang. Beberapa jenis yang termasuk lesi kulit jinak adalah sebagai berikut:

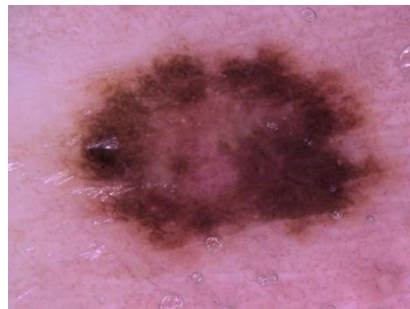


- *Benign keratosis-like lesions* (bkl) merupakan kelas lesi kulit yang bersifat jinak di mana pada dataset HAM10000 itu mencakup beberapa penyakit seperti *seborrheic keratosis*, *solar lentigo*, dan *lichen-planus like keratosis*. Ketiga kelompok ini mungkin tampak berbeda secara dermatoskopis, tetapi dalam dataset tersebut mereka dikelompokkan bersama karena memiliki kesamaan biologis dan secara histopatologis (Tschandl et al., 2018).



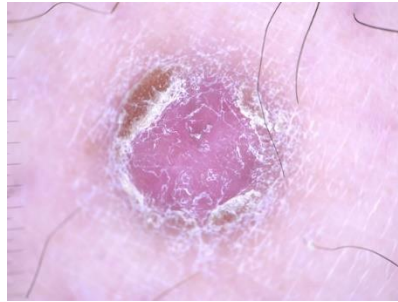
**Gambar 2.4** Dermoscopy Benign keratosis-like lesions

- *Melanocytic nevi* (nv), biasa disebut tanda lahir atau tahi lalat, ini merupakan lesi kulit jinak yang umum disebabkan oleh proliferasi lokal sel pigmen (*melanocytic*). Berwarna coklat atau hitam yang mengandung pigmen melanin.



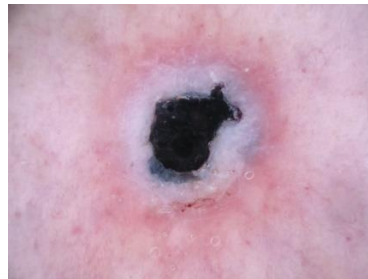
**Gambar 2.5** Dermoscopy Melanocytic nevi

- *Dermatofibroma* (df), berasal dari sel *fibrous*, merupakan lesi jinak yang muncul sebagai satu atau beberapa papula dermal yang keras, serta berwarna merah muda, kuning kecoklatan, atau coklat (Luba et al., 2003). *Dermatofibroma* sebagian besar ditemui pada orang dewasa, biasa lebih umum terjadi pada wanita daripada pria, meskipun beberapa varian histologis lebih sering diidentifikasi pada pria.



**Gambar 2.6** *Dermoscopy Dermatofibroma*

- *Vascular lesions* (vasc) merupakan lesi kulit jinak, lebih dikenal sebagai tanda lahir. Ada tiga kategori utama *vascular lesions* yaitu sebagai berikut: *Angioma/Hemangioma*, *Pyogenic Granuloma*, *Vascular Malformations*.



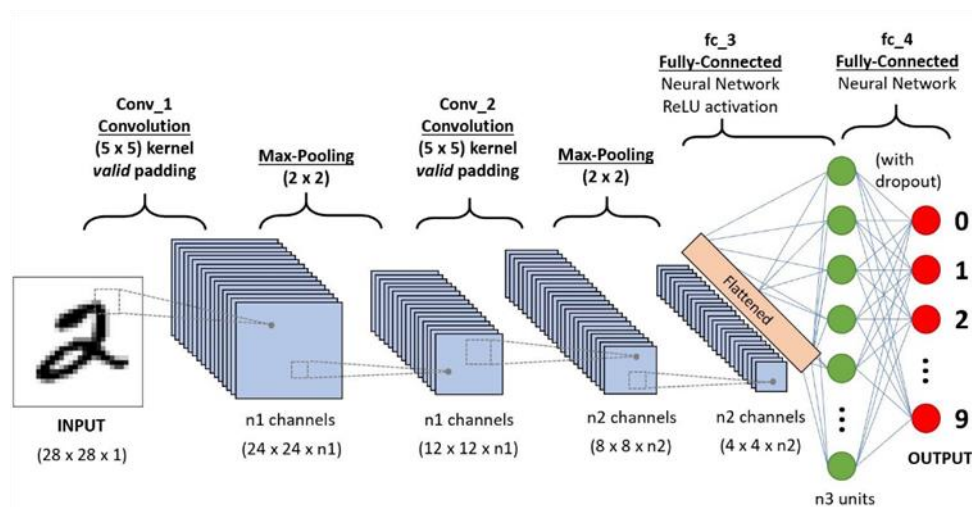
**Gambar 2.7** *Dermoscopy Vascular lesions*

## 2.2. Deep Learning

*Deep Learning* merupakan cabang dari *machine learning* dan *artificial intelligence* dengan jaringan saraf tiruan terinspirasi oleh cara kerja neuron pada otak manusia (Vargas et al., 2017). *Deep learning* terdiri dari beberapa lapisan unit pemrosesan non-linear untuk ekstraksi dan transformasi fitur (Zhang et al., 2018). Dengan begitu, *deep learning* mampu melakukan pembelajaran fitur secara otomatis, tanpa perlu keterlibatan manusia secara intensif, sehingga dapat menangani data dalam jumlah besar dan dapat mengidentifikasi pola yang rumit dalam data (Janiesch et al., 2021). Proses pembelajaran fitur umumnya berlangsung secara hierarkis, di mana fitur-fitur abstrak tingkat tinggi dibangun melalui kombinasi fitur-fitur sederhana pada tingkat yang lebih rendah. Setiap lapisan yang ada pada *deep learning* berturut-turut menggunakan keluaran dari lapisan sebelumnya sebagai masukan.

### 2.2.1. Convolutional Neural Network (CNN)

*Convolutional Neural Network (CNN)* adalah model *deep learning* yang dirancang untuk mengolah data berstruktur, seperti gambar atau data lain yang memiliki dimensi spasial. CNN dirancang untuk secara otomatis belajar hierarki fitur spasial dari data sehingga dapat mengidentifikasi pola-pola dasar pada tingkat rendah dan secara bertahap membangun pemahaman yang lebih kompleks pada tingkat yang lebih tinggi (Yamashita et al., 2018). Arsitektur CNN terdiri dari tiga jenis lapisan utama: lapisan konvolusi, *pooling*, dan *fully connected*.



Gambar 2.8 Arsitektur CNN

#### 2.2.1.1. DenseNet121

DenseNet121 adalah salah satu varian dari arsitektur jaringan saraf *Convolutional Neural Network (CNN)* yang dikenal sebagai salah satu variasi dari arsitektur DenseNet (*Densely Connected Convolutional Networks*) yang diperkenalkan oleh (Huang et al., 2016). Arsitektur ini dirancang untuk meningkatkan efisiensi dan akurasi dalam tugas klasifikasi gambar dengan mengutamakan konektivitas yang padat antara lapisan-lapisan jaringan saraf. setiap lapisan dalam jaringan menerima input dari semua lapisan sebelumnya. Alih-alih menjumlahkan *feature map* dari lapisan sebelumnya, DenseNet menggabungkan (*concatenate*) semua *feature map* tersebut sebagai input ke lapisan berikutnya. Ini membantu mengurangi jumlah parameter dan memungkinkan penggunaan ulang fitur.

DenseNet121 terdiri dari 121 lapisan yang tersusun dalam beberapa komponen utama, yaitu:

- *Initial Convolution Layer*: lapisan ini bertugas melakukan pemrosesan awal terhadap gambar input untuk mengekstraksi fitur dasar sebelum gambar tersebut diteruskan ke lapisan-lapisan berikutnya dalam jaringan.
- *Dense Blocks*: komponen utama dalam arsitektur DenseNet yang memungkinkan konektivitas padat antara lapisan-lapisan konvolusi. Dengan menggabungkan fitur dari semua lapisan sebelumnya, *Dense Blocks* meningkatkan efisiensi parameter, penggunaan ulang fitur, dan mengatasi masalah vanishing gradient, yang semuanya berkontribusi pada peningkatan performa model.
- *Transition Layers*: layer ini ditempatkan di antara *dense blocks* untuk mengurangi dimensi peta fitur dan jumlah kanal. Transition layer terdiri dari *Batch Normalization* (BN), ReLU, *1x1 Convolution*, dan *2x2 Average Pooling*.
- *Final Layers*: Setelah dense block terakhir, ada *Batch Normalization* (BN), ReLU, *Global Average Pooling*, dan *Fully Connected (Dense) Layer* dengan jumlah neuron sesuai dengan jumlah kelas yang ingin diklasifikasikan, diikuti oleh fungsi aktivasinya.

Arsitektur DenseNet121 digambarkan melalui Tabel 2.1 berikut:

**Tabel 2.1** Arsitektur DenseNet-121

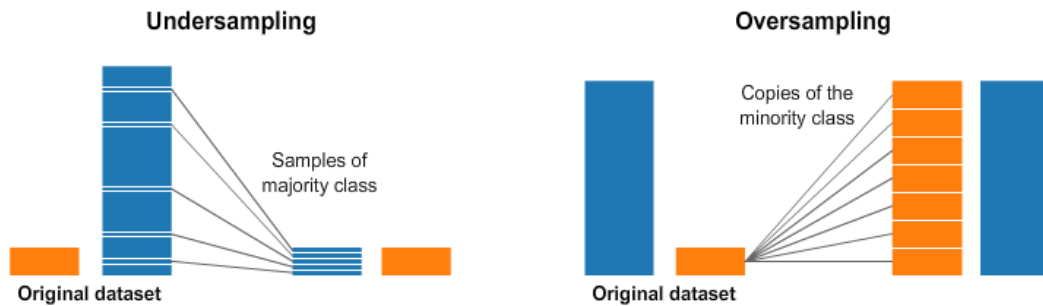
Layers	Output Size	DenseNet-121
<i>Convolution</i>	112 x 112	7 x 7 conv
<i>Pooling</i>	56 x 56	3 x 3 max pool
<i>Dense Blocks 1</i>	56 x 56	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 6$
<i>Transition Layer 1</i>	56 x 56	1 x 1 conv
	28 x 28	2 x 2 average pool
<i>Dense Blocks 2</i>	28 x 28	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 12$

<i>Transition Layer 2</i>	28 x 28	1 x 1 conv
	14 x 14	2 x 2 average pool
<i>Dense Blocks 3</i>	14 x 14	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 24$
<i>Transition Layer 3</i>	14 x 14	1 x 1 conv
	7 x 7	2 x 2 average
<i>Dense Blocks 4</i>	7 x 7	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix} \times 16$
<i>Classification Layer</i>	1 x 1	7 x 7 global average pool
		1000D fully-connected, softmax

### 2.3. Data Balancing

*Data balancing* merujuk pada proses penyesuaian distribusi data dalam suatu dataset untuk mengatasi ketidakseimbangan. Ketidakseimbangan ini terjadi ketika satu kelas memiliki jumlah sampel yang jauh lebih banyak dibandingkan dengan kelas-kelas lainnya, yang mana hal ini dapat menyebabkan model *machine learning* menjadi bias terhadap kelas mayoritas (Jadhav et al., 2022). Karenanya, menyeimbangkan dataset yang tidak seimbang sangat penting untuk mengurangi bias dalam prediksi model dan meningkatkan akurasi performa model *classifier* (Jadhav et al., 2022).

Untuk mengatasi masalah *imbalance* dataset, salah dua solusinya adalah *resampling* dan Augmentasi Data. *Resampling* umumnya dilakukan pada tahap *preprocessing* data, yang bertujuan untuk menyamakan jumlah data per kelas. Teknik *Resampling* memungkinkan penyeimbangan dataset pada tingkat *imbalance ratio* (IR) yang diinginkan. Teknik *resampling* secara garis besar meliputi:



**Gambar 2.9** Teknik Resampling

- Undersampling, teknik yang mengurangi jumlah sampel pada kelas mayoritas dengan cara menghapus sampel secara acak dari kelas tersebut (Hasib et al., 2020). Namun, kelemahan dari metode ini adalah kemungkinan dapat menghilangkan beberapa informasi yang berpotensi penting untuk proses pembelajaran model.
- Oversampling, meningkatkan sampel kelas minoritas dengan menggandakan secara acak sampel yang ada pada kelas minoritas hingga menciptakan distribusi kelas yang seimbang (Hasib et al., 2020). Keuntungan dari teknik tersebut adalah tidak ada data yang dihilangkan. Namun, ada potensi risiko overfitting karena data yang direplikasi.

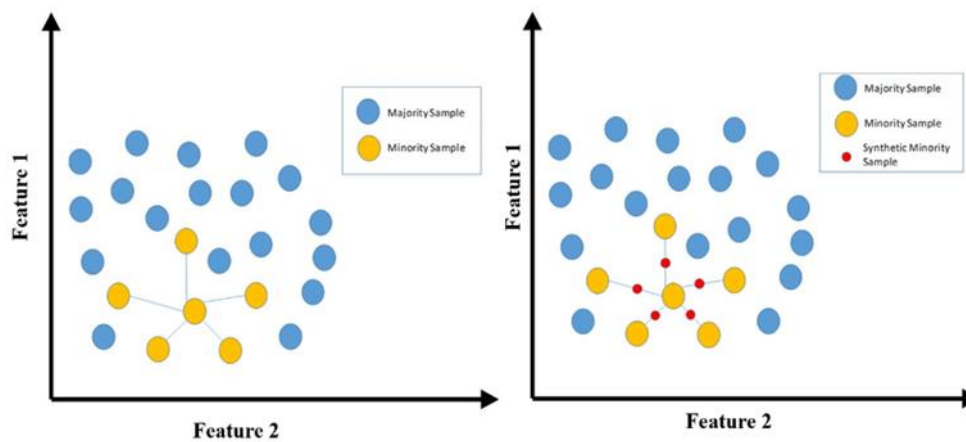
### 2.3.1. Synthetic Minority Oversampling Technique (SMOTE)

*Synthetic Minority Oversampling Technique* (SMOTE) ialah teknik oversampling sintesis yang dikembangkan oleh Chawla (Chawla et al., 2011) sebagai perbaikan terhadap *oversampling* konvensional. SMOTE tidak menggandakan sampel kelas minoritas, namun sampel-sampel baru dibuat secara sintesis dalam *feature space* yang dibentuk oleh instance dan *K-nearest neighbors* kelas minoritas seperti yang ditunjukkan pada gambar di bawah. (Chawla et al., 2011). Langkah-langkah dari teknik oversampling SMOTE adalah sebagai berikut:

- 1) Pertama, SMOTE memilih sampel acak dari kelas minoritas ( $X \in S_{min}$ ).
- 2) Selanjutnya, untuk sampel acak minoritas terpilih, SMOTE mencari beberapa tetangga terdekatnya menggunakan metode *K-Nearest Neighbors* (KNN).

- 3) Pilih salah satu tetangga terdekatnya (*nearest neighbor*) yang berasal dari sampel kelas minoritas juga ( $X_{nn} \in S_{min}$ ).
- 4) Buat sampel sintetis baru di antara dua titik yang berasal dari sampel acak kelas minoritas ( $X \in S_{min}$ ) dan satu tetangga terdekatnya ( $X_{nn} \in S_{min}$ ) berdasarkan interpolasi linear. Sampel baru ini tidak identik dengan sampel asli, tetapi berada di ruang yang sama.
- 5) Sampel sintetis baru dihasilkan berdasarkan:  

$$X_{new} = X + |X_{nn} - X| \times \lambda$$
 dimana  $\lambda \in [0,1]$  adalah variabel acak.



**Gambar 2.10** Ilustrasi dari SMOTE Oversampling

### 2.3.2. Augmentasi Data

Selain *resampling*, Augmentasi Data juga dapat mengatasi masalah ketidakseimbangan data, yang merupakan proses penerapan transformasi atau modifikasi pada data asli untuk menciptakan versi baru dari data tersebut. Teknik ini berperan dalam meningkatkan kinerja model pembelajaran mesin dengan memperluas dan memperkaya dataset pelatihan (Yang et al., 2023). Dengan teknik tersebut dapat menciptakan berbagai modifikasi dari gambar asli, sehingga membantu model menjadi lebih tahan terhadap variasi dan gangguan dalam data *input*. Beberapa contoh teknik Augmentasi Data:

- Rotasi (Rotation): gambar diputar pada berbagai sudut untuk menciptakan variasi orientasi. Contoh: rotasi 90 derajat, 180 derajat, 270 derajat.

- Translasi (Translation): gambar digeser secara horizontal atau vertikal. Contoh: geser ke kanan atau ke bawah.
- Skalasi (Scaling): gambar diperbesar atau diperkecil. Contoh: *zoom in* atau *zoom out*.
- Pemotongan Acak (Random Cropping): memotong bagian acak dari gambar asli untuk menciptakan variasi. Contoh: memotong area 224x224 dari gambar asli 256x256.
- Pembalikan (Flipping): membalik gambar secara horizontal atau vertikal. Contoh: pembalikan horizontal (*mirror flip*).
- Perubahan Kecerahan, Kontras, dan Warna (*Brightness, Contrast, dan Color Adjustment*): mengubah tingkat kecerahan, kontras, atau warna gambar. Contoh: meningkatkan kecerahan gambar sebesar 20%.

## 2.4. Metrik Evaluasi

Metrik evaluasi adalah ukuran kuantitatif yang digunakan untuk menilai kinerja dan efektivitas model machine learning. Metrik evaluasi memberikan wawasan tentang seberapa baik kinerja model, dapat membantu dalam membandingkan kinerja dua atau lebih model berbeda, dan dapat menganalisis perilaku model yang sama dengan menyetel parameter berbeda (Grandini et al., 2020). Ada beberapa metrik yang didasarkan pada *Confusion Matrix*, karena matriks ini memuat semua informasi relevan tentang performa model.

### 2.4.1. Confusion Matrix

*Confusion matrix* merupakan tabel yang digunakan untuk mengevaluasi performansi model klasifikasi. Tabel ini menampilkan jumlah prediksi yang benar dan salah dari model dibandingkan dengan hasil sebenarnya. Matriks ini sering dimanfaatkan guna menghitung metrik seperti *accuracy*, *precision*, *recall*, serta *F1 score*.



		Predicted Class	
		A	B
True Class	A	TP	FN
	B	FP	TN

(a)

		Predicted Class		
		A	B	C
True Class	A	TP	FN	FN
	B	FP	TN	FN
	C	FP	FN	TN

(b)

**Gambar 2.11** *Confusion Matrices: (a) Binary Classification Confusion Matrix, (b) Multiclass Classification Confusion Matrix*

- TP (*True Positive*), ini terjadi ketika model memprediksi hasil positif, dan prediksi tersebut benar karena hasil yang sebenarnya juga positif.
- TN (*True Negative*), ini terjadi ketika model memprediksi hasil negatif, dan prediksi tersebut benar karena hasil sebenarnya juga negatif.
- FP (*False Positive*), ini terjadi ketika model memprediksi hasil positif, tetapi hasil sebenarnya adalah negatif, sehingga prediksi model salah. Ini dikenal sebagai *Type 1 Error*.
- FN (*False Negative*), ini terjadi ketika model memprediksi hasil negatif, tetapi hasil sebenarnya adalah positif, sehingga prediksi model salah. Ini dikenal juga sebagai *Type 2 Error*.

Beberapa metrik didasari dari *confusion matrix* adalah sebagai berikut:

- 1) *Accuracy* menilai seberapa baik model klasifikasi dalam melakukan prediksi yang benar. Secara sederhana, akurasi menghitung rasio prediksi yang benar terhadap total jumlah prediksi.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- 2) *Precision* menilai seberapa akurat model dalam memprediksi kelas positif. *Precision* dihitung dengan membagi jumlah prediksi positif yang benar dengan total jumlah prediksi positif yang dilakukan oleh model.

$$Presicion = \frac{TP}{TP + FP} \quad (2)$$

- 3) *Recall*, atau dikenal juga sebagai *Sensitivity* atau *True Positive Rate*, mengukur kemampuan model dalam mendeteksi semua *instance* yang benar-benar termasuk dalam kelas positif. *Recall* dihitung dengan membagi jumlah prediksi positif yang benar dengan total jumlah *instance* yang sebenarnya positif.

$$Recall (Sensitivity) = \frac{TP}{TP + FN} \quad (3)$$

- 4) *F1-Score* menggabungkan precision dan recall guna menyediakan ukuran seimbang antara kemampuan model dalam mengenali kasus positif dan meminimalkan *false positive* dan *false negative*. Hal ini menjadikannya metrik yang cocok sangat berguna ketika memiliki kelas yang tidak seimbang (*imbalance*) dan ingin mencari keseimbangan antara precision dan recall.

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall} \quad (4)$$

## 2.5. Penelitian yang Relevan

Studi-studi berikut ini relevan dengan studi yang dilakukan, yakni:

1. Pada studi berjudul *Enhanced Dermatoscopic Skin Lesion Classification Using Machine Learning Techniques* (Varalakshmi et al., 2021) menggunakan model SVM with Polynomial Kernel dengan teknik *data balancing* SMOTE, berhasil mencapai akurasi yang lebih tinggi sebesar 96% dibandingkan dengan model *machine learning* lain.
2. Pada penelian yang berjudul *Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks* (Joloudari et al., 2023) menunjukkan bahwa SMOTE-Normalization-CNN telah mengungguli beberapa model yang berbeda dengan mencapai nilai akurasi sebesar 99,08% pada 24 dataset yang tidak seimbang.

3. Pada penelitian yang berjudul *A Novel Nonlinear Automated Multi-Class Skin Lesion Detection System Using Soft-Attention Based Convolutional Neural Networks* (Alhudhaif et al., 2023) menggunakan Soft-Attention Based CNN dengan beberapa teknik *data balancing* dan mendapatkan hasil yang paling baik yaitu menggunakan teknik SMOTE yang berhasil meningkatkan nilai akurasi hingga 95%, yang dibandingkan dengan nilai akurasi sebelum melakukan *data balancing* didapatkan sebesar 70%.
4. Pada penelitian yang berjudul *Novel Features For Art Movement Classification of Portrait Paintings* (Liu et al., 2021) mendapatkan performa terbaik dalam mengklasifikasikan *portrait paintings* dengan ResNet50 dan Augmentasi Data sebesar 90%, yang dibandingkan dengan akurasi sebelum melakukan Augmentasi Data sebesar 80%.
5. Pada penelitian yang berjudul *Improve Image Classification Using Data Augmentation and Neural Networks* (Gu et al., 2019) berhasil meningkatkan tes akurasi hingga 92% pada dataset CIFAR-10, yang dibandingkan dengan tes akurasi sebelum melakukan Augmentasi Data didapatkan sebesar 86%.

## BAB 3

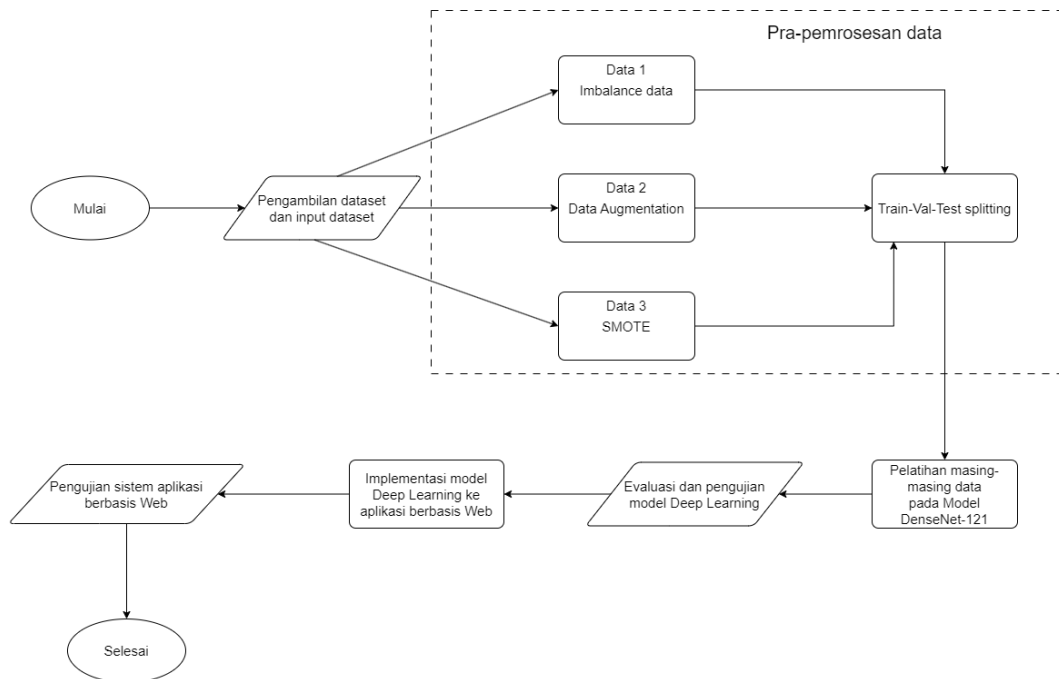
### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1. Gambaran Umum Sistem

Dalam studi ini, dikembangkan sistem klasifikasi lesi kulit yang mampu mengidentifikasi penyakit lesi kulit berbasis aplikasi *web*. Perancangan sistem ini dilakukan dengan menggunakan pendekatan DenseNet-121 dan teknik *data balancing* untuk mengatasi masalah ketidakseimbangan data antar kelas. Sistem ini dimulai dengan proses pengambilan dataset gambar lesi kulit. Dataset yang digunakan adalah dataset *Skin Cancer MNIST: HAM10000* yang dapat diakses pada <https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000> yang dikumpulkan dari Harvard Dataverse Repository. Dataset tersebut mengalami tahap pra-pemrosesan seperti *data splitting*, normalisasi, *label encoding*, dan *data balancing*, sebelum akhirnya digunakan untuk pelatihan model *deep learning*. Data balancing yang digunakan adalah Augmentasi Data dan SMOTE (*Synthetic Minority Over-sampling Technique*).

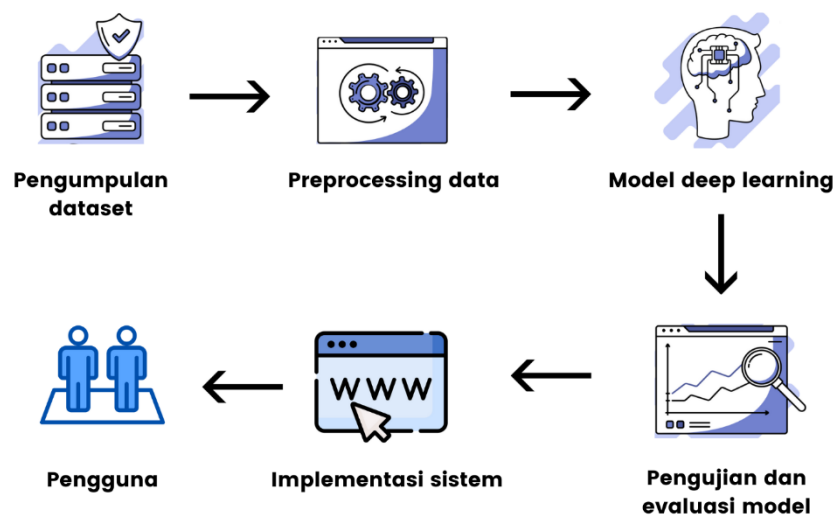
Sistem klasifikasi lesi kulit melatih tiga dataset untuk mengevaluasi model dengan akurasi terbaik. Dataset pertama tidak dilakukan penyeimbangan data sehingga data asli masih *imbalance*. Dataset kedua dan ketiga dilakukan penyeimbangan data dengan Augmentasi Data dan SMOTE. Ketiga dataset tersebut dilatih pada model *deep learning* yang sama yaitu model DenseNet-121. Setelah pelatihan model, dilakukan evaluasi terhadap model dengan metrik akurasi, loss function, presicion, recall, F1-score, dan confusion matrix untuk melihat performa dari hasil pelatihan ketiga model. Kemudian, model dengan akurasi terbaik disimpan dalam sebuah *file* untuk diimplementasikan pada aplikasi berbasis *web*. Selanjutnya, aplikasi berbasis web dikaji ulang guna mengevaluasi apakah sudah mencapai tujuan, yakni mampu mengklasifikasikan gambar *dermoscopy* lesi kulit

yang diunggah pengguna dengan akurat. Gambar 3.1 menunjukkan alur proses pengembangan sistem yang diterapkan dalam penelitian ini:



**Gambar 3.1** Alur Proses Pengembangan Sistem

Sementara itu, Gambar 3.2 menunjukkan arsitektur umum sistem yang digunakan dalam penelitian ini.



**Gambar 3.2** Arsitektur Umum Sistem

Proses yang mencakup rancangan arsitektur umum pada Gambar 3.2 meliputi:

1. Dataset yang dikumpulkan adalah dataset HAM10000 ("*Human Against Machine with 10000 training images*") yang berupa *dermoscopic image* dengan tujuh kelas lesi kulit berbeda.
2. Setelah itu, dilakukan *preprocessing data* seperti *normalization*, *train-val-test splitting*, *data balancing*, dan sebagainya.
3. Selanjutnya, data akan dilatih pada model *deep learning* dengan pendekatan *DenseNet-121*.
4. Kemudian dilakukan pengujian dengan *data testing* dan evaluasi performa model dengan berbagai metrik evaluasi.
5. Setelah model menghasilkan akurasi yang bagus, sistem akan diimplementasikan pada aplikasi berbasis *website*.
6. Hasil akhir berupa aplikasi *website* dapat digunakan oleh pengguna.

### 3.2. Pengambilan Dataset

Pada tahap awal penelitian ini adalah melakukan pengambilan dataset penelitian, yang berasal dari dataset *Skin Cancer MNIST: HAM10000* yang dapat diakses pada <https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000> yang dikumpulkan dari Harvard Dataverse Repository. Dataset tersebut berisi 10015 *dermoscopy images*, yang terdiri dari 7 jenis lesi kulit yang berbeda. Detail jumlah data per kelas dapat ditemukan pada Tabel 1.1.

### 3.3. Pra-pemrosesan Data

Selanjutnya adalah melakukan pra-pemrosesan data dengan tujuan agar model yang dibangun dapat memahami dan mempelajari data dengan optimal. Beberapa langkah yang dilakukan dalam penelitian pada tahap pra-pemrosesan data meliputi transformasi gambar ke *array NumPy*, *resize* dan *normalization*, *label encoding*, *data splitting*, *data balancing*, serta *train-val-test splitting*.

#### 3.3.1. Data Balancing

Sebagaimana terlihat pada Tabel 1.1, dataset penelitian tersebut menunjukkan adanya ketidakseimbangan jumlah data antar kelas. Oleh karena itu, diperlukan langkah penyeimbangan data untuk setiap kelas. Dalam penelitian ini, teknik

SMOTE dan Augmentasi Data diterapkan untuk mengatasi masalah tersebut. SMOTE (*Synthetic Minority Over-sampling Technique*) adalah teknik yang diterapkan untuk mengatasi ketidakseimbangan data pada klasifikasi. Berikut langkah-langkah bagaimana SMOTE dilakukan:

- 1) Identifikasi Kelas Minoritas

Langkah pertama dalam SMOTE adalah mengidentifikasi kelas minoritas dalam dataset. Kelas minoritas adalah kelas yang memiliki jumlah sampel jauh lebih sedikit dibandingkan dengan kelas mayoritas.

- 2) Pemilihan Sampel dari Kelas Minoritas

Setelah mengidentifikasi kelas minoritas, SMOTE akan memilih secara acak sampel-sampel dari kelas ini. Misalnya, jika ada 100 sampel dari kelas minoritas, SMOTE dapat memilih beberapa dari mereka secara acak untuk menciptakan data baru.

- 3) Menentukan Tetangga Terdekat (*k-Nearest Neighbors*)

Bagi setiap sampel yang dipilih, SMOTE mencari beberapa tetangga terdekat (*nearest neighbors*) dalam ruang fitur dari kelas minoritas yang sama. Jumlah tetangga terdekat yang digunakan ditentukan oleh parameter *k\_neighbors*. Sebagai contoh, jika *k\_neighbors* adalah 5, SMOTE akan menemukan 5 tetangga terdekat untuk setiap sampel yang dipilih.

- 4) Pembentukan Sampel Sintetik

Setelah tetangga terdekat ditentukan, SMOTE kemudian membuat sampel baru (sampel sintetik) dengan menginterpolasi antara sampel asli yang dipilih dan salah satu tetangga terdekatnya. Proses ini dilakukan dengan mengambil perbedaan antara sampel asli dan tetangganya, mengalikannya dengan nilai acak antara 0 dan 1, dan menambahkannya ke sampel asli. Misalnya, jika ada dua titik data,  $x_1$  dan  $x_2$ , yang merupakan tetangga terdekat, SMOTE akan membuat sampel sintetik baru di antara  $x_1$  dan  $x_2$  dengan interpolasi linear.

- 5) Penambahan Sampel Sintetik ke Dataset

Sampel sintetis yang dihasilkan ditambahkan ke dataset asli, sehingga memperbanyak sampel di kelas minoritas. Langkah ini mendukung keseimbangan model *machine learning* selama fase pelatihan.

#### 6) Pengaturan Parameter

SMOTE memiliki beberapa parameter penting, termasuk *k\_neighbors* dan *sampling\_strategy*. *k\_neighbors* menentukan jumlah tetangga terdekat yang dipertimbangkan untuk membuat sampel sintetis. Sementara itu, *sampling\_strategy* menentukan proporsi yang ingin dicapai antara kelas mayoritas dan minoritas. Pada penelitian ini, pemilihan parameter untuk SMOTE mengikuti bawaan dari pustaka yaitu *k\_neighbors* sebesar 5, yang berarti menggunakan 5 tetangga terdekat, dan *sampling\_strategy*='auto' yang berarti SMOTE akan meningkatkan jumlah sampel di kelas minoritas hingga seimbang dengan kelas mayoritas. Penggunaan parameter bawaan dipilih karena hasil pengujian pada penelitian ini telah cukup efektif dalam meningkatkan akurasi hasil.

#### 7) Evaluasi Performa

Setelah menerapkan SMOTE, dataset yang seimbang dapat digunakan untuk melatih model, dan performanya dievaluasi kembali.

Selain itu, Augmentasi Data merupakan teknik yang digunakan untuk meningkatkan jumlah sampel dalam dataset, terutama untuk kelas minoritas, dengan membuat variasi dari data yang ada. Ini sangat berguna dalam mengatasi masalah *imbalanced data* karena dapat menghasilkan data baru yang serupa tetapi berbeda dari data asli, sehingga meningkatkan representasi kelas minoritas. Berikut ialah langkah-langkah yang diterapkan dalam proses Augmentasi Data:

##### 1) Memilih Data Untuk Dilakukan Augmentasi Data

Langkah pertama adalah memilih sampel data dari kelas minoritas atau data yang kurang terwakili dalam dataset. Dalam kasus ketidakseimbangan kelas, augmentasi biasanya diterapkan pada kelas minoritas untuk meningkatkan jumlah sampelnya.

##### 2) Menentukan Teknik Augmentasi



Ada banyak teknik augmentasi yang bisa digunakan tergantung pada jenis data. Teknik augmentasi untuk gambar termasuk perubahan kontras, perubahan kecerahan, pembalikan, dan sebagainya.

### 3) Penerapan Teknik Augmentasi

Setelah teknik augmentasi dipilih, teknik tersebut diterapkan pada data yang dipilih. Misalnya, dalam augmentasi gambar, menggunakan membalik gambar secara horizontal, atau mengubah kecerahan. Ini menghasilkan versi baru dari data asli. Dalam pengolahan gambar, *library TensorFlow* digunakan untuk melakukan augmentasi secara otomatis selama pelatihan model.

### 4) Menghasilkan Data Baru

Setiap aplikasi teknik augmentasi menghasilkan data baru yang sedikit berbeda dari data asli. Misalnya, gambar yang diputar atau dibalik akan tetap memiliki label kelas yang sama tetapi tampilannya akan berbeda.

### 5) Menggabungkan Data yang Di-augmentasi ke Dataset Asli

Data yang dihasilkan melalui augmentasi kemudian digabungkan dengan dataset asli, menciptakan dataset yang lebih besar dan lebih beragam. Ini membantu meningkatkan keseimbangan antara kelas minoritas dan mayoritas.

## 3.4. Membangun Model

Setelah tahap pra-pemrosesan selesai, selanjutnya adalah membangun model menggunakan metode deep learning dengan DenseNet-121. Dalam penelitian ini, model dibuat menggunakan framework TensorFlow dan Keras. Model DenseNet-121 terdiri dari “121 lapisan” dimana secara umum terdiri dari *initial convolution layer*, *lapisan dense blocks*, *transition layers*, *final layers*. DenseNet121 adalah model *pretrained* sudah dilatih sebelumnya dengan menggunakan dataset besar seperti ImageNet. DenseNet121 dapat berfungsi sebagai dasar (backbone) untuk berbagai tugas *computer vision*, dengan memanfaatkan bobot yang telah dilatih sebelumnya.

Dengan memuat model DenseNet121 yang sudah dilatih pada dataset ImageNet, langkah selanjutnya adalah menghapus *layer fully connected* terakhir

dari model dasar untuk kemudian menambahkan *layer* baru sesuai dengan jumlah kelas pada dataset yang baru. *Layer fully connected* tambahan biasanya diakhiri dengan fungsi softmax untuk mendukung klasifikasi multikelas. Selain itu, penting untuk mempertimbangkan pembekuan (*freezing*) *layer* awal dari model dasar guna menjaga agar bobotnya tidak terpengaruh selama pelatihan awal.

### 3.5. Pelatihan Model

Setelah model selesai dibangun, selanjutnya menggunakan data pelatihan (*train set*) dan data validasi (*validation set*) untuk melatih model DenseNet121. Model tersebut dilatih pada masing-masing dataset, yaitu dataset yang pertama belum seimbang, dataset kedua diseimbangkan dengan metode Augmentasi Data, dan dataset ketiga diseimbangkan dengan metode SMOTE. Model dilatih maksimal sebanyak 60 *epochs* dengan penerapan *callbacks* berupa *EarlyStopping* dan *ReduceLROnPlateau*. Pemilihan 60 epoch dilakukan untuk memberikan cukup iterasi agar model dapat mempelajari pola dari dataset, namun tetap membatasi risiko *overfitting*. Jumlah epoch ini juga mempertimbangkan keseimbangan antara waktu pelatihan dan performa model, dengan *EarlyStopping* yang secara otomatis menghentikan pelatihan jika tidak ada peningkatan lebih lanjut.

*Epoch* adalah satu siklus lengkap di mana seluruh dataset pelatihan diproses satu kali oleh model yang sedang dilatih. Sedangkan *Early Stopping* adalah teknik yang digunakan dalam pelatihan model untuk menghentikan proses pelatihan lebih awal jika tidak ada peningkatan yang signifikan dalam metrik evaluasi (misalnya, akurasi atau loss) pada dataset validasi. Selain itu, *ReduceLROnPlateau* adalah teknik untuk mengurangi learning rate (tingkat pembelajaran) model secara otomatis saat kemajuan pelatihan mengalami kemacetan, seperti saat loss pada dataset validasi tidak menurun lagi.

### 3.6. Evaluasi dan Pengujian Model

Setelah proses pelatihan selesai dilakukan, selanjutnya dilakukan proses evaluasi dan pengujian model. Penelitian ini bertujuan meningkatkan akurasi model dengan mengatasi permasalahan ketidakseimbangan data antar kelas. Untuk itu, dilakukan tiga eksperimen berbeda: (1) membangun model tanpa teknik balancing

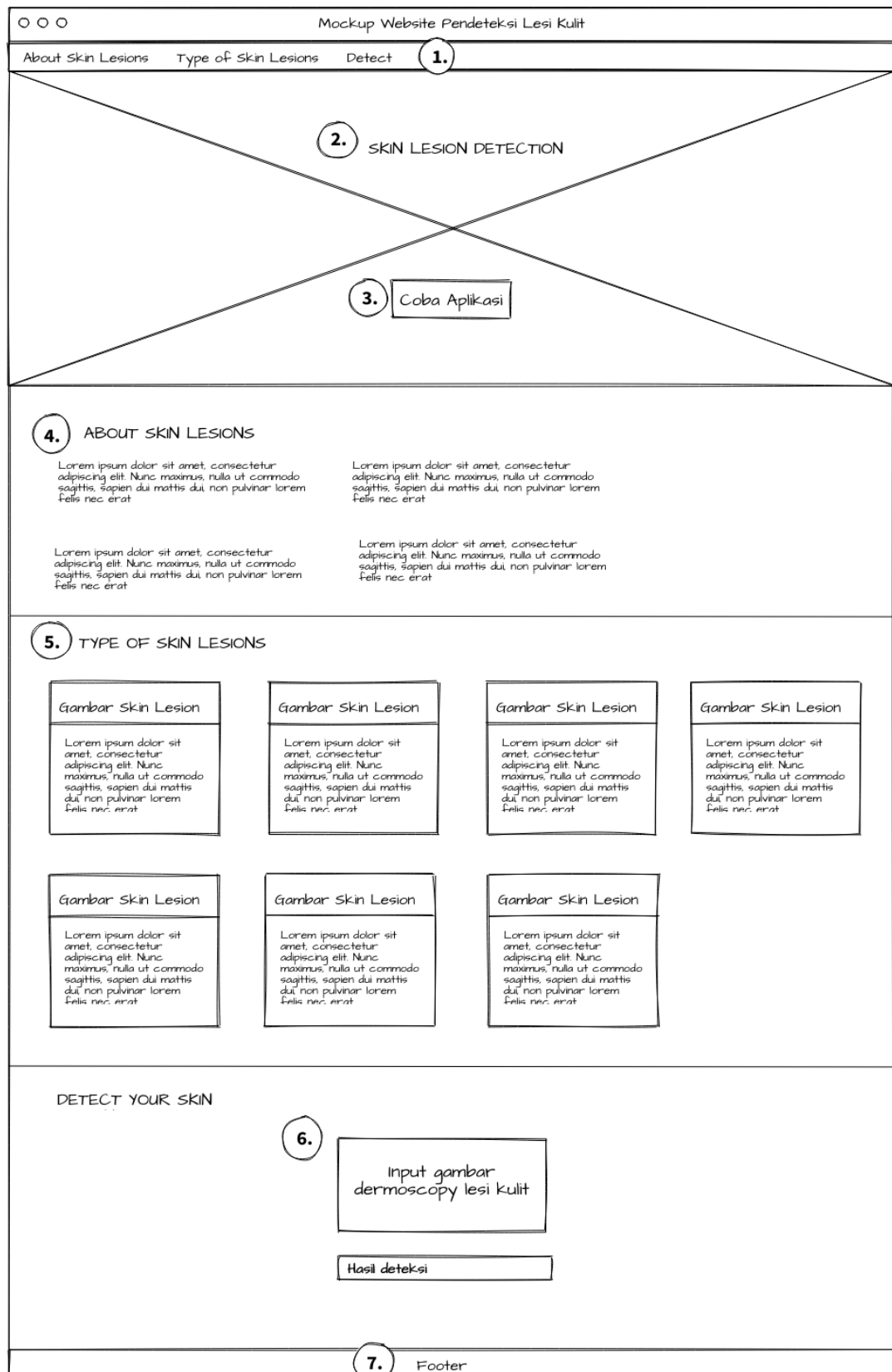
data, (2) menggunakan SMOTE untuk menyeimbangkan data, dan (3) menggunakan augmentasi data untuk menyeimbangkan data.

Kemudian, evaluasi dilakukan secara kontinu selama proses pelatihan menggunakan data pelatihan (*train set*) dan data validasi (*validation set*), dengan menggunakan metrik akurasi dan *sparse categorical crossentropy loss function*. *Sparse categorical crossentropy loss function* digunakan terutama ketika memiliki tugas klasifikasi yang memiliki lebih dari dua kelas (*multiclass classification*) di mana label kelasnya diwakili dalam bentuk bilangan bulat (*sparse labels*).

Pada tahap pengujian model dilakukan pada data uji (*test set*) dengan cara memprediksi data uji yang belum pernah dilihat oleh model ketika proses pelatihan tersebut. Untuk evaluasi data uji, metrik yang diterapkan mencakup akurasi, *precision*, *recall*, dan *F1-score*, serta *confusion matrix*. Gambaran *confusion matrix* dapat dilihat pada Gambar 2.11.

### **3.7. Perancangan Antarmuka Sistem Berbasis Website**

Antarmuka pengguna (UI) pada sistem berbasis website dirancang untuk memberikan gambaran yang jelas tentang desain dan pengalaman pengguna dari aplikasi sistem pendeteksi lesi kulit. Tujuannya adalah agar pengguna dapat berinteraksi dengan aplikasi ini dengan baik dan mudah. Berikut pada Gambar 3.3 merupakan desain keseluruhan website pendeteksi lesi kulit.



**Gambar 3.3** Gambar Desain *Mockup* Halaman Website

Keterangan pada Gambar 3.3:

1. Header sebagai kepala dari halaman website yang berisi judul website dan menu-menu navigasi halaman website.
2. Judul pada halaman *website* sebagai deskripsi singkat dan memiliki *hero image* sebagai *background*.
3. Tombol untuk memulai coba aplikasi yang akan diarahkan ke bagian deteksi lesi kulit.
4. *Section “about skin lesion”* sebagai informasi tentang lesi kulit seperti pengertian, gejala, penanganan, dan sebagainya.
5. *Section “type of skin lesions”* sebagai informasi tentang jenis-jenis lesi kulit apa saja yang diklasifikasikan pada sistem *website* ini.
6. *Section “detect your skin lesion”* sebagai pendeteksi lesi kulit, pengguna dapat memasukkan gambar dermoscopy lesi kulit lalu menunggu hasil klasifikasi termasuk lesi kulit apa.
7. *Footer* sebagai penutup halaman dan sering digunakan untuk menampilkan informasi penting atau navigasi tambahan.

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1. Implementasi Sistem

Pada penelitian ini, sistem membutuhkan beberapa komponen dan perangkat pendukung untuk dapat melakukan pengambilan dataset, pra-pemrosesan data, pelatihan pada model DenseNet121, dan penerapannya pada aplikasi berbasis website dengan baik.

##### 4.1.1. Perangkat Keras

Dalam penelitian ini, spesifikasi perangkat keras yang digunakan meliputi:

1. AMD Ryzen 7 3700U
2. Radeon Vega Mobile Gfx 2.30 GHz
3. 8 GB RAM

##### 4.1.2. Perangkat Lunak

Dalam penelitian ini, spesifikasi perangkat lunak yang digunakan meliputi:

1. Implementasi pengambilan dataset
  - Kaggle
  - Google Drive
2. Implementasi pra-pemrosesan dataset
  - Google Colaboratory
  - *Library*: Pandas, NumPy, Matplotlib, TensorFlow, imbalanced-learn, scikit-learn
3. Implementasi membangun, melatih, dan mengevaluasi model
  - Google Colaboratory
  - *Library*: Pandas, NumPy, Matplotlib, TensorFlow
4. Implementasi model ke aplikasi berbasis *website*
  - Visual Studio Code

- Bahasa: HTML, CSS, JavaScript, Python
- *Framework*: ReactJs, Bootstrap, FastAPI

## 4.2. Implementasi Tahap Pengambilan Dataset

Penelitian ini menggunakan dataset *Skin Cancer MNIST: HAM10000*, yang tersedia di platform Kaggle. Dataset tersebut asalnya dikumpulkan dari Harvard Dataverse Repository. Dataset tersebut berisi 10015 *dermoscopy images*, yang terdiri dari 7 jenis lesi kulit yang berbeda. Jumlah data dari setiap kelas bisa ditemukan di Tabel 1.1. Gambar 4.1 merupakan proses memuat dataset berupa gambar lesi kulit:

```
base_dir = '/content/drive/MyDrive/skin-cancer-mnist-ham10000'

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x for x in glob(os.path.join(base_dir, '**', '*.jpg'))}

imageid_path_dict
```

```
{'ISIC_0033304': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033304.jpg',
'ISIC_0033294': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033294.jpg',
'ISIC_0033363': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033363.jpg',
'ISIC_0033373': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033373.jpg',
'ISIC_0033356': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033356.jpg',
'ISIC_0033387': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033387.jpg',
'ISIC_0033364': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033364.jpg',
'ISIC_0033355': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033355.jpg',
'ISIC_0033357': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033357.jpg',
'ISIC_0033365': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033365.jpg',
'ISIC_0033376': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033376.jpg',
'ISIC_0033392': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033392.jpg',
'ISIC_0033381': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033381.jpg',
'ISIC_0033346': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033346.jpg',
'ISIC_0033325': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033325.jpg',
'ISIC_0033343': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033343.jpg',
'ISIC_0033395': '/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_images_part_2/ISIC_0033395.jpg',
```

**Gambar 4.1** Memuat Dataset Gambar Lesi Kulit

## 4.3. Implementasi Tahap Pra-pemrosesan Data

Setelah dataset dimuat, dilakukan tahap pra-pemrosesan data meliputi transformasi gambar ke *array NumPy*, *resize* dan *normalization*, *label encoding*, *data splitting*, *data balancing*, serta *train-val-test splitting*. Berikut proses yang dilakukan:

### 4.3.1. Transformasi Gambar ke Array NumPy, Resize dan Normalization

Gambar beserta label yang ada pada dataset diletakkan ke dalam bentuk dataframe agar memudahkan proses dalam tahap pra-pemrosesan data. Kemudian data image yang berupa path gambar diubah ke dalam bentuk array NumPy agar

gambar dapat digunakan sebagai data input untuk model. Kemudian, gambar dilakukan resize ke ukuran 120×120 px agar memastikan bahwa semua input memiliki dimensi yang sama, sehingga dapat diproses oleh model tanpa masalah. Setelah itu, dilakukan normalisasi pada gambar dengan membagi setiap nilai piksel dengan 255 untuk gambar 8-bit. Dengan cara tersebut dapat menormalkan rentang piksel menjadi 0 hingga 1 guna menyederhanakan data input dan membuatnya lebih mudah untuk diolah oleh model pembelajaran mesin. Implementasinya seperti pada Gambar 4.2:

```
size = (120, 120)

df['path'] = df['image_id'].map(imageid_path_dict.get)
df['image'] = df['path'].map(lambda x: np.asarray(Image.open(x).resize(size))/ 255)

df = df[['dx', 'image']]

df.head()
```

	dx	image
0	bkl	[[[0.7411764705882353, 0.596078431372549, 0.76...
1	bkl	[[[0.09411764705882353, 0.050980392156862744, ...
2	bkl	[[[0.7333333333333333, 0.4980392156862745, 0.5...
3	bkl	[[[0.09411764705882353, 0.043137254901960784, ...
4	bkl	[[[0.5098039215686274, 0.34509803921568627, 0....

**Gambar 4.2** Transformasi Gambar ke Array NumPy, Resize, dan Normalization

### 4.3.2. Label Encoding

*Label encoding* adalah proses mengubah *label* kelas atau kategori yang direpresentasikan sebagai teks atau string menjadi bilangan bulat atau integer. Hal ini memungkinkan model untuk mengenali dan membedakan antara kelas-kelas yang berbeda dengan menggunakan representasi numerik, yang diperlukan untuk penggunaan algoritma klasifikasi dan analisis data secara efisien. Pada penelitian ini, Tabel 4.1 berikut menunjukkan hasil dari implementasi *label encoding*:

**Tabel 4.1** Hasil Teknik *Label Encoding*

Kelas	Label	Hasil Label Encoding
Actinic Keratoses	akiec	0
Basal Cell Carcinoma	bcc	1
Benign Keratosis-Like Lesions	bkl	2



Dermatofibroma	df	3
Melanoma	mel	4
Melanocytic Nevi	nv	5
Vascular Lesions	vasc	6

#### 4.3.3. Data Splitting

Kemudian, *data splitting* menjadi variabel independen (X) dan variabel dependen (y) adalah langkah penting untuk persiapan data *machine learning*. X merepresentasikan fitur-fitur atau variabel independen dari dataset, sedangkan y merepresentasikan variabel target atau dependen yang ingin diprediksi atau dipelajari oleh model. Pada penelitian ini, variabel independen (X) berupa *image* atau gambar sedangkan variabel independen (y) berupa label atau kelas dari gambar tersebut. Hal ini memungkinkan untuk memisahkan *input* (fitur) dari *output* (target) sehingga dapat dianalisis dan diproses lebih lanjut, termasuk dalam tahap pelatihan, validasi, dan pengujian model.

#### 4.3.4. Data Balancing

Dataset dalam penelitian ini memiliki ketidakseimbangan dalam jumlah data pada masing-masing kelas seperti pada Tabel 1.1. Karenanya, perlu dilakukan data balancing atau penyeimbangan data pada setiap kelas. Penelitian ini memanfaatkan SMOTE dan Augmentasi Data. Berdasarkan Tabel 4.2 perbandingannya terhadap jumlah distribusi dataset menjadi relatif lebih seimbang pada masing-masing kelas daripada distribusi data pada dataset original.

**Tabel 4.2** Perbandingan Data Asli dengan Data yang Diterapkan *Data Balancing*

	Original	SMOTE	Augmentasi Data
Actinic keratoses (akiec)	327	6705	6699
Basal cell carcinoma (bcc)	514	6705	6694
Benign keratoses-like lesions (bkl)	1099	6705	6703
Dermatofibroma (df)	115	6705	6697
Melanoma (mel)	1113	6705	6693

<b>Melanocytic nevi (nv)</b>	6705	6705	6705
<b>Vascular lesions (vasc)</b>	142	6705	6694

Implementasi dengan SMOTE dengan cara memilih sejumlah tetangga terdekat untuk setiap sampel dalam kelas minoritas. Dengan memanfaatkan informasi dari tetangga-tetangga ini, SMOTE menghasilkan contoh baru dengan cara interpolasi. Interpolasi adalah metode untuk memperkirakan nilai-nilai di antara dua titik data yang diketahui, dalam hal ini antara dua sampel kelas minoritas. Ini berarti contoh sintetis baru dihasilkan dengan mengambil titik di antara sampel asli dan tetangganya. Ilustrasinya dapat dilihat pada Gambar 2.10.

Sedangkan implementasi dengan Augmentasi Data menggunakan teknik:

- Augmentasi Kecerahan (*Brightness Augmentation*): Mengubah kecerahan gambar secara acak dalam rentang yang ditentukan oleh parameter.
- Augmentasi Kontras (*Contrast Augmentation*): Mengubah kontras gambar secara acak dalam rentang yang ditentukan oleh parameter.
- Pembalikan Kiri-Kanan (*Flip Left-Right Augmentation*): Membalik gambar secara acak dari kiri ke kanan.
- Pembalikan Atas-Bawah (*Flip Up-Down Augmentation*): Membalik gambar secara acak dari atas ke bawah.
- Augmentasi Hue (*Hue Augmentation*): Mengubah *hue* (warna dasar) gambar secara acak dalam rentang yang ditentukan oleh parameter.
- Augmentasi Saturasi (*Saturation Augmentation*): Mengubah saturasi gambar secara acak dalam rentang yang ditentukan oleh parameter.

#### 4.3.5. Train-Val-Test Splitting

Setelah itu, dilakukan pembagian data menjadi data pelatihan (*train*), data validasi (*validation*), dan data pengujian (*test*). Pembagian ini dilakukan dengan rasio pembagian dataset 70-15-15, yaitu 70% data pelatihan dan masing-masing 15% untuk data validasi dan data pengujian. Rasio ini digunakan karena pertimbangan jumlah data yang relatif banyak dan mumpuni untuk pembagian data validasi dan data uji, sehingga data pelatihan diberikan lebih banyak daripada data

validasi dan uji. Hasil pembagian *train-val-test split* untuk masing-masing dataset, yaitu baik dataset asli, maupun dataset yang menerapkan teknik *data balancing* terdapat pada Tabel 4.3.

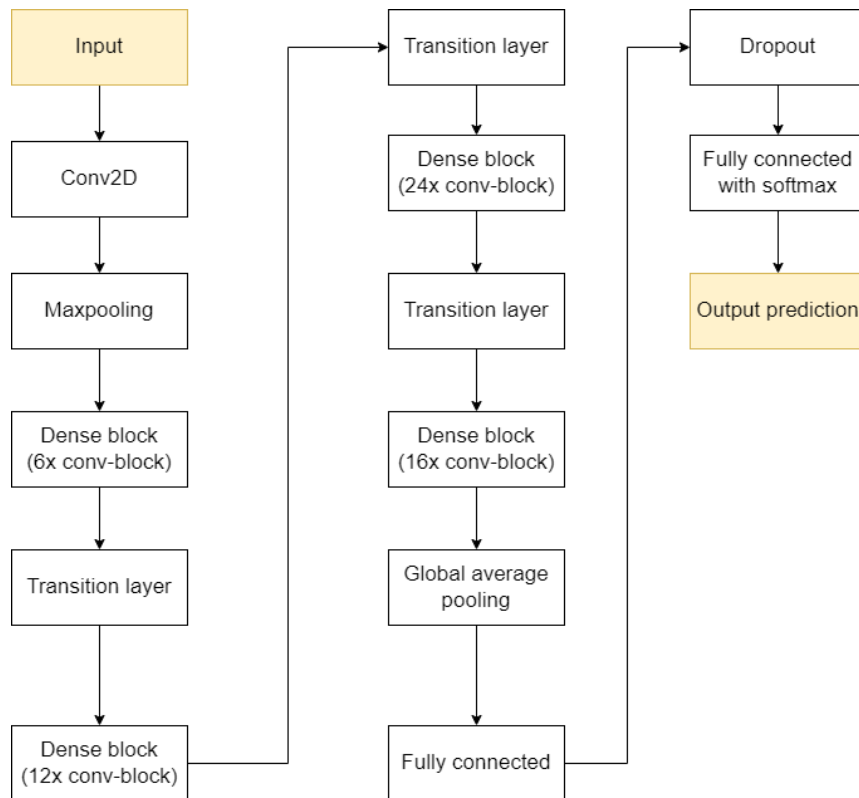
**Tabel 4.3** Distribusi Data untuk *Train-Val-Test Split*

	Original	SMOTE	Augmentasi Data
<b>X_train</b>	(7010, 120, 120, 3)	(32854, 120, 120, 3)	(32819, 120, 120, 3)
<b>X_val</b>	(1502, 120, 120, 3)	(7040, 120, 120, 3)	(7033, 120, 120, 3)
<b>X_test</b>	(1503, 120, 120, 3)	(7041, 120, 120, 3)	(7033, 120, 120, 3)
<b>y_train</b>	(7010,)	(32854,)	(32819,)
<b>y_val</b>	(1502,)	(7040,)	(7033,)
<b>y_test</b>	(1503,)	(7041,)	(7033,)

#### 4.4. Implementasi Model

Setelah tahap pra-pemrosesan selesai, selanjutnya tahap implementasi model. Pada penelitian ini menggunakan model DenseNet121. Tahap ini dilakukan dengan menggunakan Library Tensorflow dan Keras. Stelah itu, import modul yang diperlukan untuk mengimplementasikan DenseNet121. DenseNet121 dimuat dengan pretrained weights, yaitu menggunakan bobot yang telah dilatih dengan dataset ImageNet, serta membuat ``include_top=False`` yaitu tidak menyertakan fully-connected layer teratas dari DenseNet121 agar dapat menambahkan layer kustom di atasnya. Selain itu, input\_shape dari image yang dilatih adalah  $120\text{ px} \times 120\text{ px}$  dengan 3 channel RGB.

Selanjutnya, menambahkan *GlobalAveragePooling2D* untuk mengubah seluruh fitur map menjadi vektor satu dimensi. Lalu menambahkan dense layer tambahan untuk meningkatkan kapasitas model dalam memahami pembelajaran dan dropout layer untuk mencegah *overfitting*. Terakhir, layer prediction adalah layer output dari model yang menentukan kelas prediksi akhir. Layer ini biasanya berupa layer *Dense* dengan jumlah unit yang sesuai dengan jumlah kelas dalam masalah klasifikasi, dan menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas masing-masing kelas. Gambar 4.3 menampilkan ilustrasi arsitektur.



**Gambar 4.3** Arsitektur DenseNet121 yang Digunakan dalam Penelitian

Model tersebut dikompilasi (*compile*) menggunakan optimizer Adam dengan learning rate sebesar 0.001, *loss function sparse categorical cross-entropy*, dan metrik evaluasi adalah akurasi. Optimizer Adam (*Adaptive Moment Estimation*) adalah algoritma optimasi yang digunakan untuk pelatihan jaringan saraf dengan menggabungkan fitur-fitur dari algoritma *RMSprop* (*Root Mean Square Propagation*) dan Momentum.

Penggunaan Adam sering kali konvergen lebih cepat dibandingkan dengan optimizers lain serta penggunaan memori yang efisien. Sedangkan, penggunaan fungsi *loss sparse categorical crossentropy* dalam masalah klasifikasi multi-kelas memberikan keuntungan dalam hal efisiensi memori, kesederhanaan penggunaan, dan performa komputasi. Fungsi *loss* ini sangat cocok ketika label target disimpan sebagai integer, yang membuatnya menjadi pilihan yang lebih efisien.

#### 4.5. Implementasi Tahap Pelatihan Model

Selanjutnya adalah tahap pelatihan model. Pada penelitian ini, pelatihan model dilatih sebanyak 60 *epochs* dengan menggunakan *batch size* sebanyak 64.

Pelatihan juga menggunakan beberapa fungsi *callback*, yaitu *EarlyStopping* dengan parameter monitor di *val\_loss* dan *patience* sebesar 10, selanjutnya *ReduceLROnPlateau* dengan parameter monitor juga di *val\_loss* dan *patience* sebesar 10. Pelatihan model dilakukan dengan menggunakan data pelatihan, yaitu *X\_train* dan *y\_train*, serta menggunakan data validasi, yaitu *X\_val*, dan *y\_val*. Hal ini dilakukan agar dapat melihat kinerja model dengan melihat akurasi dan *loss* pada data tersebut.

Pelatihan ini dilakukan pada masing-masing dataset asli dan dataset yang sudah diterapkan teknik SMOTE dan teknik Augmentasi Data. Hasil dari pelatihan untuk setiap dataset dapat ditemukan pada Tabel 4.4 untuk pelatihan model DenseNet121 dengan dataset asli, Tabel 4.5 untuk pelatihan model DenseNet121 dengan SMOTE, dan Tabel 4.6 untuk pelatihan model DenseNet121 dengan Augmentasi Data.

**Tabel 4.4** Hasil Pelatihan Model DenseNet121 dengan Dataset Asli

epoch	accuracy	loss	learning_rate	val_accuracy	val_loss
19	0.8927	0.2976	0.001	0.7690	0.6776
20	0.8961	0.2871	0.001	0.7790	0.6862
21	0.9020	0.2672	0.001	0.7710	0.7226
22	0.9057	0.2605	0.001	0.7810	0.7037
23	0.9175	0.2366	0.001	0.7830	0.7095

**Tabel 4.5** Hasil Pelatihan Model DenseNet121 + SMOTE

epoch	accuracy	loss	learning_rate	val_accuracy	val_loss
38	0.9607	0.1091	0.001	0.9396	0.2393
39	0.9631	0.1057	0.001	0.9403	0.2306
40	0.9659	0.0965	0.001	0.9406	0.2363
41	0.9627	0.1059	0.001	0.9358	0.2507
42	0.9648	0.1011	0.001	0.9401	0.2463

**Tabel 4.6** Hasil Pelatihan Model DenseNet121 + Augmentasi Data

epoch	accuracy	loss	learning_rate	val_accuracy	val_loss
50	0.9633	0.1022	0.001	0.9315	0.2701
51	0.9637	0.1031	0.001	0.9306	0.2749
52	0.9638	0.1034	0.001	0.9275	0.2760
53	0.9636	0.1031	0.001	0.9312	0.2761
54	0.9638	0.1011	0.001	0.9322	0.2549

#### 4.6. Implementasi Tahap Evaluasi dan Pengujian Model

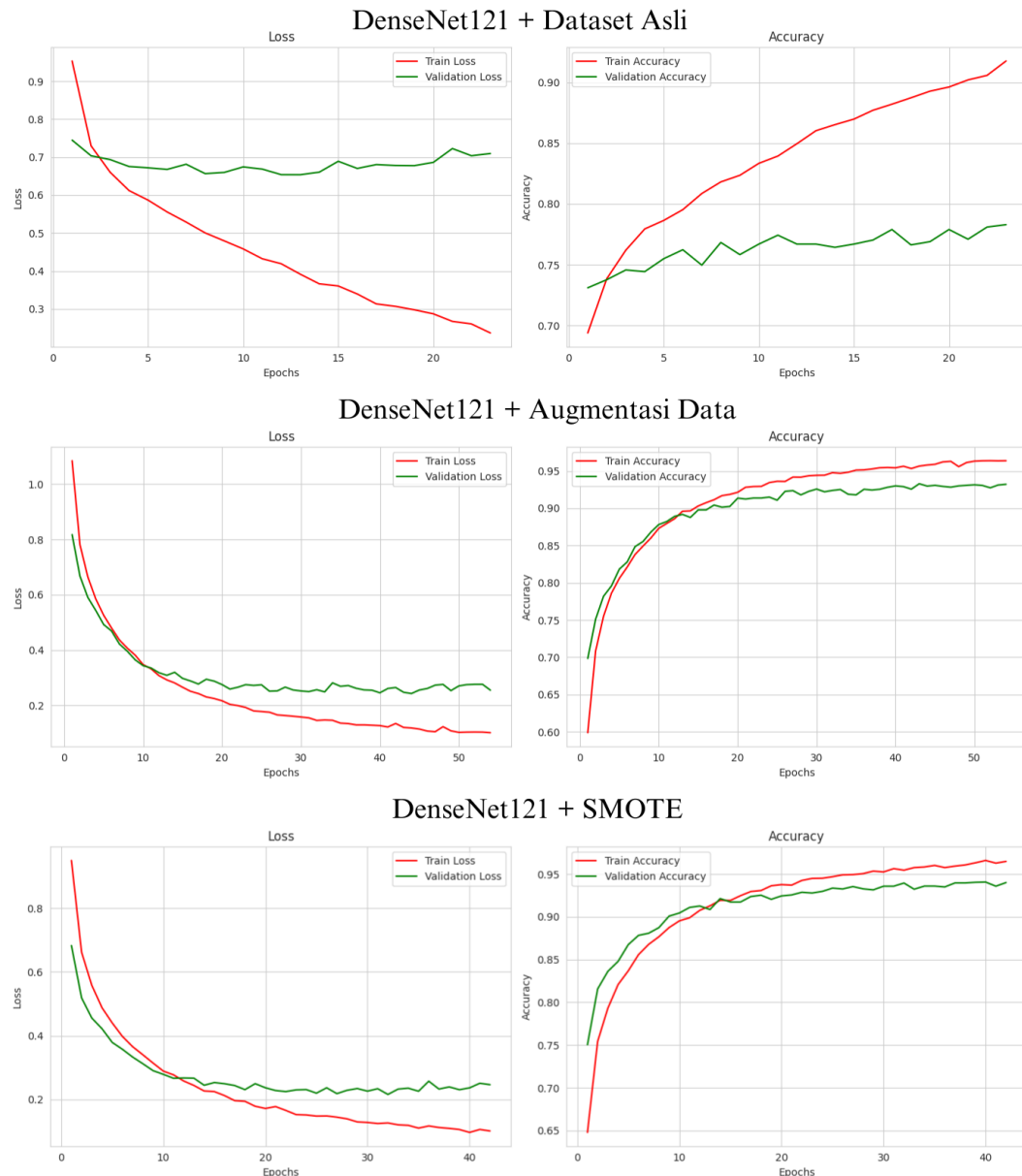
Berdasarkan hasil pelatihan model yang telah dilakukan sebelumnya, diperoleh evaluasi untuk masing-masing model. Hasil pelatihan ini ditampilkan dalam bentuk grafik untuk analisis lebih lanjut mengenai performa setiap model. Grafik pada Gambar 4.4 mencakup metrik akurasi dan *loss function* untuk pelatihan dan validasi.

Melalui Gambar 4.4 dapat diperhatikan bahwa hasil pelatihan model DenseNet121 + dataset asli itu berhenti pada *epoch* 23, yang mendapatkan akurasi pelatihan sebesar 91.75% dan akurasi validasi sebesar 78.30%, serta loss function pelatihan sebesar 0.2366 dan loss function validasi sebesar 0.7095. Sedangkan hasil pelatihan model DenseNet121 + Augmentasi Data berhenti pada *epoch* 54, yang mendapatkan akurasi pelatihan sebesar 96.38% dan akurasi validasi sebesar 93.22%, serta loss function pelatihan sebesar 0.1011 dan loss function validasi sebesar 0.2549. Di sisi lain, hasil pelatihan model DenseNet121 + SMOTE berhenti pada *epoch* 42, yang mendapatkan akurasi pelatihan sebesar 96.48% dan akurasi validasi sebesar 94.01%, serta loss function pelatihan sebesar 0.1011 dan loss function validasi sebesar 0.2463.

Hasil perbandingan secara keseluruhan antara masing-masing model DenseNet121 dengan dataset asli maupun dataset yang sudah dilakukan *data balancing* dengan Augmentasi Data dan SMOTE bisa ditemukan pada Tabel 4.7.

**Tabel 4.7** Perbandingan Metrik Akurasi dan *Loss* pada *Epoch* Terakhir Pelatihan

	<b>Last epoch</b>	<b>accuracy</b>	<b>val_acc</b>	<b>loss</b>	<b>val_loss</b>	<b>Learning _rate</b>
<b>DenseNet121</b>	23	91.75%	78.30%	0.2366	0.7095	0.001
<b>DenseNet121 + Augmentasi Data</b>	54	96.38%	93.22%	0.1011	0.2549	0.001
<b>DenseNet121 + SMOTE</b>	42	96.48%	94.01%	0.1011	0.2463	0.001



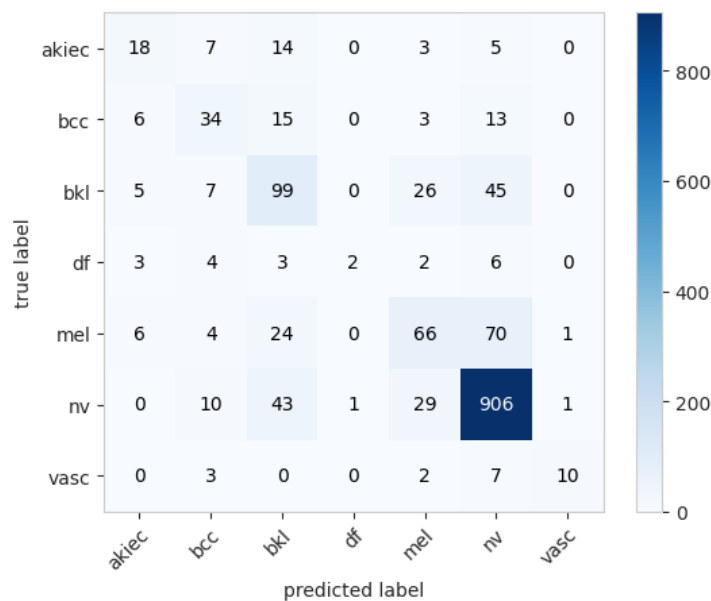
**Gambar 4.4** Grafik Akurasi dan Loss Function pada Pelatihan dan Validasi Model

Berdasarkan Tabel 4.7 menunjukkan bahwa model DenseNet121 (tanpa teknik tambahan) memiliki performa paling buruk dengan akurasi validasi rendah, validation loss tinggi, dan lebih cepat mencapai *epoch* terakhir namun tidak konvergen dengan baik. Ini menunjukkan bahwa model ini *overfitting* pada data pelatihan. Sedangkan model DenseNet121 + Augmentasi Data berhasil meningkatkan performa secara signifikan dibandingkan model dasar. Akurasi pelatihan dan validasi tinggi, serta validation loss lebih rendah. Namun,



membutuhkan lebih banyak *epochs* untuk mencapai performa terbaiknya. Di sisi lain, model DenseNet121 + SMOTE juga berhasil meningkatkan performa secara signifikan dibandingkan model dasar dengan akurasi pelatihan dan validasi tinggi, serta *validation loss* terendah di antara ketiga model. Meskipun membutuhkan lebih sedikit *epochs* dibandingkan dengan Augmentasi Data, SMOTE masih memberikan performa yang lebih unggul.

Selain itu, dilakukan pula tahap pengujian model DenseNet121 dengan dataset asli maupun dataset yang sudah dilakukan *data balancing*, menggunakan data uji (*test set*). Tahap pengujian menggunakan *confusion matrix* dan metrik evaluasi, *presicion*, *recall*, serta F1-Score untuk memeriksa kemampuan model dalam memprediksi data baru yang tidak terlihat selama pelatihan. Gambar 4.5, Gambar 4.7, dan Gambar 4.9 menunjukkan *confusion matrix* dari masing-masing model, serta Gambar 4.6, Gambar 4.8, dan Gambar 4.10 merupakan *classification report* dari masing-masing kelas yang diprediksi.



**Gambar 4.5** *Confusion Matrix* Model DenseNet121 dengan Dataset Asli

Berdasarkan Gambar 4.5 yang merupakan *Confusion Matrix* model DenseNet121 dengan Dataset Asli menunjukkan bahwa kelas *nv* merupakan kelas terbaik yang diprediksi model, karena memiliki prediksi benar sebanyak 906 dari total data 990 dan total prediksi salah sebanyak 84, dengan prediksi salah terbanyak jatuh ke kelas *bkl* sebanyak 43. Di sisi lain, kelas *df* merupakan kelas terburuk yang

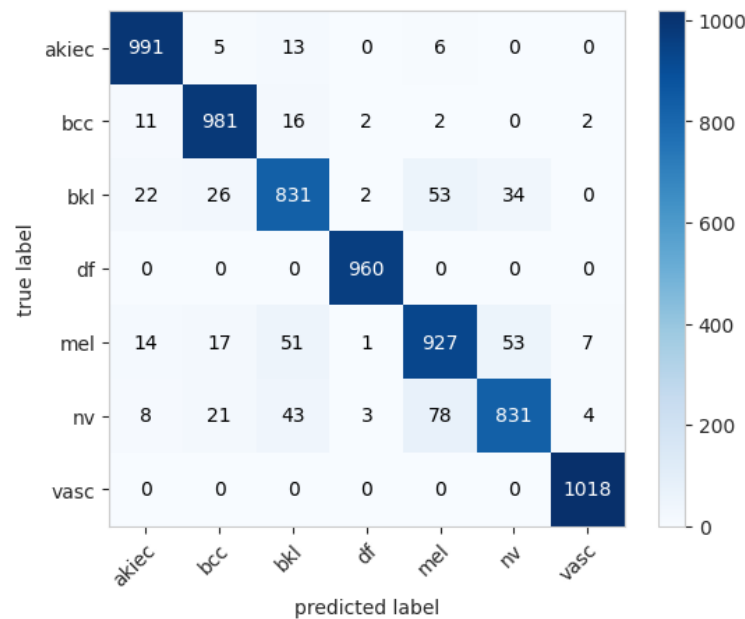
diprediksi model, karena memiliki prediksi benar hanya sebanyak 2 dari total data 20 dan total prediksi salah sebanyak 18 dengan prediksi salah terbanyak jatuh kepada kelas *nv* sebanyak 6.

	precision	recall	f1-score	support
akiec	0.47	0.38	0.42	47
bcc	0.49	0.48	0.49	71
bkl	0.50	0.54	0.52	182
df	0.67	0.10	0.17	20
mel	0.50	0.39	0.44	171
nv	0.86	0.92	0.89	990
vasc	0.83	0.45	0.59	22
accuracy			0.76	1503
macro avg	0.62	0.47	0.50	1503
weighted avg	0.74	0.76	0.74	1503

**Gambar 4.6** *Classification Report* DenseNet121 dengan Dataset Asli

Berdasarkan Gambar 4.6 yang merupakan *classification report* model DenseNet121 dengan Dataset Asli menunjukkan bahwa kelas terbaik yang diprediksi model adalah kelas *nv*, karena memiliki prediksi benar terbanyak dengan *precision* dan *recall* yang paling tinggi, masing-masing 86% dan 92%, yang membuat F-1 score nya cukup bagus yaitu 89%.

Di sisi lain, kelas *df* merupakan kelas terburuk diprediksi karena memiliki nilai F-1 score hanya 17% dan *recall* yang sangat rendah hanya 10%. Ini adalah indikasi bahwa model kesulitan dalam mengenali kelas ini. Meskipun *Precision* cukup tinggi yaitu 67%, tetapi karena *recall* sangat rendah, model sering kali mengabaikan *df*, yang menyebabkan F1-score buruk. ***Precision*** adalah dari semua prediksi positif yang dibuat oleh model, berapa banyak yang benar. Sedangkan, ***recall*** adalah dari semua kasus positif yang ada, berapa banyak yang berhasil ditemukan sebagai positif oleh model. **F1-score** yang tinggi menunjukkan keseimbangan antara *precision* dan *recall*, sehingga memberikan gambaran yang lebih komprehensif tentang kinerja model.



**Gambar 4.7** *Confusion Matrix* Model DenseNet121 + Augmentasi Data

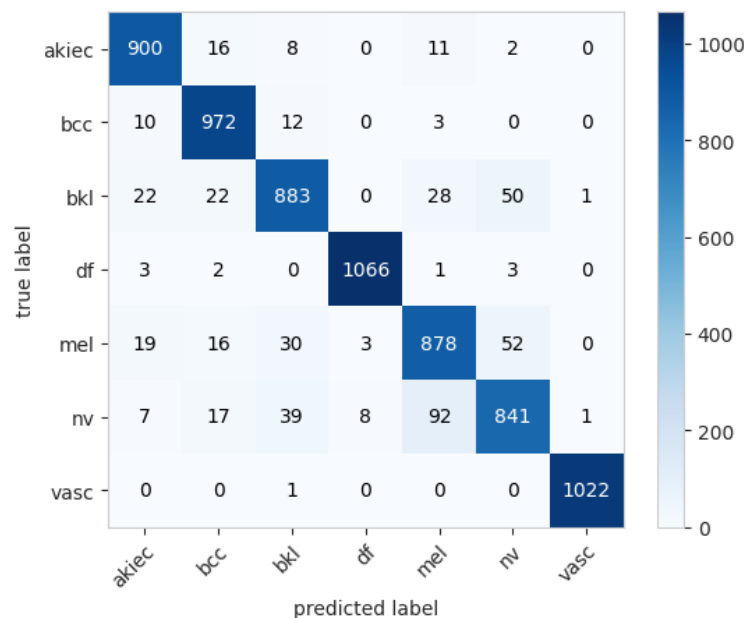
Berdasarkan Gambar 4.7 yang merupakan *Confusion Matrix* model DenseNet121 + Augmentasi Data menunjukkan bahwa kelas terbaik yang diprediksi model adalah kelas *vasc* dan *df* karena tidak terdapat prediksi yang salah, namun kelas *vasc* memiliki lebih banyak distribusi data daripada kelas *df* yaitu 1018 untuk kelas *vasc* dan 960 untuk kelas *df*. Di sisi lain, kelas *nv* merupakan kelas terburuk yang diprediksi model, karena memiliki prediksi benar sebanyak 831 dari total data 998 dan total prediksi salah sebanyak 157, dengan prediksi salah terbanyak jatuh kepada kelas *mel* sebanyak 78.

	precision	recall	f1-score	support
akiec	0.95	0.98	0.96	1015
bcc	0.93	0.97	0.95	1014
bkl	0.87	0.86	0.86	968
df	0.99	1.00	1.00	960
mel	0.87	0.87	0.87	1070
nv	0.91	0.84	0.87	988
vasc	0.99	1.00	0.99	1018
accuracy			0.93	7033
macro avg	0.93	0.93	0.93	7033
weighted avg	0.93	0.93	0.93	7033

**Gambar 4.8** *Classification Report* Model DenseNet121 + Augmentasi Data

Berdasarkan Gambar 4.8 yang merupakan *classification report* model DenseNet121 + Augmentasi Data menunjukkan bahwa kelas terbaik yang diprediksi model adalah kelas **df** karena memiliki F1-score tertinggi sebesar 100%, yang menunjukkan bahwa model mampu memprediksi kelas ini dengan sangat baik. *Recall* yang sempurna sebesar 100% berarti model berhasil mengidentifikasi semua sampel **df** tanpa salah prediksi (tidak ada *false negatives*). *Precision* yang juga sangat tinggi sebesar 99% menunjukkan bahwa hampir semua prediksi positif untuk kelas ini benar-benar positif.

Di sisi lain, kelas **bkl** merupakan kelas terburuk diprediksi karena memiliki F1-score terendah sebesar 86% di antara semua kelas, yang menunjukkan bahwa model tidak dapat memprediksi kelas ini seakurat kelas lainnya. *Recall* yang relatif lebih rendah sebesar 86% menunjukkan bahwa ada sejumlah false negatives, yaitu beberapa sampel **bkl** tidak terdeteksi dan salah diklasifikasikan sebagai kelas lain. *Precision* yang juga lebih rendah sebesar 87% menunjukkan bahwa ada juga kesalahan dalam prediksi positif untuk kelas ini, sehingga model tidak seakurat itu untuk memprediksi kelas ini dibandingkan dengan kelas lainnya.



**Gambar 4.9** *Confusion Matrix* Model DenseNet121 + SMOTE

Berdasarkan Gambar 4.9 yang merupakan *Confusion Matrix* dari Model DenseNet121 + SMOTE menunjukkan bahwa kelas terbaik yang diprediksi model adalah kelas **vasc** karena memiliki prediksi benar sebanyak 1022 dari total data

1023 dengan hanya 1 kesalahan yaitu salah prediksi ke kelas *bkl*. Ini menunjukkan kelas *vasc* memiliki *True Positives* yang sangat tinggi yang berarti bahwa model mengenali hampir seluruh sampel *vasc* dengan benar, dan tidak membingungkan kelas ini dengan yang lain.

Di sisi lain, kelas *nv* merupakan kelas terburuk yang diprediksi model, karena memiliki prediksi benar sebanyak 841 dari total data 1005 dan memiliki jumlah kesalahan yang cukup tinggi dengan total kesalahan *False Negatives* sebanyak 163 sampel, yang berarti bahwa model sering salah mengidentifikasi sampel *nv* sebagai kelas lain. Selain itu, kesalahan terjadi dari berbagai kelas yang berbeda, yang menunjukkan bahwa *nv* lebih sulit dibedakan oleh model dibandingkan kelas lain.

	precision	recall	f1-score	support
akiec	0.94	0.96	0.95	937
bcc	0.93	0.97	0.95	997
bkl	0.91	0.88	0.89	1006
df	0.99	0.99	0.99	1075
mel	0.87	0.88	0.87	998
nv	0.89	0.84	0.86	1005
vasc	1.00	1.00	1.00	1023
accuracy			0.93	7041
macro avg	0.93	0.93	0.93	7041
weighted avg	0.93	0.93	0.93	7041

**Gambar 4.10** *Classification Report* Model DenseNet121 + SMOTE

Berdasarkan Gambar 4.10 yang merupakan *classification report* model DenseNet121 + SMOTE menunjukkan bahwa kelas terbaik yang diprediksi model adalah kelas *vasc* karena memiliki nilai 100% untuk seluruh metrik yaitu *precision*, *recall*, dan F1-score, yang berarti model memprediksi kelas ini dengan sempurna. Di sisi lain, kelas *nv* merupakan kelas terburuk diprediksi karena memiliki F1-score terendah sebesar 86% di antara semua kelas, yang berarti bahwa kinerja model untuk kelas ini tidak sebaik kelas lainnya, yang menunjukkan cukup banyak *false negatives* maupun *false positives* yang diprediksi model pada kelas ini.

Sementara pada Tabel 4.8 menunjukkan perbandingan metrik evaluasi pada masing-masing model, yang menggunakan data pengujian.

**Tabel 4.8** Perbandingan Metrik Evaluasi menggunakan Data Pengujian

	<b>Accuracy</b>	<b>Loss</b>	<b>Precision</b>	<b>Recall</b>	<b>F1-Score (Weighted average)</b>
<b>DenseNet121 + Dataset Asli</b>	75.52%	0.6994	62%	47%	74%
<b>DenseNet121 + Augmentasi Data</b>	92.98%	0.2813	93%	93%	93%
<b>DenseNet121 + SMOTE</b>	93.20%	0.2341	93%	93%	93%

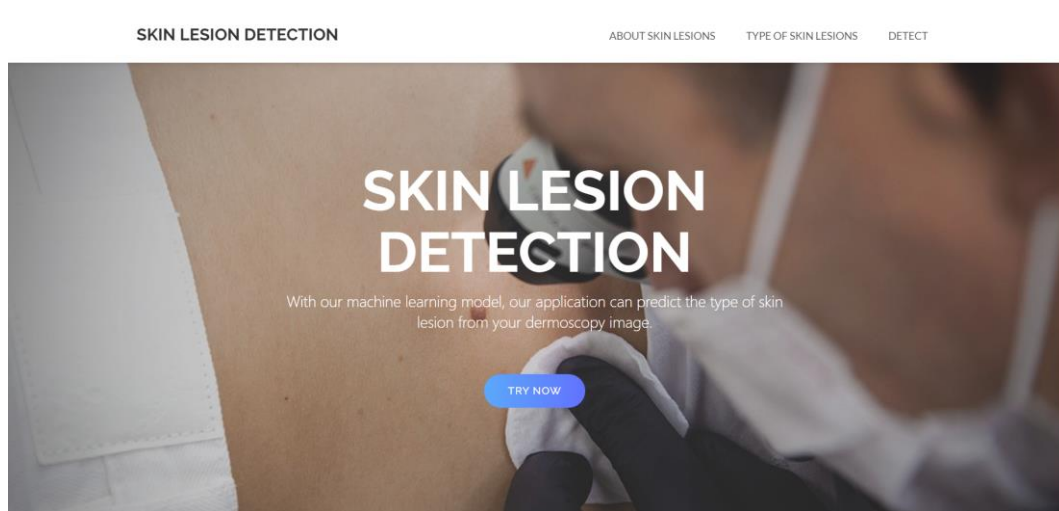
Berdasarkan Tabel 4.8 model DenseNet121 (tanpa teknik tambahan) memiliki performa terburuk di antara ketiga model. Akurasi rendah, loss tinggi, dan metrik precision, recall, serta F1-Score yang rendah menunjukkan bahwa model ini tidak efektif dalam mengenali pola dari data. Sedangkan model DenseNet121 + Augmentasi Data berhasil meningkatkan performa secara signifikan dibandingkan model dasar. Akurasi tinggi, loss rendah, dan metrik precision, recall, serta F1-Score yang tinggi menunjukkan bahwa Augmentasi Data membantu model untuk mengenali pola dari data dengan sangat baik. Di sisi lain, model DenseNet121 + SMOTE juga berhasil meningkatkan performa secara signifikan dibandingkan model dasar. Akurasi dan metrik lain sangat mirip dengan model Augmentasi Data, namun memiliki loss terendah di antara ketiga model, menunjukkan bahwa SMOTE efektif dalam menangani ketidakseimbangan data. DenseNet121 + SMOTE sedikit lebih unggul dalam hal loss, meskipun akurasi dan metrik lainnya hampir identik dengan DenseNet121 + Augmentasi Data. Oleh karena itu, DenseNet121 + SMOTE merupakan pilihan yang sedikit lebih baik dalam hal meminimalkan kesalahan prediksi.

#### **4.7. Implementasi Model ke Aplikasi Berbasis Website**

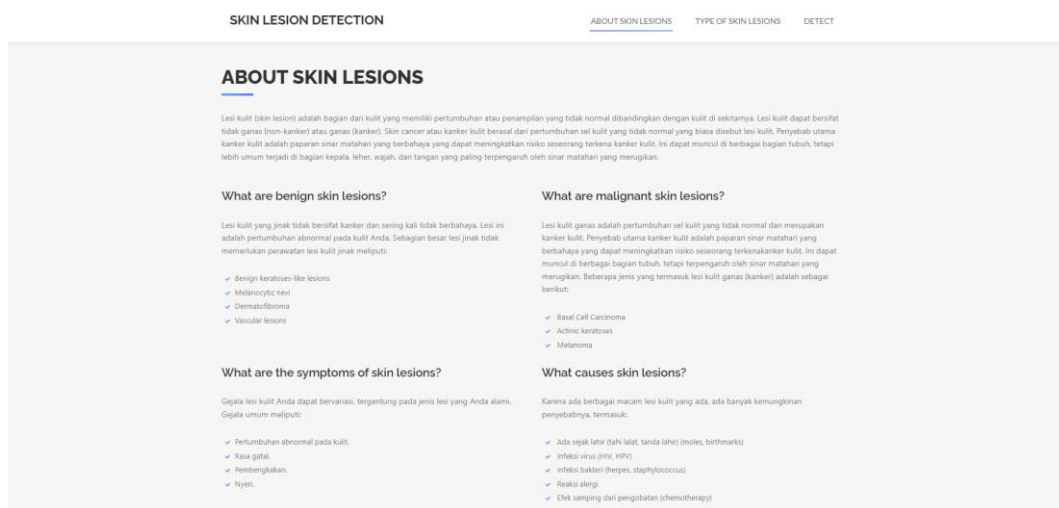
Tahap selanjutnya adalah implementasi model dengan evaluasi terbaik ke aplikasi website. Pada penelitian ini, aplikasi website dibangun dengan bahasa HTML, CSS, JavaScript, dan *framework ReactJs* dan *Bootstrap* untuk frontend.

Untuk backend menggunakan bahasa Python dan *framework FastAPI*. Model yang digunakan disimpan dalam bentuk *file* dengan format HDF5 (*Hierarchical Data Formats*) atau .h5, yang umumnya digunakan untuk menyimpan data dalam jumlah besar berupa *array* multidimensi. Ukuran berkas tersebut sebesar 34,0 MB.

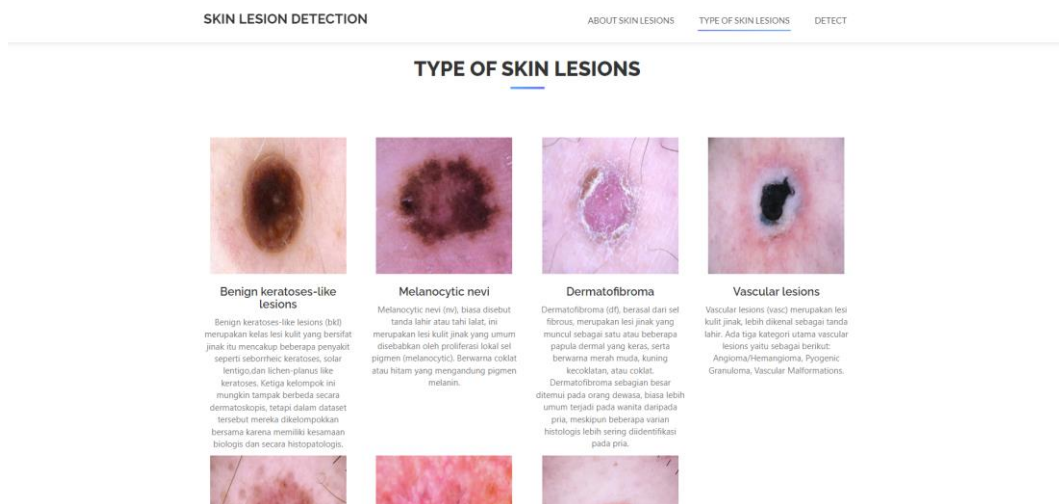
Berdasarkan analisis dan perancangan sistem pada aplikasi berbasis website yang terdapat pada Bab 3, dibangun tampilan website yang bisa ditemukan dalam Gambar 4.11 hingga Gambar 4.15.



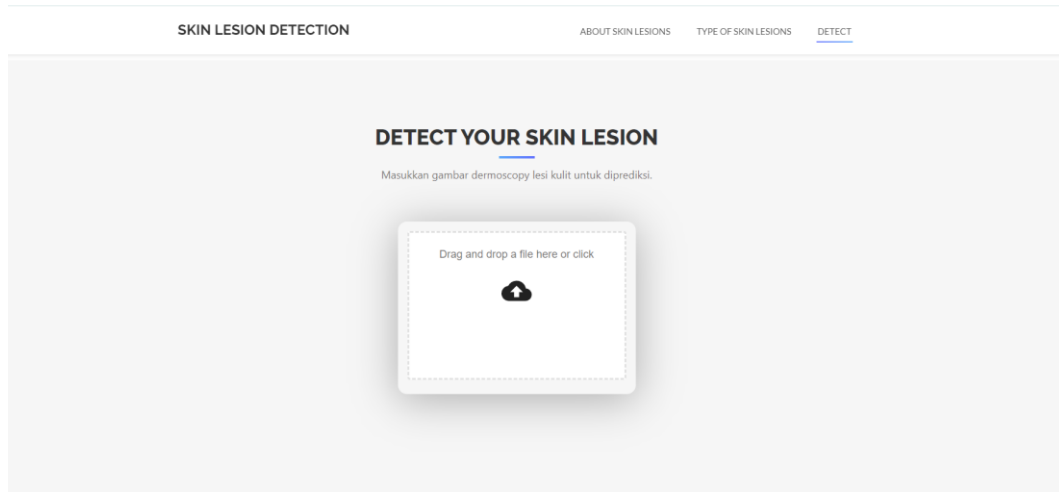
**Gambar 4.11** Tampilan Beranda



**Gambar 4.12** Tampilan *About Skin Lesions*



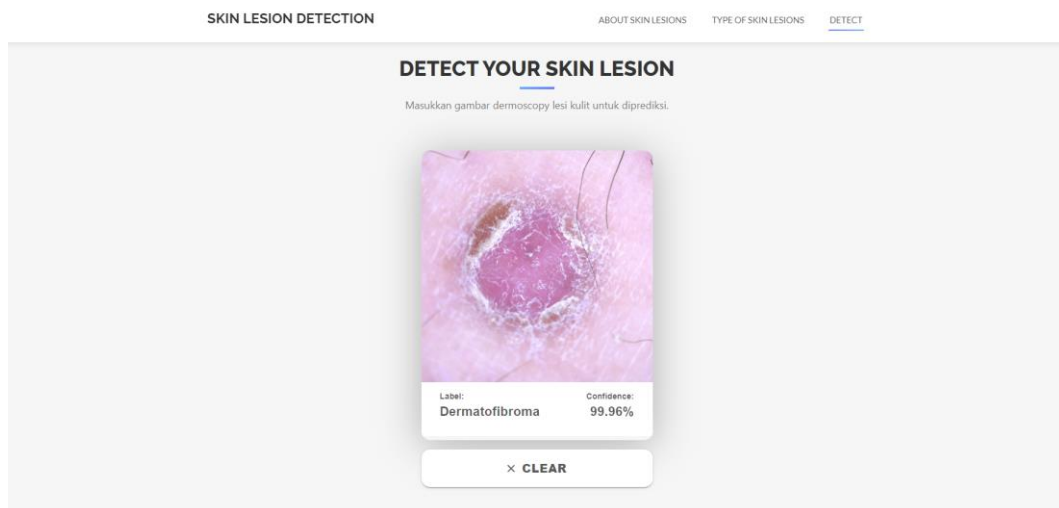
**Gambar 4.13** Tampilan Tipe *Skin Lesions*



© 2024 University of Sumatera Utara. All rights reserved.

**Gambar 4.14** Tampilan Deteksi *Skin Lesion*



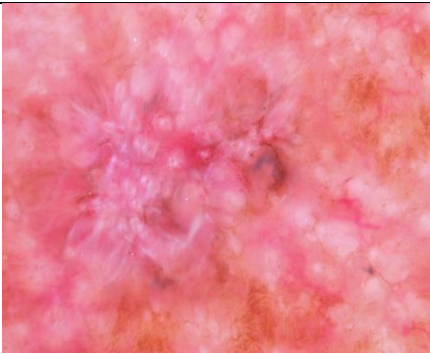


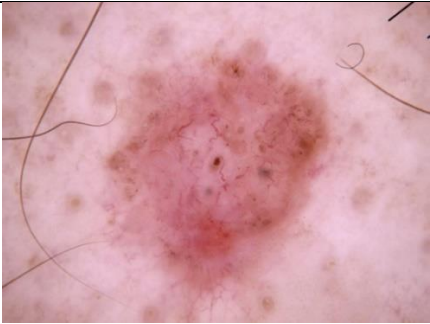


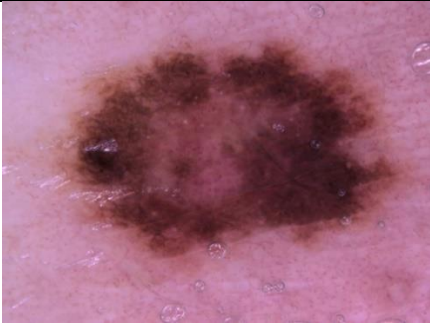
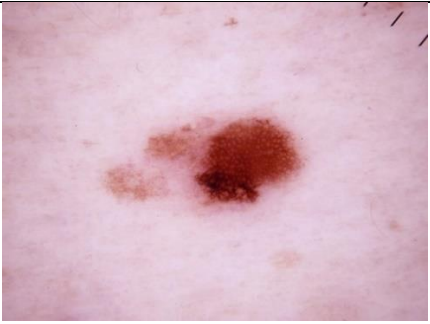
**Gambar 4.15** Tampilan Hasil Deteksi *Skin Lesion*


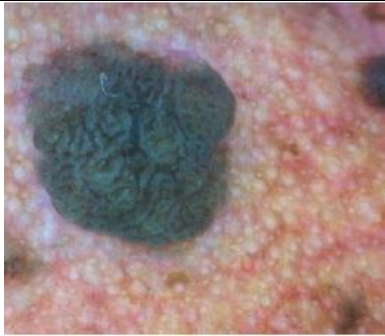
#### 4.8. Pengujian Sistem

Tahap berikutnya melibatkan evaluasi sistem, bertujuan memastikan bahwa sistem berfungsi baik dan akurat dalam memprediksi gambar lesi kulit. Pada tahap pengujian, terdapat 7 gambar dermoscopy lesi kulit yang akan digunakan untuk prediksi lesi kulit. Tabel 4.9 menunjukkan daftar data pengujian.

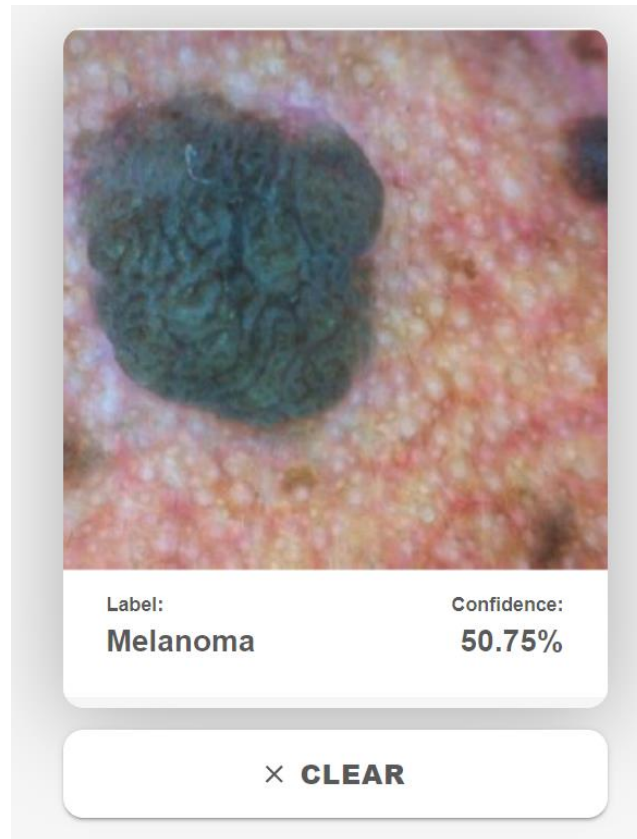
**Tabel 4.9** *Data Testing* yang Menjadi *Input* Ketika Pengujian Sistem

No.	Input	Nama Lesi Kulit	Hasil Prediksi Sistem
1.		Actinic keratoses	Actinic keratoses

2.		Basal cell carcinoma	Basal cell carcinoma
3.		Benign keratosis like-lesions	Benign keratosis like-lesions
4.		Dermatofibroma	Dermatofibroma
5.		Melanocytic nevi	Melanocytic nevi
6.		Melanoma	Melanoma

7.		Vascular lesions	Vascular lesions
8.		Benign keratoses like-lesions	Melanoma

Berdasarkan Tabel 4.9, ketujuh gambar *dermoscopy* diprediksi benar oleh model kecuali satu data *dermoscopy* yang diprediksi salah menjadi Melanoma. Kemudian, pengujian terhadap data gambar dermoscopy dari RS Sanglah, Bali (Sadeli Marrietta et al., 2021), yang dilakukan pada sistem klasifikasi berbasis *web* yang telah dibangun, ditunjukkan pada Gambar 4.16. Terlihat bahwa terdapat kesalahan klasifikasi pada gambar tersebut. Model memprediksi gambar sebagai Melanoma, padahal yang benar adalah Benign Keratoses-like Lesions. Kesalahan ini mungkin terjadi karena perbedaan warna dan pola pada gambar *dermoscopy* lesi kulit antara dataset penelitian dan data dari RS lokal, yang menyebabkan kesalahan klasifikasi.



**Gambar 4.16** Pengujian gambar *dermoscopy* dari RS Sanglah, Bali pada sistem klasifikasi berbasis *website*

## BAB 5

### PENUTUP

#### 5.1. Kesimpulan

Dari hasil implementasi yang dibahas di Bab 4, penulis dapat menarik beberapa kesimpulan, yaitu:

1. Pada penelitian ini, menggunakan model DenseNet121 dan *data balancing* dengan teknik SMOTE (*Synthetic Minority Oversampling Technique*) berhasil meningkatkan akurasi pengujian hingga 93.20% dengan *loss* sebesar 0.2341, yang dibandingkan dengan model DenseNet121 tanpa teknik *data balancing* mendapatkan akurasi pengujian sebesar 75.52% dengan *loss* sebesar 0.6994.
2. Model DenseNet121 dengan teknik SMOTE lebih unggul dari model DenseNet121 dengan teknik Augmentasi Data dalam hal penurunan *loss*. Pada pengujian DenseNet121 dengan teknik SMOTE mendapatkan *loss* sebesar 0.2341, sedangkan dengan teknik Augmentasi Data mendapatkan *loss* sebesar 0.2813.
3. Dengan penggunaan fungsi *callback* seperti *EarlyStopping* dapat menghentikan proses pelatihan model ketika metrik *val\_loss* tidak menunjukkan adanya peningkatan selama 10 *epochs* dan *ReduceLROnPlateau* dapat menurunkan *learning rate* model pada proses pelatihan ketika metrik *val\_loss* tidak menunjukkan adanya peningkatan selama 10 *epochs*.
4. Performa yang baik dari model menunjukkan bahwa teknik *data balancing* efektif dalam menangani ketidakseimbangan data.

## 5.2. Saran

Beberapa saran dan rekomendasi untuk pengembangan sistem selanjutnya yang diusulkan oleh penulis berdasarkan hasil penelitian ini, yaitu:

1. Penelitian ini menggunakan data berupa *dermoscopy image*, yang di mana untuk mendapatkan data tersebut harus melalui pemeriksaan laboratorium dengan alat dermatoskop. Karenanya, akan lebih optimal jika terdapat sistem yang memungkinkan pengguna mendeteksi penyakit lesi kulit dengan mempertimbangkan data gejala umum yang dirasakan pasien, sehingga pengguna tetap dapat menggunakan sistem tersebut tanpa harus memeriksa terlebih dahulu ke dokter dan mendapatkan hasil uji laboratorium.
2. Eksplorasi dalam penggunaan model CNN serta teknik *data balancing* lainnya dapat menghasilkan akurasi yang lebih baik.
3. Menambahkan fitur tambahan seperti riwayat dari gambar penyakit yang sudah dideteksi oleh pengguna dan fitur rekomendasi penanganan untuk setiap penyakit yang dideteksi.
4. Berdasarkan uji program terhadap data *dermoscopy* yang diambil dari Rumah Sakit (RS) Sanglah, Denpasar, Bali, sistem klasifikasi yang dibangun oleh penulis menghasilkan kesalahan prediksi. Hal ini disebabkan oleh perbedaan warna dan pola pada gambar *dermoscopy* lesi kulit antara dataset penelitian dan data dari RS lokal, yang menyebabkan kesalahan klasifikasi. Oleh karena itu, untuk penelitian lebih lanjut disarankan untuk mengambil dataset penelitian dari RS lokal.
5. Pengembangan aplikasi berbasis Android untuk *mobile* dapat meningkatkan aksesibilitas dan kenyamanan pengguna, dengan menghadirkan fitur kamera secara *real time*.

## DAFTAR PUSTAKA

- Alhudhaif, A., Almaslukh, B., Aseeri, A. O., Guler, O., & Polat, K. (2023). A novel nonlinear automated multi-class skin lesion detection system using soft-attention based convolutional neural networks. *Chaos, Solitons and Fractals*, 170. <https://doi.org/10.1016/j.chaos.2023.113409>
- Bichakjian, C., Armstrong, A., Baum, C., Bordeaux, J. S., Brown, M., Busam, K. J., Eisen, D. B., Iyengar, V., Lober, C., Margolis, D. J., Messina, J., Miller, A., Miller, S., Mostow, E., Mowad, C., Nehal, K., Schmitt-Burr, K., Sekulic, A., Storrs, P., ... Rodgers, P. (2018). Guidelines of care for the management of basal cell carcinoma. *Journal of the American Academy of Dermatology*, 78(3), 540–559. <https://doi.org/10.1016/j.jaad.2017.10.006>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2011). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- De Berker, D., McGregor, J. M., & Hughes, B. R. (2007). Guidelines for the management of actinic keratoses. In *British Journal of Dermatology* (Vol. 156, Issue 2, pp. 222–230). <https://doi.org/10.1111/j.1365-2133.2006.07692.x>
- Eftekhari, N., Masoudi, M., Pourreza, H., Shirazi, K. G., & Saeedi, E. (2017). *Microaneurysm Detection in Fundus Images Using a Two-step Convolutional Neural Networks*. <https://doi.org/10.48550/arXiv.1710.05191>
- Ghalejoogh, G. S., Kordy, H. M., & Ebrahimi, F. (2020). A hierarchical structure based on Stacking approach for skin lesion classification. *Expert Systems with Applications*, 145. <https://doi.org/10.1016/j.eswa.2019.113127>
- Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: an Overview*. <http://arxiv.org/abs/2008.05756>
- Gu, S., Pednekar, M., Slater, R., Gu, S. ;, & Pednekar, M. ; (2019). Improve Image Classification Using Data Augmentation and Neural Networks. In *SMU Data*

- Science Review* (Vol. 2, Issue 2).  
<https://scholar.smu.edu/datasciencereview/vol2/iss2/1>
- Hasib, K. M., Iqbal, M. S., Shah, F. M., Mahmud, J. Al, Popel, M. H., Showrov, M. I. H., Ahmed, S., & Rahman, O. (2020). A Survey of Methods for Managing the Classification and Solution of Data Imbalance Problem. *Journal of Computer Science*, 16(11), 1546–1557.  
<https://doi.org/10.3844/JCSSP.2020.1546.1557>
- Hatem, M. Q. (2022). Skin lesion classification system using a K-nearest neighbor algorithm. *Visual Computing for Industry, Biomedicine, and Art*, 5(1).  
<https://doi.org/10.1186/s42492-022-00103-6>
- Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). *Densely Connected Convolutional Networks*. <http://arxiv.org/abs/1608.06993>
- Jadhav, A., Mostafa, S. M., Elmannai, H., & Karim, F. K. (2022). An Empirical Assessment of Performance of Data Balancing Techniques in Classification Task. *Applied Sciences (Switzerland)*, 12(8).  
<https://doi.org/10.3390/app12083928>
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). *Machine learning and deep learning*. 31, 685–695. <https://doi.org/10.1007/s12525-021-00475-2>
- Joloudari, J. H., Marefat, A., Nematollahi, M. A., Oyelere, S. S., & Hussain, S. (2023). Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks. *Applied Sciences (Switzerland)*, 13(6).  
<https://doi.org/10.3390/app13064006>
- Liu, S., Yang, J., Agaian, S. S., & Yuan, C. (2021). Novel features for art movement classification of portrait paintings. *Image and Vision Computing*, 108.  
<https://doi.org/10.1016/j.imavis.2021.104121>
- Luba, M., Bangs, S., Mohler, A., & Stulberg, D. (2003). *Common benign skin tumors*. 67. <https://pubmed.ncbi.nlm.nih.gov/12613727/>
- Marsden, J. R., Newton-Bishop, J. A., Burrows, L., Cook, M., Corrie, P. G., Cox, N. H., Gore, M. E., Lorigan, P., MacKie, R., Nathan, P., Peach, H., Powell, B., & Walker, C. (2010). Revised U.K. guidelines for the management of



- cutaneous melanoma 2010. In *British Journal of Dermatology* (Vol. 163, Issue 2, pp. 238–256). <https://doi.org/10.1111/j.1365-2133.2010.09883.x>
- Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3), 563–597. <https://doi.org/10.1007/s10844-015-0368-1>
- O'gara, S., & McGuinness, K. (2019). *Comparing Data Augmentation Strategies for Deep Image Classification*. <https://doi.org/10.21427/148b-ar75>
- Qian, Y., Liang, Y. C., & Guan, R. C. (2014). Improving activated sludge classification based on imbalanced data. *Journal of Hydroinformatics*, 16(6), 1331–1342. <https://doi.org/10.2166/hydro.2014.123>
- Rahman, Z., Hossain, M. S., Islam, M. R., Hasan, M. M., & Hridhee, R. A. (2021). An approach for multiclass skin lesion classification based on ensemble learning. *Informatics in Medicine Unlocked*, 25. <https://doi.org/10.1016/j.imu.2021.100659>
- Sadeli Marrietta, Wardhana Made, Windari Martina, & Rahmawati Ana. (2021). *GAMBARAN KLINIS DAN FITUR DERMOSKOPI KERATOSIS SEBOROIK DI RUMAH SAKIT SANGLAH, DENPASAR*.
- Swana, E. F., Doorsamy, W., & Bokoro, P. (2022). Tomek Link and SMOTE Approaches for Machine Fault Classification with an Imbalanced Dataset. *Sensors*, 22(9). <https://doi.org/10.3390/s22093246>
- Tschandl, P., Rosendahl, C., & Kittler, H. (2018). Data descriptor: The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5. <https://doi.org/10.1038/sdata.2018.161>
- Varalakshmi, P., Aruna Devi, V., Ezhilarasi, M., & Sandhiya, N. (2021). Enhanced Dermatoscopic Skin Lesion Classification using Machine Learning Techniques. *2021 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2021*, 68–71. <https://doi.org/10.1109/WiSPNET51692.2021.9419466>

- Vargas, R., Mosavi, A., & Ruiz, L. (2017). *DEEP LEARNING: A REVIEW*.  
<https://doi.org/10.20944/preprints201810.0218.v1>.
- Wang, Y., Feng, Y., Zhang, L., Zhou, J. T., Liu, Y., Goh, R. S. M., & Zhen, L. (2022). Adversarial multimodal fusion with attention mechanism for skin lesion classification using clinical and dermoscopic images. *Medical Image Analysis*, 81. <https://doi.org/10.1016/j.media.2022.102535>
- Yakshit, Kaur, G., Kaur, V., Sharma, Y., & Bansal, V. (2022, December 24). Analyzing various Machine Learning Algorithms with SMOTE and ADASYN for Image Classification having Imbalanced Data. *Proceedings of 2022 IEEE International Conference on Current Development in Engineering and Technology, CCET* 2022. <https://doi.org/10.1109/CCET56606.2022.10080783>
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In *Insights into Imaging* (Vol. 9, Issue 4, pp. 611–629). Springer Verlag. <https://doi.org/10.1007/s13244-018-0639-9>
- Yang, Z., Sinnott, R. O., Bailey, J., & Ke, Q. (2023). A survey of automated data augmentation algorithms for deep learning-based image classification tasks. *Knowledge and Information Systems*, 65(7), 2805–2861. <https://doi.org/10.1007/s10115-023-01853-2>
- Zhang, W.J. & Yang, Guosheng & Lin, Yingzi & Ji, Chunli, & Gupta, M. (2018). *On Definition of Deep Learning*. <https://doi.org/10.23919/WAC.2018.8430387>

# LAMPIRAN

## LISTING PROGRAM

### 1. Model.ipynb

```
import pandas as pd
import numpy as np
import tensorflow as tf
import os
from glob import glob
from PIL import Image
import shutil
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import plot_model
from mixtend.plotting import plot_confusion_matrix

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv('/content/drive/MyDrive/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')

base_dir = '/content/drive/MyDrive/skin-cancer-mnist-ham10000'

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x for x in glob(os.path.join(base_dir, '**', '*.jpg'))}

size = (128, 128)

df['path'] = df['image_id'].map(imageid_path_dict.get)
df['image'] = df['path'].map(lambda x: np.asarray(Image.open(x).resize(size))/ 255)

df = df[['dx', 'image']]

df.head()

df.info()

df['image'].map(lambda x: x.shape).value_counts()

df['dx'].value_counts()

counts = pd.DataFrame(df['dx'].value_counts()).reset_index()
```

```

# visualize the distribution of images
import seaborn as sns

sns.set_style('whitegrid')
fig, axes = plt.subplots(figsize=(12,8))
ax = sns.countplot(x='dx', data=df, order=df['dx'].value_counts().index, hue='dx', palette='Paired', legend=False)
for container in ax.containers:
    ax.bar_label(container)
plt.title('Distribusi Jenis Lesi Kulit')
plt.xticks(rotation=45)
plt.show()

arr = df.iloc[1]['image']
#Image.fromarray(arr)
plt.imshow(arr)
plt.axis('off')
plt.show()

n_samples = 5
fig, m_axs = plt.subplots(7, n_samples, figsize = (12,12))
for n_axs, (type_name, type_rows) in zip(m_axs,
                                         df.sort_values(['dx']).groupby('dx')):
    n_axs[0].set_title(type_name)
    for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples, random_state=42).iterrows()):
        c_ax.imshow(c_row['image'])
        c_ax.axis('off')
# fig.savefig('category_samples.png', dpi=300)

labelEncoder = LabelEncoder()
df['label'] = labelEncoder.fit_transform(df['dx'])

x = np.asarray(df['image'].to_list())
y = df['label']

# Flatten each image into a 1D array
x_flat = np.asarray([img.flatten() for img in x])

from imblearn.over_sampling import SMOTE

oversample = SMOTE(random_state=42)
data, label = oversample.fit_resample(x_flat, y)

data_resaped = data.reshape(-1, 120, 120, 3)
print('Shape of data :', data_resaped.shape)

```

```

label = np.array(label)

# Ubah label_resampled menjadi objek pandas Series
label_series = pd.Series(label)

# Hitung jumlah data dari tiap label
label_counts = label_series.value_counts()

# Ubah hasil value_counts menjadi DataFrame
label_counts_df = label_counts.reset_index()

# Ganti nama kolom menjadi 'Label' dan 'Count'
label_counts_df.columns = ['Label', 'Count']

# Tampilkan hasil
print("Jumlah data dari tiap label:")
print(label_counts_df)

import matplotlib.pyplot as plt

# Dataframe label_counts_df
label_counts_df = label_counts.reset_index()
label_counts_df.columns = ['Label', 'Count']

# Plotting diagram batang
plt.figure(figsize=(10, 6))
plt.bar(label_counts_df['Label'], label_counts_df['Count'], color='skyblue')
plt.xlabel('Label')
plt.ylabel('Count')
plt.title('Jumlah data dari tiap labe setelah SMOTE')
plt.xticks(rotation=45)
plt.tight_layout()

# Tampilkan diagram
plt.show()

x_train, x_temp, y_train, y_temp = train_test_split(data_resaped, label, test_size=.30, random_state=42, shuffle=True)
x_val, x_test, y_val, y_test = train_test_split(x_temp, y_temp, test_size=.50, random_state=42, shuffle=True)

print(f'x_train shape: {x_train.shape}\nx_val shape: {x_val.shape}\nx_test shape: {x_test.shape}')
print('-'*20)
print(f'y_train shape: {y_train.shape}\ny_val shape: {y_val.shape}\ny_test shape: {y_test.shape}')

# Load the pre-trained DenseNet model without top layers
base_model = DenseNet121(weights='imagenet', include_top=False, input_shape=(120, 120, 3))

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(7, activation='softmax')(x)

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the layers in the base model
for layer in base_model.layers:
    layer.trainable = False

model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])

reduce_lr = ReduceLRonPlateau(monitor='val_loss', patience=10, factor=0.5, min_lr=0.00001)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True, verbose=1)

history = model.fit(x_train, y_train, epochs=60, batch_size=64, validation_data=(x_val, y_val), callbacks=[early_stopping, reduce_lr])

test_loss, test_accuracy = model.evaluate(x_test, y_test)

print('Test Accuracy:', test_accuracy)
print('Test Loss:', test_loss)

tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

epochs = range(1, len(tr_acc) + 1)

plt.figure(figsize=(14, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs, tr_loss, 'r', label='Train Loss')
plt.plot(epochs, val_loss, 'g', label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epochs')

```

```

plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, tr_acc, 'r', label='Train Accuracy')
plt.plot(epochs, val_acc, 'g', label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()

plot_model(model, show_shapes=True, show_layer_names=False, dpi=50)

predictions = model.predict(x_test)

cm = confusion_matrix(y_test, np.argmax(predictions, axis=1))

cm_plot_labels = ['akiec', 'bcc', 'bkl', 'df', 'mel', 'nv', 'vasc']

fig, ax = plot_confusion_matrix(conf_mat=cm, class_names=cm_plot_labels, colorbar=True)
plt.show()

report = classification_report(y_test, np.argmax(predictions, axis=1), target_names=cm_plot_labels)
print(report)

import matplotlib.pyplot as plt

# Fungsi untuk memplot gambar dengan label prediksi dan sebenarnya
def plot_images(images, true_labels, pred_labels, class_names, num_images=10):
    plt.figure(figsize=(20, 10))
    for i in range(num_images):
        plt.subplot(2, num_images // 2, i + 1)
        img = images[i]
        # Pastikan nilai piksel berada dalam rentang [0, 1]
        img = np.clip(img, 0, 1)
        plt.imshow(img)
        true_label = class_names[true_labels[i]]
        pred_label = class_names[np.argmax(pred_labels[i])]
        plt.title(f'True: {true_label}\nPred: {pred_label}')
        plt.axis('off')
    plt.show()

class_names = ['Actinic Keratoses', 'Basal Cell Carcinoma', 'Benign Keratosis-Like Lesions', 'Dermatofibroma', 'Melanoma', 'Melanocytic Nevi', 'Vascular Lesions']

# Plot 10 gambar pertama dari test set
plot_images(x_test, y_test, predictions, class_names, num_images=10)

# save model
model.save('/content/drive/MyDrive/model-skin4-smote.h5')

```

## 2. Main.py

```

from fastapi import FastAPI, File, UploadFile
from fastapi.middleware.cors import CORSMiddleware
from io import BytesIO
from PIL import Image
import numpy as np
import tensorflow as tf
import uvicorn
import h5py

app = FastAPI()

origins = [
    "http://localhost",
    "http://localhost:3000",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

MODEL = tf.keras.models.load_model(h5py.File('saved_model/model-skin4-smote.h5', 'r'))
CLASS_NAMES = ['Actinic Keratoses', 'Basal Cell Carcinoma', 'Benign Keratosis-Like Lesions', 'Dermatofibroma', 'Melanoma', 'Melanocytic Nevi', 'Vascular Lesions']

@app.get("/ping")
async def ping():
    return "Hello, I'm alive!"

def read_file_as_image(data) -> np.ndarray:
    image = Image.open(BytesIO(data))
    image = image.resize((120, 120)) # Mengubah ukuran gambar
    image = np.array(image)
    if image.shape[-1] == 4: # Jika gambar memiliki kanal alpha, buang kanal alpha
        image = image[..., :3]
    image = image / 255.0 # Normalisasi nilai pixel
    return image

@app.post("/predict")
async def predict(
    file: UploadFile = File(...)

```

```

}):
    image = read_file_as_image(await file.read())
    img_batch = np.expand_dims(image, 0)

    predictions = MODEL.predict(img_batch)

    predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
    confidence = np.max(predictions[0])
    return {
        'class': predicted_class,
        'confidence': float(confidence)
    }

if __name__ == "__main__":
    uvicorn.run(app, host="localhost", port=8080)

```

### 3. App.jsx

```

import React, { useState, useEffect } from 'react';
import { Navigation } from './components/navigation';
import { Header } from './components/header';
import { Lesions } from './components/lesions';
import { About } from './components/about';
import { ImageUpload } from './components/detect';
import jsonData from './data/data.json';
import SmoothScroll from 'smooth-scroll';
import './App.css';

export const scroll = new SmoothScroll('a[href*="#"]', {
    speed: 1000,
    speedAsDuration: true,
});

const App = () => {
    const [landingPageData, setLandingPageData] = useState({});
    useEffect(() => {
        setLandingPageData(jsonData);
    }, []);

    return (
        <div>
            <Navigation />
            <Header data={landingPageData.Header} />
            <About data={landingPageData.About} />
            <Lesions data={landingPageData.Lesions} />
            <ImageUpload />
        </div>
    );
};

export default App;

```