

TUGAS AKHIR - SS234862

KLASIFIKASI OBJEK PARIWISATA MELALUI UNGGAHAN FOTO DI MEDIA SOSIAL DENGAN METODE *PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK*

AKHMAD MIFTAKHUL ILMI

NRP 06211940000062

Dosen Pembimbing

Dr. Kartika Fithriasari, M.Si.

NIP 19691212 199303 2 002

Program Studi Sarjana Statistika

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2023



TUGAS AKHIR - SS234862

**KLASIFIKASI OBJEK PARIWISATA MELALUI
UNGGAHAN FOTO DI MEDIA SOSIAL DENGAN
METODE *PRE-TRAINED CONVOLUTIONAL NEURAL
NETWORK***

AKHMAD MIFTAKHUL ILMI

NRP 06211940000062

Dosen Pembimbing

Dr. Kartika Fithriasari, M.Si

NIP 19691212 199303 2 002

Program Studi Sarjana Statistika

Departemen Statistika

Fakultas Sains dan Analitika Data

Institut Teknologi Sepuluh Nopember

Surabaya

2023



FINAL PROJECT - KS184822

CLASSIFICATION OF TOURISM OBJECTS THROUGH UPLOADED PICTURES ON SOCIAL MEDIA USING PRE- TRAINED CONVOLUTIONAL NEURAL NETWORK METHOD

AKHMAD MIFTAKHUL ILMI

NRP 06211940000062

Advisor

Dr. Kartika Fithriasari, M.Si

NIP 19691212 199303 2 002

Undergraduate Study Program of Statistics

Department of Statistics

Faculty of Science and Data Analytics

Institut Teknologi Sepuluh Nopember

Surabaya

2023

LEMBAR PENGESAHAN

KLASIFIKASI OBJEK PARIWISATA MELALUI UNGGAHAN FOTO DI MEDIA SOSIAL DENGAN METODE *PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Statistika pada
Program Studi Sarjana Statistika
Departemen Statistika
Fakultas Sains dan Analitika Data
Institut Teknologi Sepuluh Nopember

Oleh: **Akhmad Miftakhul Ilmi**
NRP. 06211940000062

Disetujui Oleh:



Pembimbing:

1. Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002



Penguji:

1. Prof. Dr.rer.pol. Heri Kuswanto, M.Si
NIP. 19820326 200312 1 004
2. Widhianingsih Tintrim Dwi Ary, S.Si., M.Stat., Ph.D.
NPP. 2022199512001

(Halaman Ini Sengaja Dikosongkan)

APPROVAL SHEET

CLASSIFICATION OF TOURISM OBJECTS THROUGH UPLOADED PICTURES ON SOCIAL MEDIA USING PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK METHOD

FINAL PROJECT


Submitted to fulfill one of the requirements
for obtaining a degree in Bachelor of Statistics at
Undergraduate Program of Statistics
Departement of Statistics
Faculty of Science and Data Analytics
Institut Teknologi Sepuluh Nopember

by: **Akhmad Miftakhul Ilmi**
NRP. 06211940000062

Approved by:

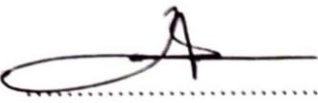
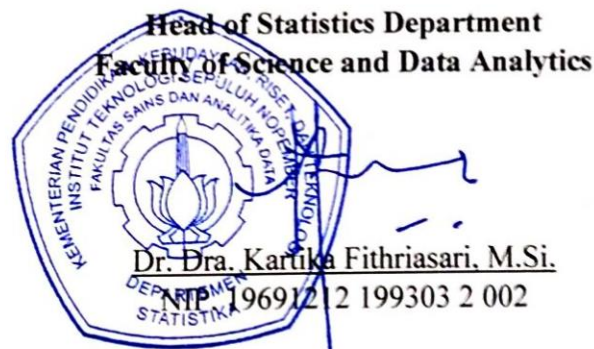

Advisor:

1. Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002



Examiners:

1. Prof. Dr.rer.pol. Heri Kuswanto, M.Si
NIP. 19820326 200312 1 004
2. Widhianingsih Tintrim Dwi Ary, S.Si., M.Stat., Ph.D.
NPP. 2022199512001

(Halaman Ini Sengaja Dikosongkan)

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa/NRP : Akhmad Miftakhul Ilmi/ 06211940000062
Departement : Statistika
Dosen Pembimbing : Dr. Dra. Kartika Fithriasari, M.Si.

Dengan ini menyatakan bahwa Tugas Akhir dengan judul “**Klasifikasi Objek Pariwisata Melalui Unggahan Foto di Media Sosial dengan Metode *Pre-trained Convolutional Neural Network***” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 01 Juli 2023

Mengetahui
Dosen Pembimbing

Mahasiswa



(Dr. Dra. Kartika Fithriasari, M.Si.)
NIP. 19691212 199303 2 002



(Akhmad Miftakhul Ilmi)
NRP. 06211940000062

(Halaman Ini Sengaja Dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

Student's Name/NRP : Akhmad Miftakhul Ilmi/ 06211940000062
Departement : Statistics
Advisor : Dr. Dra. Kartika Fithriasari, M.Si.

Hereby declare that the Final Project with the title of "**Classification of Torism Object Through Uploaded Pictures on Social Media Using Pre-trained Convolutional Neural Network Method**" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 01 July 2023

Acknowledged
Advisor



(Dr. Dra. Kartika Fithriasari, M.Si.)
NIP. 19691212 199303 2 002

Student



(Akhmad Miftakhul Ilmi)
NRP. 06211940000062

(Halaman Ini Sengaja Dikosongkan)

ABSTRAK

KLASIFIKASI OBJEK PARIWISATA MELALUI UNGGAHAN FOTO DI MEDIA SOSIAL DENGAN METODE *PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK*

Nama Mahasiswa / NRP : Akhmad Miftakhul Ilmi / 06211940000062
Departemen : Statistika FSAD - ITS
Dosen Pembimbing : Dr. Kartika Fithriasari, M.Si

Abstrak

Pariwisata merupakan salah satu sektor penting dalam aktivitas sosial dan ekonomi suatu negara. Saat ini sektor pariwisata erat kaitannya dengan media sosial, karena banyak wisatawan yang mengunggah foto berwisata maupun ulasan tempat wisata di akun sosial medianya. Promosi yang tepat sasaran pada media sosial merupakan salah satu strategi untuk mengembangkan sektor pariwisata, oleh karena itu perlu dibangun sistem rekomendasi objek pariwisata. Dalam membangun sistem rekomendasi khususnya yang berbasis *Fuzzy Rules*, umumnya perlu dilakukan klasifikasi objek pariwisata terlebih dahulu. Maka dari itu, pada penelitian ini akan dilakukan klasifikasi citra foto dari objek pariwisata menggunakan tiga jenis arsitektur *pre-trained model Convolutional Neural Network* (CNN) yaitu GoogLeNet, AlexNet, dan VGG16. *Training* pada 6000 data dilakukan pada ketiga jenis arsitektur *pre-trained model* CNN tersebut. Kemudian, masing-masing *pre-trained model* dilakukan *training* sebanyak lima kali, dengan kombinasi *hyperparameter* berbeda-beda untuk menemukan performa terbaik dari masing-masing model. Setelah dilakukan perbandingan, didapatkan model terbaik yaitu GoogLeNet dengan kombinasi *hyperparameter* meliputi penggunaan *layer inception block 5* dalam kondisi *unfreeze*, *learning rate* sebesar 0,0001, *batch size* 32, dan *dropout* 0,5. Model dengan kombinasi tersebut mencapai akurasi data training sebesar 0,9871, nilai *loss* data training sebesar 0,0631, akurasi data *validation* sebesar 0,9125, dan nilai *loss* data *validation* sebesar 0,2533. Selanjutnya model tersebut diaplikasikan pada 600 data *testing* dan menghasilkan akurasi sebesar 0,9467 dan nilai *loss* sebesar 0,1158. Model ini memiliki performa yang baik dalam mengklasifikasikan objek pariwisata pada data *test*, termasuk dalam konteks objek pariwisata di Indonesia.

Kata kunci: CNN, Klasifikasi, Pariwisata, *Pre-trained model*

(Halaman Ini Sengaja Dikosongkan)

ABSTRACT

CLASSIFICATION OF TOURISM OBJECTS THROUGH PHOTO UPLOADS ON SOCIAL MEDIA WITH PRE-TRAINED CONVOLUTIONAL NEURAL NETWORK METHOD

Student Name / NRP : Akhmad Miftakhul Ilmi / 06211940000062
Department : Statistika FSAD - ITS
Advisor : Dr. Kartika Fithriasari, M.Si

Abstract

Tourism is an important sector in the social and economic activities of a country. Currently, the tourism sector is closely related to social media, as many tourists upload photos of tourist destinations on their social media accounts. Targeted promotion on social media is one of the strategies to develop the tourism sector, therefore it is necessary to build a tourism object recommendation system. In building recommendation systems, especially those based on Fuzzy Rules, it is generally necessary to classify tourism objects first. Therefore, this research will classify image photos of tourism objects using three types of pre-trained Convolutional Neural Network (CNN) models: GoogLeNet, AlexNet, and VGG16. Training will be conducted on 6000 data using these three types of pre-trained CNN models. Then, each pre-trained model will be trained five times with different combinations of hyperparameters to find the best performance for each model. After comparison, the best model is found to be GoogLeNet with the combination of hyperparameters including the use of the inception block 5 layer in an unfreeze condition, a learning rate of 0.0001, a batch size of 32, and a dropout of 0.5. The model with this combination achieves a train accuracy of 0.9871, a train loss of 0.0631, a validation accuracy of 0.9125, and a validation loss of 0.2533. Subsequently, this model is applied to 600 testing data and achieves an accuracy of 0.9467 and loss of 0.1158. This model performs well in classifying tourism objects in the test data, including in the context of tourism objects in Indonesia.

Keywords: Classification, CNN, Pre-trained model, Tourism

(Halaman Ini Sengaja Dikosongkan)

KATA PENGANTAR

Alhamdulillah, puji syukur kehadiran Allah SWT yang telah melimpahkan nikmat serta karunia-Nya yang tak terhingga, sehingga penulis dapat menyelesaikan laporan Tugas Akhir dengan judul **“Klasifikasi Objek Pariwisata Melalui Unggahan Foto di Media Sosial dengan Metode Pre-trained Convolutional Neural Network”** dengan baik dan lancar.

Penulisan laporan Tugas Akhir ini tidak dapat diselesaikan secara lancar tanpa bantuan serta dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih dan penghargaan kepada:

1. Ibunda tercinta Susiani, ayah Suhariadi, adik tersayang Umi Milatusolikah, nenek tercinta Marmunah, serta segenap keluarga dan kerabat yang tak henti-hentinya memberikan dukungan dan do'a kepada penulis.
2. Ibu Dr. Kartika Fithriasari, M.Si selaku dosen pembimbing atas arahan, saran, serta bimbingannya selama penyusunan Tugas Akhir.
3. Prof Dr.rer.pol Heri Kuswanto selaku dosen wali dan dosen penguji pertama yang memberikan saran dan arahan tidak hanya dalam menulis Tugas Akhir, namun dalam seluruh proses pembelajaran di Departemen Statistika.
4. Ibu Widhianingsih Tintrim Dwi Ary, S.Si., M.Stat., Ph.D. selaku dosen penguji kedua yang memberikan saran dan masukan terkait penulisan Tugas Akhir.
5. Bapak dan Ibu dosen Departemen Statistika ITS yang telah memberikan ilmu yang bermanfaat bagi penulis
6. Sahabat penulis di grup “Bismillah” yang tersayang yaitu Aulia Kharis Rakhmasari serta Dede Yusuf P. Kuntaritas yang senantiasa mendampingi perjalanan penulis selama belajar di Departemen Statistika ITS.
7. Sahabat Penulis lainnya yaitu Mohammad Bagussurya Basuni, Rachmat Winardiansyah, M Zaim Husnun Niam, dan Dwi Purwanto yang senantiasa memberikan dukungan serta semangat dalam penyelesaian laporan Tugas Akhir.
8. Teman-teman Statistika ITS Angkatan 2019 “Abhista”, serta semua pihak yang turut membantu dalam pelaksanaan Tugas Akhir yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penulisan tugas akhir ini masih belum sempurna. Ketidak sempurnaan tersebut menjadi celah untuk pembelajaran dan pengembangan keilmuan kedepannya. Semoga Tugas Akhir ini bisa menjadi bahan referensi dan memberikan manfaat bagi pengembangan keilmuan statistika.

Surabaya, 01 Juli 2023

Penulis

(Halaman Ini Sengaja Dikosongkan)

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
STATEMENT OF ORIGINALITY	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xii
DAFTAR ISI	xv
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xix
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	3
BAB 2 TINJAUAN PUSTAKA	5
2.1 Pengolahan Citra Digital	5
2.2 <i>Data Preprocessing</i>	6
2.3 <i>Convolutional Neural Network (CNN)</i>	6
2.3.1 <i>Convolutional Layer</i>	7
2.3.2 <i>Pooling Layer</i>	7
2.3.3 <i>Fully Connected Layer</i>	8
2.4 Training pada Model CNN	9
2.5 Hyperparameter Tuning	10
2.6 Fungsi Aktivasi	10
2.6.1 Fungsi Aktivasi ReLU	10
2.6.2 Fungsi Aktivasi Softmax	11
2.7 <i>Batch Normalization</i>	12
2.8 <i>Learning Rate</i>	13
2.9 <i>Dropout</i>	14
2.10 <i>Epoch</i>	15
2.11 <i>Early Stopping</i>	15
2.12 <i>Adam Optimizer</i>	16
2.13 <i>Transfer Learning</i>	17
2.13.1 GoogLeNet	18
2.13.2 AlexNet	19
2.13.3 VGG16	20
2.14 Ketepatan Klasifikasi	21
2.15 Pariwisata	22
BAB 3 METODOLOGI	23
3.1 Sumber Data	23
3.2 Variabel Penelitian	23
3.3 Struktur Data	23

3.4	Langkah Analisis	24
3.5	Diagram Alir	25
BAB 4	HASIL DAN PEMBAHASAN	27
4.1	Data Preprocessing	27
4.2	Evaluasi Performa Model	29
4.2.1	Performa Model VGG16.....	29
4.2.2	Performa Model GoogLeNet.....	32
4.2.3	Performa Model AlexNet	34
4.3	Pemilihan Model dengan Performa Terbaik	36
4.4	Implementasi Model Terbaik pada Data Test	36
4.5	Analisis Prediksi yang Tidak Tepat terhadap Data Test.....	37
BAB 5	PENUTUP.....	45
5.1	Kesimpulan	45
5.2	Saran	45
DAFTAR PUSTAKA		47
LAMPIRAN.....		51
BIODATA PENULIS		65

DAFTAR GAMBAR

Gambar 2.1 Komposisi Citra RGB (Pramoditha, 2021).....	5
Gambar 2.2 Arsitektur <i>Convolutional Neural Network</i> (Tammina, 2019)	6
Gambar 2.3 Cara Kerja <i>Convolutional Layer</i> (Md Anwar Hossain et al., 2019).....	7
Gambar 2.4 Cara Kerja <i>Pooling Layer</i> (Md Anwar Hossain et al., 2019)	8
Gambar 2.5 <i>Fully Connected Layer</i> (Peneliti).....	9
Gambar 2.6 Proses <i>Training</i> pada CNN (Antunes et al., 2023)	9
Gambar 2.7 Fungsi Aktivasi ReLU (Agarap, 2018).....	11
Gambar 2.8 Efek Beberapa Nilai <i>Learning Rate</i> terhadap Performa Model (Alzubaidi et al., 2021).....	13
Gambar 2.9 <i>Neural Network</i> Tanpa <i>Dropout</i> (a) dan Dengan <i>Dropout</i> (b) (Srivastava et al., 2014).....	14
Gambar 2.10 Perbandingan Antara <i>Standard Network</i> (a) dan <i>Dropout Network</i> (b) (Srivastava et al., 2014).....	15
Gambar 2.11 Konsep <i>Transfer Learning</i> (Hosna et al., 2022).....	17
Gambar 2.12 <i>Fine-tuning</i> Parsial pada <i>Base Network</i> (Yosinski et al., 2014).....	18
Gambar 2.13 Arsitektur GoogLeNet (Szegedy et al., 2014)	19
Gambar 2.14 Arsitektur Alexnet (Krizhevsky et al., 2012).....	20
Gambar 2.15 Arsitektur VGG16 (Simonyan & Zisserman, 2014).....	20
Gambar 2.16 Confusion Matrix (Ali et al., 2019)	21
Gambar 3.1 Diagram Alir Penelitian.....	25
Gambar 4.1 Konversi <i>Color Channel</i> Gambar dari BGR (a) ke RGB (b)	27
Gambar 4.2 Histogram <i>Color Channel</i> RGB dari Satu Gambar pada Setiap Kategori.	28
Gambar 4.3 Contoh Hasil dari Proses <i>Resize</i> (a) Gambar Asli (b) Gambar <i>Resize</i>	28
Gambar 4.4 Proporsi Data <i>Training</i> dan Data <i>Validation</i>	29
Gambar 4.5 Akurasi Model VGG16 dengan Kombinasi Pengujian Keempat.....	31
Gambar 4.6 Nilai <i>Loss</i> Model VGG16 dengan Kombinasi Pengujian Keempat	31
Gambar 4.7 Akurasi Model GoogLeNet dengan Kombinasi Pengujian Kedua.....	33
Gambar 4.8 Nilai <i>Loss</i> Model GoogLeNet dengan Kombinasi Pengujian Kedua	33
Gambar 4.9 Akurasi Model AlexNet dengan Kombinasi Pengujian Pertama	35
Gambar 4.10 Nilai <i>Loss</i> Model AlexNet dengan Kombinasi Pengujian Pertama.....	35
Gambar 4.11 <i>Confusion Matrix</i> Prediksi Model GoogLeNet pada Data <i>Test</i>	37
Gambar 4.12 Contoh <i>Output</i> Prediksi Model GoogLeNet pada Data <i>Test</i>	37
Gambar 4.13 Kesalahan Klasifikasi Kategori <i>Beach</i> menjadi <i>Forest</i>	38
Gambar 4.14 Kesalahan Klasifikasi Kategori <i>Beach</i> menjadi <i>Mountain</i>	39
Gambar 4.15 Kesalahan Klasifikasi Kategori <i>Beach</i> menjadi <i>Waterfall</i>	39
Gambar 4.16 Kesalahan Klasifikasi Kategori <i>Forest</i> menjadi <i>Mountain</i>	40
Gambar 4.17 Kesalahan Klasifikasi Kategori <i>Mountain</i> menjadi <i>Forest</i>	41
Gambar 4.18 Kesalahan Klasifikasi Kategori <i>Mountain</i> menjadi <i>Beach</i>	41
Gambar 4.19 Kesalahan Klasifikasi Kategori <i>Museum</i> menjadi <i>Restaurant</i>	42
Gambar 4.20 Kesalahan Klasifikasi Kategori <i>Restaurant</i> menjadi <i>Museum</i>	42
Gambar 4.21 Kesalahan Klasifikasi Kategori <i>Waterfall</i> menjadi <i>Forest</i>	43
Gambar 4.22 Kesalahan Klasifikasi Kategori <i>Waterfall</i> menjadi <i>Mountain</i>	44

(Halaman Ini Sengaja Dikosongkan)

DAFTAR TABEL

Tabel 3.1 Jenis Objek Pariwisata	23
Tabel 3.2 Variabel Penelitian	23
Tabel 3.3 Struktur Data	24
Tabel 3.4 Struktur Data yang akan Diproses	24
Tabel 4.1 Arsitektur VGG16 dari Tensorflow	30
Tabel 4.2 Performa Berbagai Kombinasi Pengujian untuk VGG16	30
Tabel 4.3 Arsitektur GoogLeNet dari <i>Library</i> PyTorch.....	32
Tabel 4.4 Performa Berbagai Kombinasi Pengujian untuk GoogLeNet	32
Tabel 4.5 Arsitektur AlexNet dari <i>Library</i> PyTorch	34
Tabel 4.6 Performa Berbagai Kombinasi Pengujian untuk AlexNet	34
Tabel 4.7 Performa Terbaik dari Masing-masing Model	36

(Halaman Ini Sengaja Dikosongkan)

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pariwisata merupakan sektor yang memiliki peran penting dalam aktivitas sosial dan ekonomi suatu negara, dikarenakan sektor ini mampu menciptakan berbagai macam bisnis yang dapat berdampak pada perluasan lapangan kerja. Di Indonesia, kontribusi sektor pariwisata terhadap Produk Domestik Bruto (PDB) meningkat setiap tahunnya, data terbaru menunjukkan bahwa kontribusi sektor pariwisata terhadap PDB Indonesia hingga kuartal III 2022 yaitu sebesar 3,6%, jumlah tersebut meningkat dari tahun 2021 sebesar 2,4% (MENPAN RB, 2022). Sektor pariwisata yang berbasis pengalaman memerlukan pemahaman yang mendalam terkait pengelolaannya, salah satunya yaitu berkaitan dengan preferensi serta kebiasaan wisatawan, pemahaman yang mendalam pada sektor pariwisata akan berdampak pada efisiensi manajemen dan pengelolaan sumber daya yang maksimal (Xiang & Fesenmaier Editors, 2017).

Saat ini sektor pariwisata erat kaitannya dengan media sosial, karena banyak wisatawan yang mengunggah foto mengenai objek wisata serta ulasan dari tempat wisata yang telah dikunjungi di akun sosial medianya, seperti contohnya yaitu Instagram. Maraknya wisatawan yang menggunakan akun media sosialnya untuk mengunggah foto berwisatanya akhirnya juga memicu pemerintah untuk berinovasi dalam memanfaatkan fenomena ini demi kemajuan sektor pariwisata, salah satunya yaitu melakukan promosi tempat wisata melalui media sosial. Namun terkadang meskipun promosi mengenai tempat wisata sudah dilakukan oleh pemerintah, wisatawan masih tidak mengetahui mengenai tempat wisata yang dipromosikan tersebut, walaupun sebenarnya tempat wisata tersebut sesuai dengan preferensinya. Oleh karena itu diperlukan suatu sistem rekomendasi untuk wisatawan pengguna media sosial yang berkaitan dengan objek pariwisata. Sistem rekomendasi tempat wisata bertujuan untuk mencocokkan karakteristik objek wisata dengan preferensi wisatawan. Penyusunan sistem rekomendasi dapat menggunakan beberapa pendekatan. Pendekatan yang digunakan dalam menyusun sistem rekomendasi dapat terbagi menjadi beberapa metode yaitu *Collaborative Filtering* (CF), *Content Based* (CB), *Hybrid Methods* (CF dan CB), dan metode *Deep Learning*. Namun, dalam membangun sistem rekomendasi pariwisata diperlukan klasifikasi data citra foto objek pariwisata dari wisatawan, terutama sistem rekomendasi yang berbasis *Fuzzy Rules*. Hasil klasifikasi tersebut nantinya akan menjadi basis dari pembentukan sistem rekomendasi pariwisata (Brusch, 2022).

Klasifikasi data citra foto dapat dilakukan menggunakan beberapa metode. Salah satu metode yang sering digunakan untuk melakukan klasifikasi data tidak terstruktur seperti data citra foto yaitu permodelan dengan *Machine Learning* (ML). Klasifikasi citra foto dengan permodelan ML dapat terbagi lagi menjadi beberapa metode seperti *Artificial Neural Network* (ANN), *Naïve Bayes Classifier*, *K-Nearest Neighbor*, *Support Vector Machine* (SVM), *Decision Tree*, *Fuzzy Measure*, *Multi-Layered Perceptron* (MLP), dll. (Ponnusamy, 2017). Pada penelitian ini akan digunakan salah satu model klasifikasi citra foto yang merupakan pengembangan dari metode MLP yaitu model *Convolutional Neural Network* (CNN). Metode CNN dipilih karena memiliki keunggulan yaitu dapat mendeteksi fitur-fitur penting dari suatu citra foto secara otomatis tanpa pengawasan manusia secara terus menerus. Selain itu, metode CNN juga efisien dalam melakukan proses komputasinya, sehingga proses klasifikasi dapat dilakukan dengan lebih cepat tanpa menggunakan sumber daya yang banyak (Alzubaidi et al., 2021).

Penelitian terkait penggunaan metode *Machine Learning* untuk klasifikasi data citra foto pada sektor pariwisata pernah dilakukan oleh (Brusch, 2022), dimana pada penelitian tersebut digunakan beberapa metode diantaranya SVM, *Self-trained CNN*, dan *Pre-trained CNN* untuk menjadi basis dalam menentukan preferensi gaya berwisata pengguna menggunakan metode *Fuzzy Cluster Algorithm*. Pada penelitian tersebut digunakan data dari 80 anggota komunitas wisatawan pada platform TripAdvisor. Berdasarkan penelitian tersebut didapatkan klasifikasi citra foto dengan performa masing-masing model diukur menggunakan akurasi dengan hasil yaitu, SVM sebesar 0,827 kemudian *Self-trained CNN* sebesar 0,844 dan *Pre-trained CNN* sebesar 0,938. Selain itu, penelitian oleh (Ramadijanti et al., 2022) juga memanfaatkan metode *Pre-trained CNN* berupa VGG-16 dan ResNet untuk melakukan klasifikasi citra foto objek pariwisata yang terdapat pada platform Google Images. Hasil penelitian tersebut menunjukkan bahwa kedua model *Pre-trained CNN* tersebut memiliki hasil akurasi yang tinggi. Model VGG16 memiliki akurasi sebesar 0,985 sedangkan model ResNet memiliki akurasi sebesar 0,978. Berdasarkan penelitian sebelumnya, maka penelitian ini akan digunakan metode ML yaitu berupa *pre-trained* model CNN yang terdiri dari GoogLeNet, Alexnet, dan VGG16. Ketiga arsitektur *pre-trained* model CNN tersebut dipilih karena ketiganya sering digunakan dalam kasus klasifikasi citra foto. Selain itu, ketiga jenis *pre-trained* model tersebut juga memiliki performa yang baik untuk berbagai kasus klasifikasi citra gambar, seperti contohnya yaitu klasifikasi pada dataset ImageNet. *Pre-trained* model memiliki keunggulan yaitu dapat mencapai performa optimal yang lebih cepat dibandingkan dengan model yang dibangun sendiri dari nol (Kurama, 2020). Pada penelitian ini akan dibandingkan performa dari masing-masing model untuk mendapatkan model dengan performa terbaik dalam melakukan klasifikasi objek pariwisata.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah diuraikan, dapat dirumuskan permasalahan pada penelitian ini yaitu, dalam membangun suatu sistem rekomendasi diperlukan model machine learning untuk melakukan klasifikasi citra foto objek pariwisata. Sehingga perlu didapatkan model klasifikasi citra foto objek pariwisata menggunakan metode *Pre-trained CNN* dengan performa yang terbaik.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini yaitu:

1. Model ML dalam melakukan klasifikasi yaitu GoogLeNet, Alexnet, dan VGG16 *Pre-trained Convolutional Neural Network* (CNN)
2. Kategori objek pariwisata yang terdapat pada penelitian ini terdiri dari enam kategori yaitu Pantai, Hutan, Gunung, Air Terjun, Restoran, dan Museum.

1.4 Tujuan

Berdasarkan rumusan masalah tersebut, maka tujuan dari penelitian ini yaitu sebagai berikut

1. Mendapatkan model-model klasifikasi data foto objek pariwisata dengan metode *Pre-trained CNN*

2. Mengetahui model *Pre-trained* CNN dengan performa terbaik untuk klasifikasi data foto objek pariwisata sebagai basis dalam membangun sistem rekomendasi objek pariwisata

1.5 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat untuk beberapa pihak yaitu sebagai berikut

1. Bagi peneliti, dapat memberikan wawasan mengenai metode *Convolutional Neural Network* (CNN) serta aplikasinya dalam klasifikasi citra foto objek pariwisata
2. Bagi pihak yang bergerak pada sektor pariwisata, dapat memberikan pengetahuan tambahan dalam pemanfaatan klasifikasi citra foto objek pariwisata dalam sistem rekomendasi agar promosi objek pariwisata menjadi lebih tepat

(Halaman Ini Sengaja Dikosongkan)

BAB 2 TINJAUAN PUSTAKA

Bab ini menjelaskan mengenai tinjauan pustaka yang digunakan dalam melaksanakan penelitian yang meliputi Pengolahan Citra Digital, *Data Preprocessing*, *Convolutional Neural Network* (CNN), Fungsi Aktivasi, *Batch Normalization*, *Learning Rate*, *Dropout*, *Epochs*, *Transfer Learning*, Ketepatan Klasifikasi, dan Pariwisata.

2.1 Pengolahan Citra Digital

Citra digital merupakan array yang terdiri dari nilai-nilai yang direpresentasikan dengan suatu bit. Pengolahan citra secara digital secara umum yaitu berupa pemrosesan gambar dengan bantuan computer. Secara lebih detail, pengolahan citra digital dapat diartikan sebagai pengolahan data dua dimensi. Sebuah citra dapat didefinisikan sebagai suatu fungsi berukuran U baris dan V kolom, dimana u dan v menggambarkan koordinat spasial dan x pada titik koordinat (u,v) menggambarkan intensitas keabuan citra pada titik tersebut (Kusumanto & Tompunu, 2011). Citra digital dapat dituliskan pada Persamaan (2.1).

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,V} \\ x_{2,1} & x_{2,2} & \dots & x_{2,V} \\ \vdots & \vdots & \ddots & \vdots \\ x_{U,1} & x_{U,2} & \dots & x_{U,V} \end{bmatrix} \quad (2.1)$$

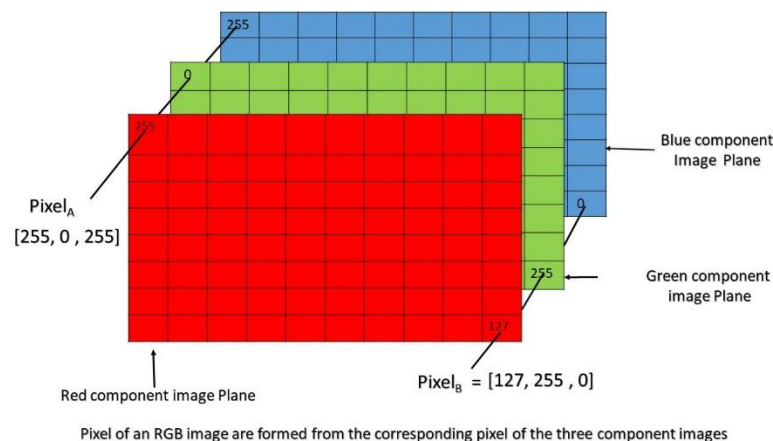
dimana,

U : Banyaknya *pixel* baris pada matriks

V : Banyaknya *pixel* kolom pada matriks

$x_{u,v}$: Tingkat warna dari citra

Suatu citra digital 8-bit nilai $x_{u,v}$ setara dengan $2^8 = 256$ tingkat *gray level* dengan 0 menggambarkan warna hitam dan 255 menggambarkan warna putih. Salah satu jenis citra digital yang sering digunakan yaitu citra RGB (*Red, Green, Blue*). Pada jenis citra RGB, setiap *pixel* pada citra warna mewakili warna yang merupakan kombinasi dari ketiga warna dasar merah, hijau, dan biru seperti yang diilustrasikan pada **Gambar 2.1**. Setiap titik *pixel* pada citra warna membutuhkan data sebesar 3 bit. Warna dasar pada citra RGB memiliki intensitas tertentu dengan nilai minimum nol (0) dan nilai maksimum 255 (8 bit) (Pramoditha, 2021).



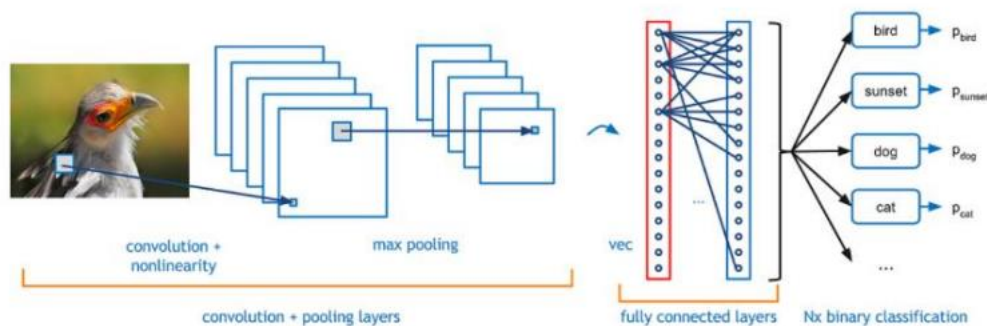
Gambar 2.1 Komposisi Citra RGB (Pramoditha, 2021)

2.2 Data Preprocessing

Data preprocessing merupakan suatu proses untuk mempersiapkan data untuk pemodelan *machine learning* agar memudahkan dalam proses komputasi dan analisisnya. Data yang akan digunakan untuk pemodelan khususnya data gambar, seringkali tidak beraturan formatnya dikarenakan data dapat berasal dari sumber yang berbeda-beda. Oleh karena itu, data perlu dinormalisasi dan dibersihkan sebelum memasuki proses pemodelan. Pada penelitian ini, *data preprocessing* yang akan digunakan yaitu *image normalization* dan *image standardizing*. *Image normalization* merupakan proses normalisasi rentang dari intensitas piksel suatu gambar kedalam suatu rentang yang sudah ditentukan, umumnya antara rentang (0,1) atau (-1,1). *Image normalization* berguna agar *learning rate* yang digunakan dapat terstandarkan untuk setiap gambar. Kemudian *image standardizing* merupakan suatu metode untuk mengatur gambar agar memiliki panjang dan lebar piksel yang sama. Hal tersebut bertujuan agar gambar dapat diproses pada model, karena setiap arsitektur model memiliki *input* ukuran piksel gambar yang spesifik (Koo & Cha, 2017).

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) merupakan suatu metode yang termasuk dalam *Deep Neural Network*. CNN dikembangkan dari metode *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data seperti citra gambar. CNN terbentuk oleh lapisan-lapisan (*layer*) neuron yang memiliki berat (*weight*) serta bias tiap *layer* yang dapat disesuaikan. CNN pada umumnya terdiri atas tiga jenis *layer* neuron yaitu, *input layer*, *hidden layer*, serta *output layer* seperti yang diilustrasikan pada **Gambar 2.2**.



Gambar 2.2 Arsitektur *Convolutional Neural Network* (Tammina, 2019)

Hidden layer pada CNN umumnya terdiri atas tiga jenis yaitu, *convolutional layer*, *pooling layer*, dan *fully connected layer* (Tammina, 2019). Pada penelitian ini penggunaan model CNN bertujuan untuk mengklasifikasikan gambar objek pariwisata. Proses membangun model untuk klasifikasi tersebut secara umum diawali dengan melakukan *input* data berbagai kategori objek pariwisata. Kemudian, gambar melewati filter-filter (kernel) pada *convolutional layer* untuk dilakukan deteksi fitur-fitur pada gambar seperti tepi dan bentuk objek pariwisata. Selanjutnya, gambar melewati *pooling layer* untuk dilakukan reduksi ukuran output dari *convolutional layer* serta melakukan ekstraksi fitur-fitur penting dari gambar. Lalu, gambar melewati *fully connected layer* untuk dilakukan klasifikasi gambar kedalam kategori objek pariwisata yang sudah ditentukan. Terakhir, gambar melalui *output layer* yang terdiri dari 6 neuron sesuai dengan kategori objek pariwisata yang sudah ditentukan pada penelitian ini.

Neuron dengan probabilitas yang paling tinggi kemudian dipilih menjadi prediksi kelas atau kategori untuk gambar yang sudah diinputkan (Li et al., 2017).

2.3.1 Convolutional Layer

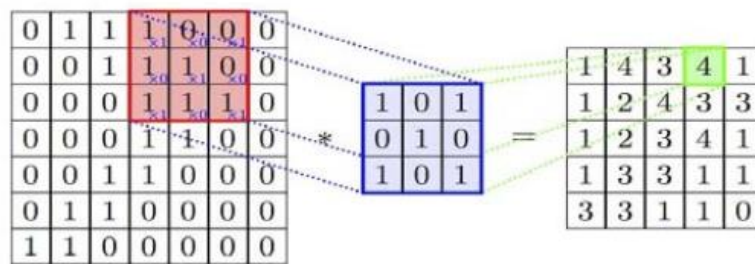
Convolutional layer merupakan inti dari arsitektur CNN, dimana pada layer ini terdiri atas k filter (kernel) yang selalu di-update dalam proses *learning*. Setiap filter pada *convolutional layer* memiliki panjang dan lebar tertentu, dimana umumnya kernel berbentuk persegi. Kernel-kernel ini akan bergeser diatas daerah *input*, kemudian melakukan *dot-product* yang bertujuan untuk menghasilkan *output* yang disebut sebagai *feature map* (Md Anwar Hossain et al., 2019). *Convolutional layer* dapat digambarkan dengan Persamaan (2.2).

$$c_{p^*, q^*, r^*}^{(layer)} = f(B^{(layer)} + \sum_{i=1}^l \sum_{j=1}^m \sum_{k=1}^n w_{i,j,k}^{(layer)} x_{p^*+i-1, q^*+j-1, k}^{(layer-1)}) \quad (2.2)$$

dimana,

- $c_{p^*, q^*, r^*}^{(layer)}$: *output convolutional layer* baris ke- p^* , kolom ke- q^* , *color channel* ke- r^*
- $B^{(layer)}$: nilai *bias* atau simpangan pada *feature map convolutional layer*
- l : panjang sisi *kernel*
- m : lebar sisi *kernel*
- n : *color channel*
- $x_{p^*+i-1, q^*+j-1, k}^{(layer-1)}$: *input* dari *layer* sebelumnya
- $w_{i,j,k}^{(layer)}$: *weight* dari *convolution kernel*
- p^* : lebar *feature map*
- q^* : panjang *feature map*
- r^* : *color channel feature map*
- f : *activation function*

selanjutnya, untuk cara kerja *convolutional layer* secara umum dapat diilustrasikan seperti pada **Gambar 2.3** berikut.



Gambar 2.3 Cara Kerja *Convolutional Layer* (Md Anwar Hossain et al., 2019)

2.3.2 Pooling Layer

Pooling layer dapat ditambahkan setelah *convolution layer*, *pooling layer* digunakan untuk mereduksi jumlah parameter input dengan cara *down sampling*. *Down sampling* dilakukan dengan tujuan mempertahankan informasi-informasi penting dari *output*. *Down sampling* berguna untuk mempercepat proses komputasi, dikarenakan parameter yang di-update menjadi lebih sedikit. Beberapa metode *pooling* yang dapat digunakan yaitu *max pooling* dan *average pooling* (Md Anwar Hossain et al., 2019). *Pooling layer* dengan metode *max pooling* dapat digambarkan dengan Persamaan (2.3).

$$d_{p^*, q^*, r^*}^{(layer+1)} = \max_{i,j} \{ c_{p^*+i-1, q^*+j-1, k}^{(layer)} \} \quad (2.3)$$

dimana,

$d_{p^*, q^*, r^*}^{(layer+1)}$: output pooling layer

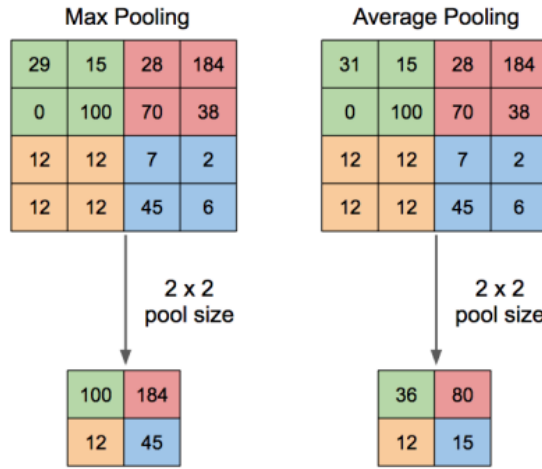
$c_{p^*+i-1, q^*+j-1, k}^{(layer)}$: input dari layer sebelumnya

p^* : lebar feature map

q^* : panjang feature map

r^* : indeks color channel

selanjutnya, untuk cara kerja pooling layer secara umum dapat diilustrasikan seperti pada Gambar 2.4 berikut.



Gambar 2.4 Cara Kerja Pooling Layer (Md Anwar Hossain et al., 2019)

2.3.3 Fully Connected Layer

Fully connected layer merupakan layer terakhir dalam arsitektur CNN yang dapat tersusun dari beberapa layer. Output dari pooling layer akan dilanjutkan pada layer ini dengan persyaratan yaitu dilakukan transformasi data menjadi satu dimensi (Md Anwar Hossain et al., 2019). Sebelum output dari layer sebelumnya diproses kedalam fully connected layer, dilakukan hasil output dari layer sebelumnya di lakukan flattened terlebih dahulu menjadi vektor 1 dimensi menggunakan Persamaan (2.4) berikut.

$$g = \text{reshape}(d_{p^*, q^*, r^*}^{(layer+1)}, (D,)) \quad (2.4)$$

dimana, D merupakan vektor dengan dimensi $D = p^* \times q^* \times 1$, dan $d_{p^*, q^*, r^*}^{(layer+1)}$ merupakan output dari layer sebelumnya. Setelah dilakukan flatten, maka fully connected layer dapat dihitung dengan Persamaan (2.5) berikut.

$$y_t = f\left(\sum_{s=1}^S w_{t,s} g_s + B_t\right) \quad (2.5)$$

dimana,

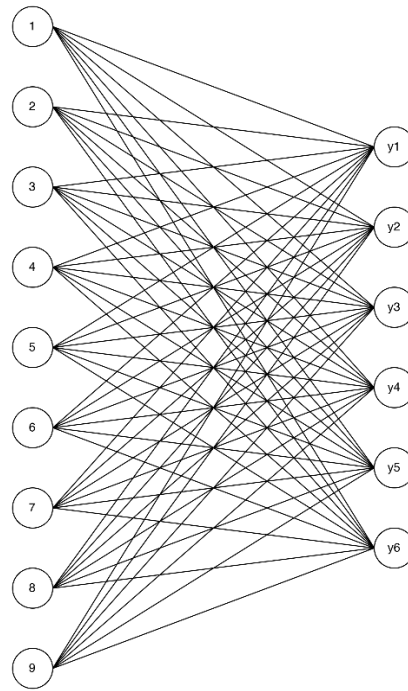
y_t : output fully connected layer pada output neuron ke- t

B_t : nilai bias atau simpangan dari setiap neuron ke- t fully connected layer

$w_{t,s}$: bobot fully connected layer untuk setiap neuron ke- t input ke- s

g_s : *input* ke- s dari *layer* sebelumnya
 s : vektor *input* dari *layer* sebelumnya
 t : vektor *output* dari *fully connected layer*
 f : *activation function*

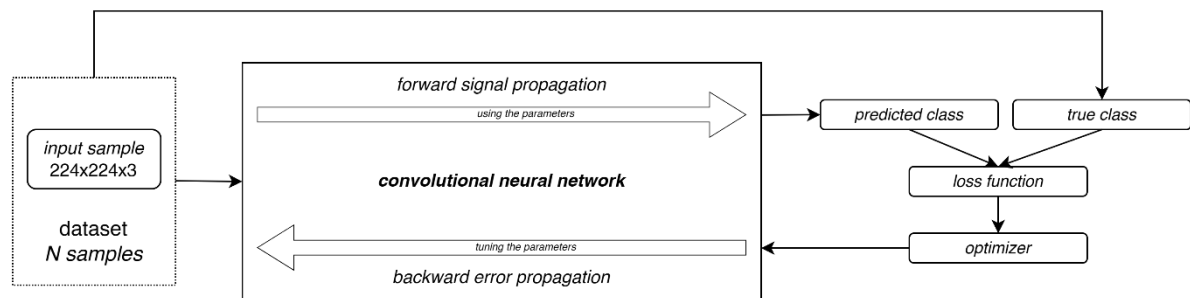
Neuron pada *fully connected layer* memiliki koneksi dari semua neuron di *layer* sebelumnya maupun neuron di *layer* berikutnya (Tammina, 2019). Ilustrasi *fully connected layer* dapat dilihat pada **Gambar 2.5**.



Gambar 2.5 Fully Connected Layer (Peneliti)

2.4 Training pada Model CNN

Proses training pada model *neural network* melibatkan penetapan bobot setiap neuron, sehingga ketika terdapat suatu *input* yang diberikan pada neuron tersebut, maka akan didapatkan *output* yang diharapkan. Proses *training* terjadi dalam dua tahap yaitu *forward signal propagation* dan *backward error propagation* (Antunes et al., 2023). Kedua tahapan tersebut dapat diilustrasikan pada **Gambar 2.6** berikut.



Gambar 2.6 Proses Training pada CNN (Antunes et al., 2023)

Pada tahap *forward signal propagation*, input data berupa gambar dilewatkan melalui beberapa *layer* pada model CNN. Setiap *layer* melakukan operasi seperti *convolution*, *activation*, dan *pooling*. Dari hasil operasi setiap *layer* tersebut akan didapatkan *output* berupa prediksi kelas dari gambar yang diproses. Setelah proses *forward signal propagation* sudah selesai, selanjutnya akan dilakukan proses *backward error propagation (backpropagation)*. *Backpropagation* diawali dengan perbandingan hasil dari *output* prediksi kelas model CNN dengan *ground truth label* (label sebenarnya). Proses perbandingan ini dibantu dengan *loss function*, dalam penelitian ini yaitu digunakan *categorical entropy* untuk mengkuantifikasikan perbedaan antara hasil prediksi dengan label yang sebenarnya. Setelah itu, dilakukan penghitungan gradien dari *loss function* terhadap bobot dan bias dari setiap neuron. Gradien tersebut menunjukkan bagaimana bobot dari setiap neuron memberikan efek terhadap nilai *loss* secara keseluruhan. Selanjutnya, dilakukan *update* parameter neuron dengan bantuan algoritma *optimizer*, yang dalam penelitian ini digunakan *Adam optimizer*. Proses *forward signal propagation* dan *backpropagation* ini diulang sebanyak *epoch* yang ditentukan oleh peneliti (Antunes et al., 2023).

2.5 Hyperparameter Tuning

Hyperparameter merupakan variabel yang berkaitan dengan struktur jaringan maupun proses *training* model CNN diluar dari data. Beberapa *hyperparameter* yang digunakan pada *training* model CNN yaitu algoritma *optimizer*, *learning rate*, *batch size*, dan *dropout value*. Agar didapatkan model dengan performa yang terbaik, maka perlu dilakukan *Hyperparameter tuning*. *Hyperparameter* tuning bertujuan untuk mendapatkan nilai *hyperparameter* yang optimal untuk masing-masing *pre-trained* model (Aszemi & Dominic, 2019). *Hyperparameter* pertama yang dilakukan pengaturan yaitu *learning rate*, yang merupakan *hyperparameter* untuk mengontrol seberapa cepat atau lambat model melakukan *training*. Pada penelitian ini *learning rate* ditentukan antara salah satu dari dua nilai yaitu 0,0001 dan 0,00001. Kemudian, *hyperparameter* selanjutnya yaitu *batch size*. *Batch size* merupakan jumlah *sampling* data yang digunakan dalam proses *training* dalam satu waktu. Pada penelitian ini digunakan dua macam *batch size* yaitu 32 dan 64. *Hyperparameter* yang selanjutnya dilakukan pengaturan yaitu *dropout*. *Dropout* bermanfaat untuk mencegah *overfitting* dengan mengatur beberapa neuron dalam *layer* menjadi bernilai 0. Pada penelitian ini digunakan dua macam nilai *dropout* yaitu 0,5 dan 0,3 (Aszemi & Dominic, 2019).

2.6 Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang bertujuan untuk memperkenalkan operasi non-linearitas pada *layer* CNN. Secara sederhana dapat dikatakan bahwa fungsi aktivasi bertugas melakukan regulasi apakah suatu neuron harus diaktifkan ataukah tidak. Sehingga nantinya fungsi aktivasi akan menentukan apakah *input* dari suatu neuron terhadap jaringan *neural network* penting ataukah tidak dalam melakukan proses prediksi (Dubey et al., 2021). Pada penelitian ini, akan digunakan dua jenis fungsi aktivasi yaitu fungsi aktivasi ReLu dan fungsi aktivasi Softmax.

2.6.1 Fungsi Aktivasi ReLU

Fungsi aktivasi ReLU (*Rectified Linear Unit*) merupakan fungsi aktivasi yang memiliki gambaran seperti fungsi linier, namun ReLU memiliki fungsi turunan sehingga memungkinkan model untuk melakukan *backpropagation* dan secara simultan membuat proses komputasi lebih

efisien. Hal tersebut dikarenakan, fungsi aktivasi ReLU tidak mengaktifkan semua neuron pada model secara bersamaan. Fungsi aktivasi ReLU dapat digambarkan dengan Persamaan (2.6).

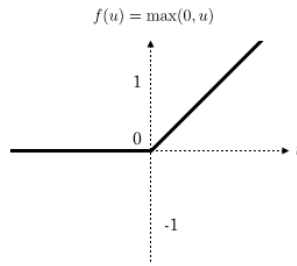
$$f(x) = \max(0, x) \quad (2.6)$$

dimana,

x : data *input*

$f(x)$: *output* fungsi aktivasi ReLU

Ukuran ReLU mengikuti output dari *layer* sebelumnya. Apabila ReLU diterapkan pada *convolutional layer* maka akan berbentuk matriks. Sedangkan jika fungsi ReLU diterapkan pada *fully connected layer*, maka akan terbentuk vektor dengan ukuran yang serupa. ReLU merupakan salah satu fungsi aktivasi dimana pada *layer* ReLU akan diaplikasikan fungsi *non-linear* pada *output* di *layer* sebelumnya. *Layer* ReLU dapat mempercepat konvergensi pada proses training di CNN (Agarap, 2018). Fungsi aktivasi ReLU diilustrasikan pada **Gambar 2.7** berikut.



Gambar 2.7 Fungsi Aktivasi ReLU (Agarap, 2018)

Gambar 2.7 menunjukkan bahwa fungsi aktivasi ReLU akan menonaktifkan neuron yang memiliki *output* dari transformasi linear yang kurang dari 0, sedangkan untuk *output* yang lebih dari 0, maka neuron akan diaktifkan dengan nilai yang sama dengan hasil *output*.

2.6.2 Fungsi Aktivasi Softmax

Softmax merupakan fungsi matematika yang digunakan pada *output layer* arsitektur CNN untuk melakukan klasifikasi *multi-class*. Fungsi aktivasi Softmax mengambil vektor skor sebagai *input* dan kemudian dilakukan normalisasi menjadi distribusi probabilitas dari kelas-kelas yang sudah ditentukan (Sharma et al., 2020). Fungsi aktivasi Softmax secara matematis dapat dituliskan seperti pada Persamaan (2.7).

$$f_t = \frac{e^{\rho_t^{(s)}}}{\sum_{s=1}^S e^{\rho_t^{(s)}}} \quad (2.7)$$

dimana, $\rho_t^{(s)} = \sum_{s=1}^S w_{t,s} g_s + B_t$,

keterangan:

- f_t : nilai probabilitas prediksi kelas ke- t
- $\rho_t^{(s)}$: jumlah perkalian *input* yang terboboti pada kelas h dan sebanyak s *input* (neuron) *layer* sebelumnya
- t : $(1, 2, \dots, T)$, dimana T merupakan banyaknya kelas atau kategori respon

- s : banyak *input* (neuron) dari *layer* sebelumnya
 g_s : *input* (neuron) pada $s=1, 2, \dots, S$ dari *layer* sebelumnya
 B_t : nilai *bias* atau simpangan pada *layer softmax* untuk kelas ke- t
 $w_{t,s}$: bobot atau *weight* pada *layer softmax* untuk kelas ke- t dan neuron ke- s

Pada *layer softmax* dihitung nilai *loss* atau *error* untuk mengetahui perbedaan antara nilai prediksi dengan label yang sebenarnya. Pada penelitian ini, karena dilakukan klasifikasi dengan kategori lebih dari dua, maka digunakan *categorical entropy loss function*. Penghitungan *categorical entropy loss* dilakukan menggunakan Persamaan (2.8) (Ho & Wookey, 2020).

$$f_a = -\frac{1}{N} \sum_{t=1}^T \sum_{n=1}^N y_{n,t} \log(f_{n,t}) \quad (2.8)$$

keterangan:

- f_a : *loss function*
 N : jumlah citra foto
 $y_{n,t}$: nilai target kelas ke- t citra ke- n berupa nilai 0 dan 1
 $f_{n,t}$: nilai probabilitas prediksi kelas ke- t citra ke- n dari persamaan (2.7)
 T : banyak kelas atau kategori respon

2.7 Batch Normalization

Proses *training* dalam membangun model CNN merupakan proses yang rumit, dikarenakan distribusi *input* pada setiap *layer* akan selalu berubah apabila parameter dari *layer* sebelumnya berubah. Fenomena tersebut dinamakan dengan *internal covariate shift* yang dapat memperlambat proses *training* model, karena perlu digunakan *learning rate* yang lebih rendah serta inisialisasi parameter yang harus lebih berhati-hati. Oleh karena itu, digunakan metode *batch normalization* dimana nantinya akan dilakukan normalisasi pada arsitektur model serta pada setiap *mini batch* dari proses *training*. Pada *batch normalization*, digunakan *input* yaitu berupa *mini batch* dengan ukuran u , $O_z = \{x_{z,1}, x_{z,2}, \dots, x_{z,u}\}$. Penghitungan untuk *batch normalization* ditunjukkan dengan Persamaan (2.9).

$$\tau_{z,\lambda} = BN(x_{z,\lambda}) = \gamma x_{z,\lambda} + \delta \quad (2.9)$$

Selanjutnya, komponen dari Persamaan (2.9) dijabarkan pada Persamaan (2.10), Persamaan (2.11), dan Persamaan (2.12)

$$x_{z,\lambda} = \frac{x_{z,\lambda}^{(layer-1)} - \mu_z}{\sqrt{\sigma_z^2 + \epsilon}} \quad (2.10)$$

$$\mu_z = \frac{1}{u} \sum_{\lambda=1}^u x_{z,\lambda}^{(layer-1)} \quad (2.11)$$

$$\sigma_z^2 = \frac{1}{u} \sum_{\lambda=1}^u (x_{z,\lambda}^{(layer-1)} - \mu_z)^2 \quad (2.12)$$

keterangan:

- $\tau_{z,\lambda}$: *output* ke- λ *batch normalization* pada *mini-batch* ke- z
 $x_{z,\lambda}$: nilai *input* ke- λ yang sudah dinormalisasi pada *mini-batch* ke- z

- μ_z : nilai *mean* dari *mini-batch* ke- z
 σ_z^2 : nilai *varians* dari *mini-batch* ke- z
 ε : nilai *error*
 u : banyak input pada *mini batch* (*batch size*)
 $x_{z,\lambda}^{(layer-1)}$: input ke- i pada *mini-batch* ke- z dari *layer* sebelumnya dimana $i = 1, 2, \dots, s$
 z : banyak *mini batch* dalam satu iterasi, dimana $z = 1, 2, \dots, Z$ dan

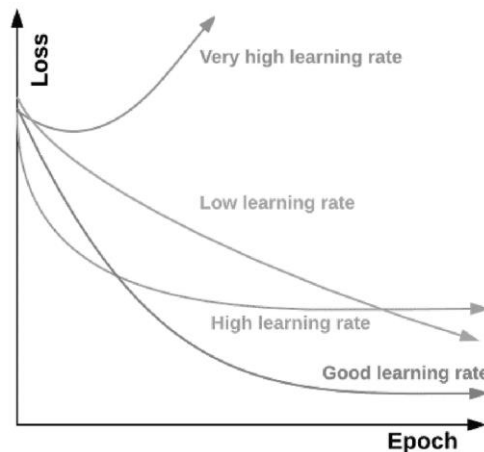
$$Z = \frac{\text{banyak sampel}}{s}$$

 O_z : *mini-batch* ke- z
 γ : parameter *scale*
 δ : parameter *shift*

Batch size merupakan banyaknya sampel yang melewati *neural network* dalam satu waktu. Misalkan terdapat sampel sebanyak 10000, jika digunakan *batch size* 10, maka terdapat 1000 *batches* dimana akan ada 10 data yang secara bersamaan melewati *neural network*. Selanjutnya, ketika semua data telah melewati *neural networks* maka disebut sebagai satu *epoch* (Ioffe & Szegedy, 2015).

2.8 Learning Rate

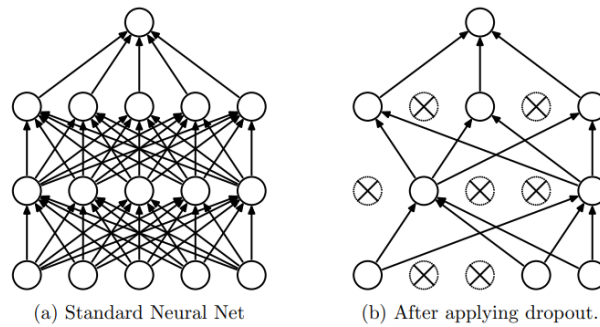
Learning rate merupakan suatu *hyperparameter* yang mengatur berapa besar perubahan pada parameter model berdasarkan estimasi *error* ketika bobot model diperbarui. *Learning rate* dapat dikonfigurasi pada proses *training* model CNN dengan rentang antara 0 hingga 1. *Learning rate* yang bernilai kecil menyebabkan model melakukan perubahan yang lebih sedikit, sehingga memerlukan *epochs* yang lebih banyak untuk mencapai performa yang baik. Sedangkan *learning rate* bernilai besar akan menyebabkan model melakukan perubahan yang lebih cepat, dalam setiap pembaharuan bobotnya, sehingga hanya perlu digunakan *epochs* yang lebih sedikit. *Learning rate* yang terlalu besar dapat menyebabkan model terlalu cepat konvergen pada solusi yang kurang optimal, sedangkan *learning rate* yang terlalu kecil dapat menyebabkan proses *training* cenderung stagnan. Oleh karena itu, penentuan nilai *learning rate* harus dilakukan dengan cermat agar didapatkan performa model yang terbaik (Alzubaidi et al., 2021). Pengaruh penggunaan beberapa nilai *learning rate* terhadap nilai *loss* dari model ditampilkan pada **Gambar 2.8** berikut.



Gambar 2.8 Efek Beberapa Nilai *Learning Rate* terhadap Performa Model (Alzubaidi et al., 2021)

2.9 Dropout

Dropout merupakan suatu metode regularisasi yang bekerja dengan cara menonaktifkan neuron pada suatu model *neural network* secara *random*. Namun pada metode *dropout* ini, neuron tidak dinonaktifkan secara permanen. Ketika neuron dinonaktifkan, maka koneksi dari dan menuju neuron tersebut juga dinonaktifkan. Metode *dropout* dapat diilustrasikan seperti pada **Gambar 2.9** berikut.



Gambar 2.9 Neural Network Tanpa Dropout (a) dan Dengan Dropout (b) (Srivastava et al., 2014)

Secara sederhana, metode *dropout* memberikan setiap neuron probabilitas p yang menunjukkan apakah neuron tersebut akan dinonaktifkan atau tidak. Umumnya, nilai probabilitas p tersebut diatur sebesar 0.5 yang merupakan nilai optimum untuk berbagai macam kasus dan model *neural network*. *Dropout* bertujuan untuk mencegah terjadinya *overfitting*, yaitu ketika model mempelajari data terlalu detail sehingga *noise* data juga tertangkap yang menyebabkan data memiliki performa yang baik pada data *training* namun performanya menurun drastis ketika diaplikasikan pada data *validation*. Dengan adanya *dropout*, maka kompleksitas *neural network* yang berukuran besar dapat dikurangi sehingga memungkinkan data yang tersedia dapat dilatih pada *neural network* yang berbeda beda. Model dengan *feed-forward* operation yang sudah terdapat *dropout* dapat digambarkan dengan Persamaan (2.13) dan Persamaan (2.14) berikut (Srivastava et al., 2014).

$$z_i^{(l+1)} = w_i^{(l+1)} y^l + B_i^{(l+1)} \quad (2.13)$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \quad (2.14)$$

Dimana, $y^{(l)} = r^{(l)} * y^{(l)}$,

keterangan:

$z_i^{(l)}$: vektor *input* pada *layer l* neuron ke- i

$y_i^{(l)}$: vektor *output* dari *layer l* neuron ke- i

$w_i^{(l)}$: bobot pada *layer l* neuron ke- i

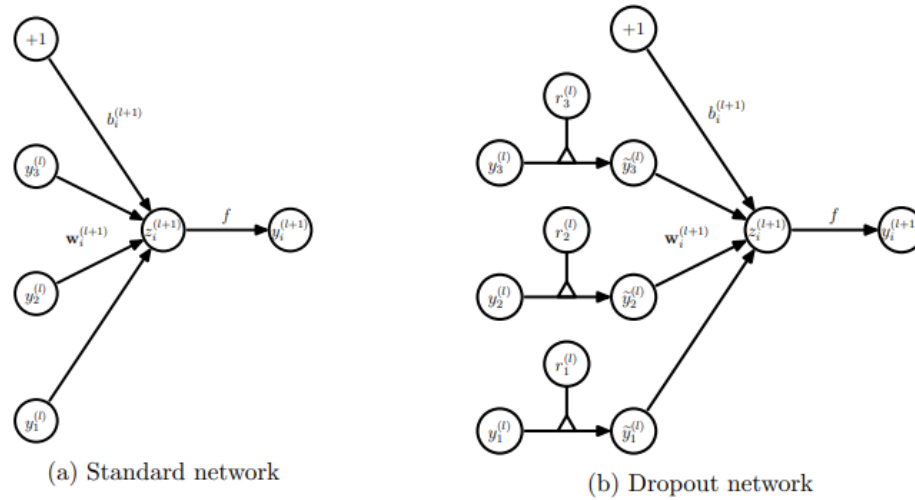
$B_i^{(l)}$: bias pada *layer l* neuron ke- i

$y^{(l)}$: vektor *output layer l* dengan *dropout*

$r^{(l)}$: vektor dari variabel random independen Bernoulli dengan probabilitas p yaitu 1

f : fungsi aktivasi pada *layer*

gambaran operasi *feed forward* pada model *neural network* yang tidak terdapat *dropout* maupun terdapat *dropout* ditunjukkan pada **Gambar 2.10** berikut.



Gambar 2.10 Perbandingan Antara *Standard Network* (a) dan *Dropout Network* (b) (Srivastava et al., 2014)

2.10 Epoch

Epoch dapat didefinisikan sebagai satu siklus dalam proses *training neural network* menggunakan seluruh data yang tersedia. Satu *epoch* dapat terdiri dari satu atau lebih *batch* data, dimana pada setiap *batch* akan digunakan sebagian data untuk melakukan *training* pada model *neural network*. Jumlah *epoch* dalam proses *training* model merupakan salah satu *hyperparameter*, hal tersebut dikarenakan jumlah *epoch* turut menentukan berapa kali dataset harus melalui algoritma *machine learning*. Umumnya, algoritma *machine learning* memerlukan beberapa *epoch* untuk meminimalisir *error* pada model yang sedang dilatih. Namun jumlah *epoch* yang optimal untuk *training* model CNN berbeda-beda antara satu dataset dengan dataset yang lainnya. Proses *training* harus segera dihentikan ketika nilai *validation loss* mulai meningkat setelah konsisten menurun. Penambahan *epoch* yang lebih lanjut dapat berpotensi menyebabkan model menjadi *overfitting* (Afaq & Rao, 2020). *Epoch* biasanya disalah artikan dengan iterasi, iterasi merupakan banyak pengulangan pada semua *batch* pada data *training* yang diperlukan untuk menyelesaikan satu *epoch*. Misalkan pada penelitian digunakan 1000 data dan *batch size* sebesar 1000, maka diperlukan satu kali iterasi per *epoch*. Sedangkan jika digunakan *batch size* sebesar 500, maka diperlukan dua kali iterasi setiap *epoch*. Lalu, apabila digunakan *batch size* sebesar 100, maka perlu 10 iterasi per *epoch* (Afaq & Rao, 2020).

2.11 Early Stopping

Early stopping merupakan suatu metode dalam pencegahan terjadinya *overfitting* pada proses *training* model. *Overfitting* yaitu suatu kondisi dimana model terlalu spesifik mempelajari fitur-fitur data *training*, sehingga ketika diaplikasikan kepada data lain seperti data *validation*, maka model akan menunjukkan performa yang buruk. *Early stopping* bekerja dengan menghentikan proses *training* model pada *epoch* yang lebih awal dari jumlah *epoch* yang sudah ditentukan sebelumnya. Proses *training* tersebut dihentikan ketika diprediksikan *training* selanjutnya tidak menghasilkan model dengan performa yang lebih baik. Dalam melihat performa *training* model tersebut digunakan metrik tertentu untuk evaluasinya seperti akurasi maupun nilai *loss*. *Early stopping* memiliki *hyperparameter* berupa *patience*, dimana *patience*

berfungsi untuk menentukan berapa kali *epoch* yang ditoleransi terkait hasil *training* yang lebih buruk setelah didapatkan performa terbaik dari model. Misalkan jika ditentukan matriks evaluasi berupa *loss* dan *patience* sebesar 2, maka jika didapatkan nilai *loss* terendah, dengan nilai *patience* sebesar 2, proses *training* akan dihentikan jika model tidak mendapatkan nilai *loss* yang lebih rendah dalam 2 *training* selanjutnya. Pemilihan *patience* dalam *early stopping* memiliki efek yang signifikan dalam proses *training* model. Semakin kecil nilai *patience*, maka model akan lebih sering melakukan *early stopping* dikarenakan ruang untuk terjadinya performa lebih buruk pada *training* selanjutnya lebih sering terjadi pada nilai *patience* yang kecil dibandingkan dengan *patience* yang lebih besar (Muhammad et al., 2022).

2.12 Adam Optimizer

Adam optimizer merupakan suatu metode optimasi untuk mengurangi nilai *loss* berbasis *gradient descent stochastic* yang dapat diaplikasikan pada berbagai kasus *deep learning* seperti *computer vision*. Metode *Adam optimizer* diawali dengan melakukan inisiasi untuk β_1 dan β_2 yang merupakan *exponential decay rate*. *Exponential decay rate* yaitu suatu proses reduksi bilangan dengan tingkat presentase yang konsisten dari waktu ke waktu. Selain itu juga ditentukan inisiasi nilai *learning rate* $\alpha = 0.0001$. Selanjutnya, ditentukan fungsi *stochastic objective function* dengan parameter θ yang dapat disimbolkan dengan $f(\theta)$. Kemudian, dilakukan inisiasi momen pertama (m_0) dan momen kedua (v_0) serta *timestep* (a) yang bernilai 0. Lalu, dilakukan langkah *forward* untuk setiap *layer* pada masing-masing arsitektur. Setelah itu, dihitung *gradient loss function* (g_a) berdasarkan Persamaan (2.15).

$$g_a = \nabla_{\theta} f_a(\theta_{a-1}) \quad (2.15)$$

Dimana f_a merupakan *loss function* yang dijabarkan pada Persamaan (2.8). Langkah selanjutnya yaitu melakukan update bias estimasi momen pertama (m_a) dan estimasi momen kedua (v_a) menggunakan Persamaan (2.16) dan Persamaan (2.17).

$$m_a = \beta_1 \cdot m_{a-1} + (1 - \beta_1) \cdot g_a \quad (2.16)$$

$$v_a = \beta_2 \cdot v_{a-1} + (1 - \beta_2) \cdot g_a^2 \quad (2.17)$$

Kemudian dihitung estimasi koreksi bias momen pertama (m_a) dan momen kedua (\hat{v}_a) menggunakan Persamaan (2.18) dan Persamaan (2.19).

$$m_a = \frac{m_a}{(1 - \beta_1^a)} \quad (2.18)$$

$$\hat{v}_a = \frac{v_a}{(1 - \beta_2^a)} \quad (2.19)$$

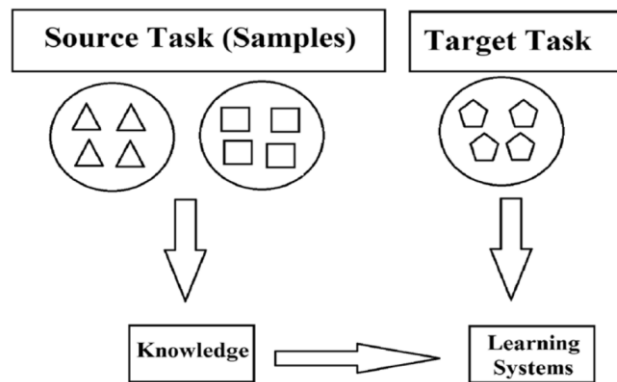
β_1^a dan β_2^a dilakukan *update* sesuai dengan nilai *timestep* yaitu a . Semakin tinggi a , maka nilai β_1^a dan β_2^a akan semakin mendekati 0 sehingga estimasi koreksi bias semakin mendekati 1. Selanjutnya, dilakukan *update* parameter dengan Persamaan (2.20).

$$\theta_a = \theta_{a-1} - \alpha \cdot \frac{m_a}{\sqrt{\hat{v}_a} + \varepsilon} \quad (2.20)$$

Iterasi dilakukan hingga didapatkan parameter bias maupun *weight* yang konvergen (Kingma & Ba, 2014).

2.13 Transfer Learning

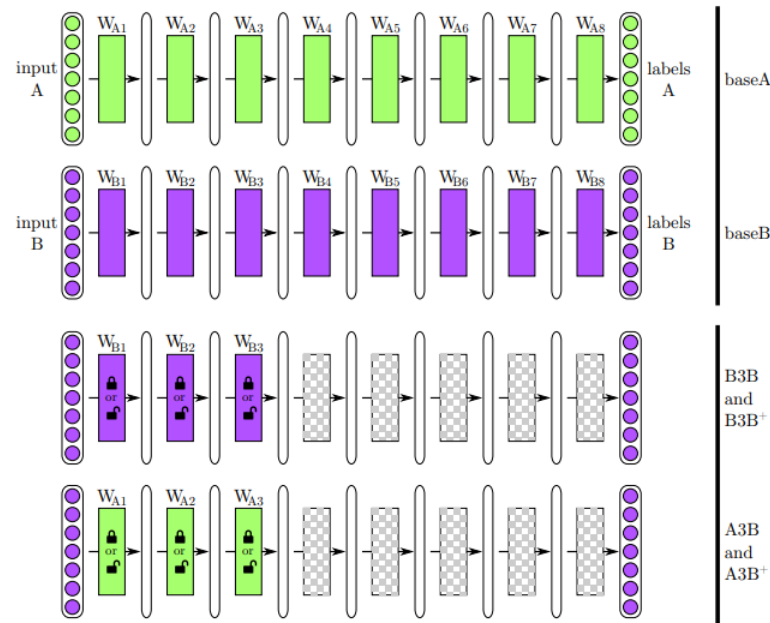
Transfer Learning merupakan suatu metode dalam penggunaan model yang telah di-*training* dengan dataset tertentu untuk melakukan komputasi pada permasalahan yang serupa. Model ini nantinya akan diperbarui dengan parameter yang baru sesuai dengan dataset yang akan digunakan (Rochman & Junaedi, 2020). Ilustrasi dari *Transfer Learning* dapat dilihat pada **Gambar 2.11** berikut.



Gambar 2.11 Konsep *Transfer Learning* (Hosna et al., 2022)

Transfer learning memiliki keunggulan yaitu dapat mempercepat proses *learning* dan pengembangan model dengan memanfaatkan fragmen-fragmen dari modul yang telah dikembangkan sebelumnya. Model yang sudah dikembangkan dan dapat digunakan untuk kasus lainnya disebut dengan *pre-trained* model. *Pre-trained* model umumnya dilatih dengan dataset yang besar. Pendekatan *transfer learning* yang biasanya dilakukan yaitu dengan melakukan *training* pada *base network*, selanjutnya *n layer* pertama dari *network* yang sudah dilakukan *training* disalin pada *target network* yang baru. Sisa *layer* dari *target network* tersebut kemudian diinisiasi secara random dan dilakukan *training* sesuai dengan tugas yang baru. Selain itu, dapat dilakukan pula *backpropagating* dari *error* yang didapatkan dari tugas baru pada *base network* yang sudah disalin (*unfreeze base network*), hal tersebut bertujuan untuk melakukan *fine-tuning base network* pada tugas yang baru. Selain itu, *base network* yang disalin dapat dibekukan sepenuhnya, yang artinya *base network* tidak berubah baik *weight* maupun biasnya untuk tugas yang baru. Namun, walaupun *base network* dibekukan, *layer* terakhir yaitu *layer output* tetap diganti jumlah kategori kelasnya dan disesuaikan dengan jumlah kategori kelas untuk tugas yang baru. Keputusan untuk melakukan *fine-tuning* atau tidak pada *base network* ditentukan dari dataset dan tugas yang akan dikerjakan oleh model. Jika target dataset sedikit dan jumlah parameternya banyak, maka *fine-tuning* dapat memicu *overfitting*, sehingga sebaiknya *base network* dibekukan saja. Namun jika target dataset banyak dan jumlah parameternya sedikit, maka *base network* dapat dilakukan *fine-tuning* pada tugas baru untuk meningkatkan performa model (Yosinski et al., 2014).

Fine-tuning untuk *base network* yang disalin dari model yang sudah dilakukan *training* dengan dataset yang lain dapat dilakukan secara keseluruhan untuk setiap *layer* atau *network* yang disalin maupaun dilakukan sebagian saja. Gambaran *fine-tuning* secara parsial dapat dilihat pada **Gambar 2.12**.

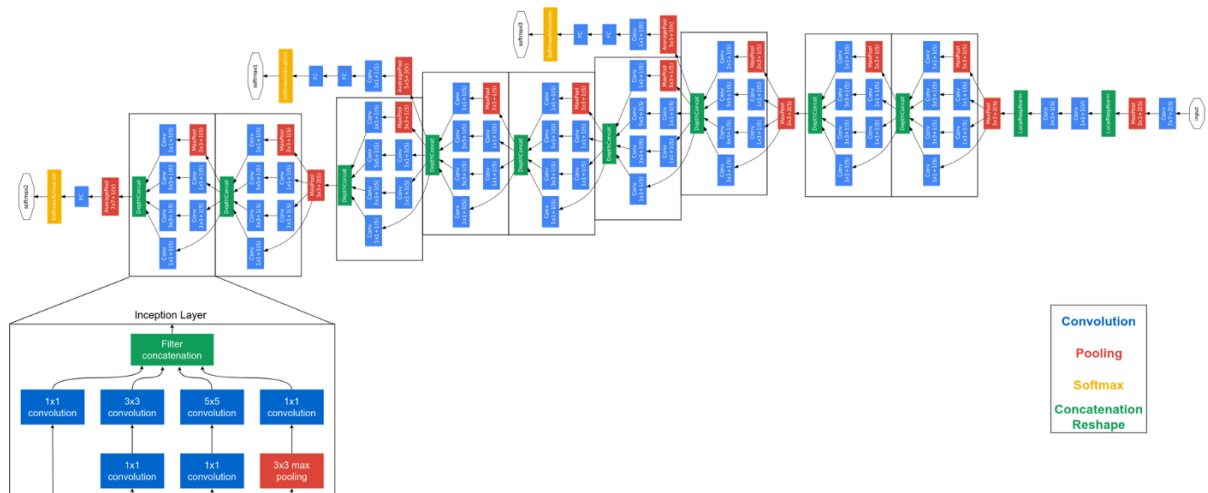


Gambar 2.12 *Fine-tuning* Parsial pada *Base Network* (Yosinski et al., 2014)

Berdasarkan **Gambar 2.12** dapat dilihat bahwa pada model di baris ketiga *base network* yang sudah didapatkan pada model baris kedua digunakan pada tiga *layer* pertama dari model untuk dataset yang sama yaitu dataset B, sedangkan untuk *layer* selanjutnya, parameter diinisiasi secara random. Kemudian untuk model pada baris keempat, *base network* yang didapatkan pada model baris pertama digunakan pada tiga *layer* pertama model baru untuk dataset yang berbeda dengan *base network*-nya, yaitu dataset B, sedangkan untuk *layer* selanjutnya, parameter diinisiasi secara random. Baik pada model baris ketiga maupun baris keempat, *base network* yang disalin dapat dikondisikan menjadi *freeze* ataupun *unfreeze* untuk dilakukan *fine-tuning* sesuai dengan kebutuhan untuk tugas yang baru. Inisiasi parameter berdasarkan *base network* yang sudah dilakukan *training* dapat meningkatkan performa model walaupun sudah dilakukan *fine-tuning* secara sebagian maupun keseluruhan pada model baru (Yosinski et al., 2014).

2.13.1 GoogLeNet

GoogLeNet merupakan salah satu modifikasi CNN yang memiliki prinsip kerja yaitu deteksi citra foto dengan jumlah *layer* hingga 22 *layer*. Arsitektur GoogLeNet menggunakan input gambar beresolusi 224x224 pixel dengan RGB *color channel*. GoogLeNet dilatih menggunakan dataset Imagenet dengan jumlah data sebanyak 15 juta foto dan 1000 kategori. Arsitektur GoogLeNet secara umum terdiri dari beberapa jenis *layer* diantaranya *convolutional layer* dan *pooling layer* untuk melakukan ekstraksi data, serta *fully connected layer* untuk menangkap hasil pemrosesan pada *layer* sebelumnya serta berfungsi untuk melakukan klasifikasi *multi class* (Sa'idah et al., 2022). Arsitektur GoogLeNet diilustrasikan pada **Gambar 2.13** berikut.



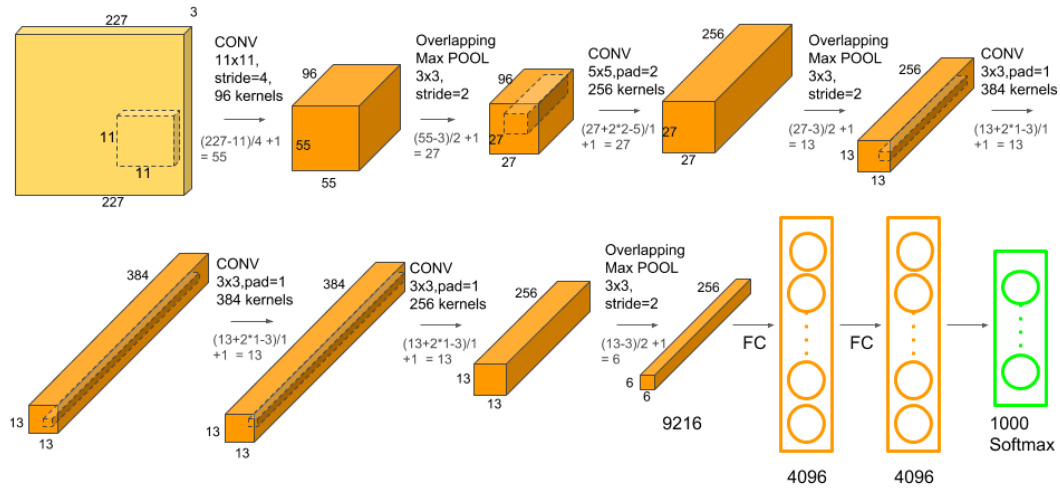
Gambar 2.13 Arsitektur GoogLeNet (Szegedy et al., 2014)

Arsitektur GoogLeNet memiliki *layer* unik yang membedakan dengan arsitektur CNN lainnya yang disebut sebagai *inception layer* atau *inception module*. Modul ini terdiri dari *layer-layer* paralel berupa *convolutional layer* dengan kernel berukuran 1x1, 3x3, dan 5x5 yang berfungsi untuk mengekstrak informasi penting dari citra foto dengan ukuran spasial yang beragam. Selain itu terdapat pula *max pooling layer* dengan kernel berukuran 3x3. Pada arsitektur ini, sebelum *convolutional layer* berukuran 3x3 dan 5x5, terlebih dahulu ditambahkan *convolutional layer* berukuran 1x1, penambahan *layer* tersebut bertujuan untuk mengurangi kompleksitas dari model. Selain itu, *convolutional layer* berukuran 1x1 juga ditambahkan setelah *max pooling layer* agar tidak terjadi kenaikan jumlah *output* yang tidak terkontrol pada *layer – layer* selanjutnya karena adanya *merge* antara *output convolutional layer* dan *max pooling layer*. Kemudian, dari *layer – layer* tersebut akan disambungkan dengan filter *concatenation* menjadi satu *output* yang selanjutnya akan menjadi *input* untuk tahapan berikutnya. Selain *inception layer*, GoogLeNet juga memiliki *fully connected layer* dengan 1024 neuron, diikuti dengan *layer softmax* yang memiliki kategori klasifikasi sebanyak 1000 kelas (Szegedy et al., 2014).

2.13.2 AlexNet

AlexNet merupakan salah satu arsitektur *Convolutional Neural Network* yang dibangun dengan dataset ImageNet, AlexNet mampu meningkatkan akurasi klasifikasi gambar secara signifikan pada dataset ImageNet yang berisikan foto dengan jumlah lebih dari 15 juta foto beresolusi tinggi yang telah dikategorikan dan mencakup sebanyak 1000 kategori. AlexNet bekerja dengan memaksimalkan rata-rata setiap *training* dari log-probabilitas label yang benar pada hasil prediksinya. Arsitektur Alexnet terdiri dari 7 *layer*, dengan *layer-layer* penyusunnya yaitu 5 *convolutional layer*, dan 2 *fully connected layer*, serta *layer Softmax* untuk melakukan klasifikasi (Krizhevsky et al., 2012). *Convolutional layer* pertama dari AlexNet melakukan filter terhadap *input* gambar yang memiliki ukuran 227 x 227-pixel dengan RGB color channel. Pada *layer* ini terdapat 96 kernel berukuran 11x11x3 dengan *stride* yaitu 4 pixel. Kemudian, *convolutional layer* kedua mengambil *input* dari *convolutional layer* pertama setelah sebelumnya melalui *max pooling layer*, lalu melakukan filter terhadap foto menggunakan 256 kernel berukuran 5x5x48. Kemudian proses dilanjutkan pada *max pooling layer* dan berlanjut dengan *convolutional layer* ketiga dengan 384 kernel berukuran 3x3x256. Lalu, dilanjutkan dengan *convolutional layer* keempat dan kelima dengan ukuran kernel yang sama yaitu

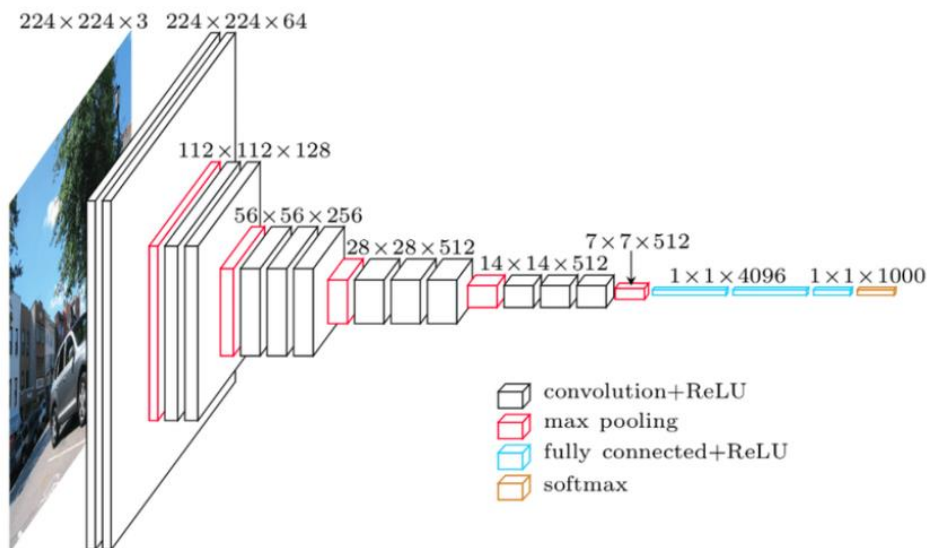
3x3x192 namun dengan jumlah kernel yang berbeda, dimana pada *convolutional layer* keempat terdapat 384 kernel, sedangkan pada *convolutional layer* kelima terdapat 256 kernel. Selanjutnya terdapat 2 *fully connected layer* yang memiliki 4096 neuron yang terhubung satu sama lain baik dengan *layer* sebelumnya maupun dengan *layer* sesudahnya yang berupa *softmax layer* dengan jumlah kategori yaitu sebanyak 1000 kategori (Krizhevsky et al., 2012). Arsitektur Alexnet dapat diilustrasikan seperti **Gambar 2.14** berikut.



Gambar 2.14 Arsitektur Alexnet (Krizhevsky et al., 2012)

2.13.3 VGG16

VGG16 merupakan arsitektur CNN yang didesain untuk melakukan klasifikasi gambar. Arsitektur VGG16 tersusun dari 16 *layer*, dengan 13 *convolution layer* dan 3 *fully connected layer*. Data *input* untuk arsitektur VGG16 yaitu berupa citra foto RGB dengan ukuran 224x224 *pixel*. Arsitektur VGG16 telah mencapai kinerja tinggi pada berbagai dataset *benchmark* salah satunya yaitu ImageNet dimana pada dataset tersebut terdapat 15 juta foto dengan 1000 kategori klasifikasi (Simonyan & Zisserman, 2014). Arsitektur VGG16 dapat diilustrasikan seperti pada **Gambar 2.15** berikut.



Gambar 2.15 Arsitektur VGG16 (Simonyan & Zisserman, 2014)

VGG16 dibangun dengan menggunakan kernel-kernel yang berukuran 3x3 dan 1x1 pada *convolutional layer*-nya. Kemudian pada *convolutional layer* tersebut digunakan *stride* sebesar 1 *pixel*, digunakan pula *padding* agar resolusi foto tetap terjaga setelah melalui proses pada *convolutional layer*. Selain itu, terdapat lima *max pooling layer* setelah beberapa *convolutional layer*. Pada *max pooling layer* tersebut, terdapat kernel-kernel yang memiliki ukuran 2x2 dengan *stride* yaitu 2 *pixel*. Selain disusun dengan tumpukan-tumpukan *convolutional layer* yang sudah dijelaskan sebelumnya, arsitektur VGG16 juga memiliki susunan yaitu tiga *fully connected layer* yang terletak di akhir arsitektur dengan neuron sebanyak 4096 pada masing-masing *layer* diikuti dengan *softmax layer* yang memiliki kategori klasifikasi sebanyak 1000 *class*. Semua *hidden layer* yang terdapat pada arsitektur ini menggunakan fungsi aktivasi ReLU (Simonyan & Zisserman, 2014).

2.14 Ketepatan Klasifikasi

Kinerja suatu algoritma klasifikasi perlu diukur dan dievaluasi untuk menentukan apakah algoritma klasifikasi tersebut dapat melakukan klasifikasi dengan baik atau tidak. Salah satu metode yang dapat digunakan untuk evaluasi performa dari algoritma klasifikasi yaitu *Confusion Matrix*. Prinsip dari *Confusion Matrix* yaitu membandingkan hasil klasifikasi dari algoritma dengan klasifikasi yang sebenarnya (Ali et al., 2019). Hasil klasifikasi yang diukur dengan *Confusion Matrix* dibedakan menjadi empat jenis yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN) yang diilustrasikan pada **Gambar 2.16**. Berdasarkan keempat jenis klasifikasi tersebut, dapat dihitung tiga pengukuran ketepatan klasifikasi yaitu presisi, akurasi, dan *recall*.

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

Gambar 2.16 Confusion Matrix (Ali et al., 2019)

Presisi merupakan salah satu pengukuran ketepatan klasifikasi dengan cara membagi data kelas positif yang tepat terklasifikasi antara prediksi dan hasil yang sebenarnya dengan total data yang terklasifikasi positif (Ali et al., 2019). Presisi dapat dinyatakan seperti pada Persamaan 2.21 berikut.

$$presisi = \frac{TP}{(TP + FP)} \quad (2.21)$$

Akurasi merupakan salah satu pengukuran ketepatan klasifikasi dengan cara membagi data kelas positif dan negatif yang tepat terklasifikasi antara prediksi dan hasil yang sebenarnya dengan semua data (Ali et al., 2019). Akurasi dapat dinyatakan seperti pada Persamaan (2.22) berikut.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.22)$$

Recall merupakan salah satu pengukuran ketepatan klasifikasi yang dihitung dengan cara membagi data kelas positif yang tepat terklasifikasi antara prediksi dan hasil yang sebenarnya dengan total data yang tepat dalam kelas positif dan yang tidak tepat dalam kelas negatif (Ali et al., 2019). *Recall* dapat dinyatakan dalam Persamaan (2.23) berikut.

$$recall = \frac{TP}{TP + FN} \quad (2.23)$$

2.15 Pariwisata

Pariwisata merupakan kegiatan bepergian ke tempat tujuan di luar lingkungan tempat tinggal untuk tujuan rekreasi, budaya, atau bisnis. Pariwisata adalah industri yang memiliki banyak aspek yang melibatkan berbagai pemangku kepentingan, termasuk wisatawan, penyedia layanan pariwisata, dan masyarakat setempat. Pariwisata dapat melibatkan berbagai kegiatan, seperti jalan-jalan, belanja, kuliner, dan berpartisipasi dalam acara budaya (UNWTO, 2019). Menurut Organisasi Pariwisata Dunia Perserikatan Bangsa-Bangsa (UNWTO), kunjungan wisatawan internasional tumbuh sebesar 5% pada tahun 2018 hingga mencapai total 1,4 miliar. Pariwisata merupakan sumber pendapatan penting bagi banyak negara, dan diperkirakan menyumbang sekitar 10% dari PDB global dan satu dari sepuluh pekerjaan di seluruh dunia (UNWTO, 2019).

BAB 3 METODOLOGI

3.1 Sumber Data

Data yang digunakan pada penelitian ini merupakan data sekunder yaitu berupa dataset Places365 sebanyak 6000 data gambar yang didapatkan dari website <http://places2.csail.mit.edu/download.html>. Dataset Places365 digunakan sebagai data *training* dan data *validation*. Namun, dari dataset tersebut, tidak digunakan semua kategori, beberapa kategori yang dipilih dari dataset Places365 yaitu pantai, hutan, gunung, air terjun, restoran, dan museum. Kategori-kategori objek pariwisata tersebut dipilih karena dapat ditemukan di Indonesia. Selanjutnya digunakan pula data *testing* yang berasal dari website Google Images sebanyak 600 data gambar. Data *testing* tersebut digunakan untuk mengetahui performa model terhadap gambar objek pariwisata spesifik yang berada di Indonesia.

3.2 Variabel Penelitian

Variabel penelitian yang digunakan yaitu berupa citra atau gambar dari beberapa kategori objek wisata yang ditampilkan pada **Tabel 3.1** sebagai berikut.

Tabel 3.1 Jenis Objek Pariwisata	
No	Objek Wisata
1	Pantai
2	Hutan
3	Gunung
4	Air Terjun
5	Restoran
6	Museum

Selanjutnya dari objek wisata yang sudah ditentukan tersebut, didapatkan variabel penelitian seperti yang tercantum pada **Tabel 3.2** berikut

Tabel 3.2 Variabel Penelitian			
Variabel	Keterangan	Skala	Nilai
$(X_{n,u,v})_{\text{RGB}}$	Nilai dari setiap <i>pixel</i>	Interval	$0 \leq X_{n,u,v} \leq 255$
$Y_{n,k}$	Kategori citra objek wisata ke- n	Nominal	1, 2, ..., 6

keterangan:

$(X_{n,u,v})_{\text{RGB}}$: Interval nilai per *channel* warna RGB

$0 < u \leq U$; U sejumlah m pixel merupakan baris dari citra gambar




$0 < v \leq V$; V sejumlah m pixel merupakan kolom dari citra gambar

$0 < n \leq N$; N = Total data dari keseluruhan kategori objek wisata

3.3 Struktur Data

Struktur data yang digunakan pada penelitian ini dapat terbagi menjadi 2, yaitu struktur data secara general dan struktur data secara spesifik yang akan digunakan pada proses *training* model CNN. Struktur data secara general ditampilkan pada **Tabel 3.3**, sedangkan struktur data yang lebih spesifik ditampilkan pada **Tabel 3.4**. Berikut merupakan struktur data secara general yang digunakan pada penelitian ini.

Tabel 3.3 Struktur Data

No.	Foto (X)	Kategori Objek Wisata (Y)
1		Y_1
2		Y_2
:	:	:
N		Y_N

Sementara untuk struktur data yang lebih detail dan yang akan digunakan untuk melakukan proses klasifikasi yaitu ditampilkan pada **Tabel 3.4**. Karena pada penelitian ini digunakan data gambar RGB, maka didapatkan struktur data sebagai berikut.

Tabel 3.4 Struktur Data yang akan Diproses

Citra Ke-	$X_{1,1}$	$X_{2,1}$...	$X_{U,1}$...	$X_{U,v}$	Y_k
1	$([X_{1,1,1}]_R,$ $[X_{1,1,1}]_G,$ $[X_{1,1,1}]_B)$	$([X_{1,2,1}]_R,$ $[X_{1,2,1}]_G,$ $[X_{1,2,1}]_B)$...	$([X_{1,U,1}]_R,$ $[X_{1,U,1}]_G,$ $[X_{1,U,1}]_B)$...	$([X_{1,U,v}]_R,$ $[X_{1,U,v}]_G,$ $[X_{1,U,v}]_B)$	Y_1
2	$([X_{2,1,1}]_R,$ $[X_{2,1,1}]_G,$ $[X_{2,1,1}]_B)$	$([X_{2,2,1}]_R,$ $[X_{2,2,1}]_G,$ $[X_{2,2,1}]_B)$...	$([X_{2,U,1}]_R,$ $[X_{2,U,1}]_G,$ $[X_{2,U,1}]_B)$...	$([X_{2,U,v}]_R,$ $[X_{2,U,v}]_G,$ $[X_{2,U,v}]_B)$	Y_2
:	:	:	⋮	:	⋮	:	:
N	$([X_{N,1,1}]_R,$ $[X_{N,1,1}]_G,$ $[X_{N,1,1}]_B)$	$([X_{N,2,1}]_R,$ $[X_{N,2,1}]_G,$ $[X_{N,2,1}]_B)$...	$([X_{N,U,1}]_R,$ $[X_{N,U,1}]_G,$ $[X_{N,U,1}]_B)$...	$([X_{N,U,v}]_R,$ $[X_{N,U,v}]_G,$ $[X_{N,U,v}]_B)$	Y_N

3.4 Langkah Analisis

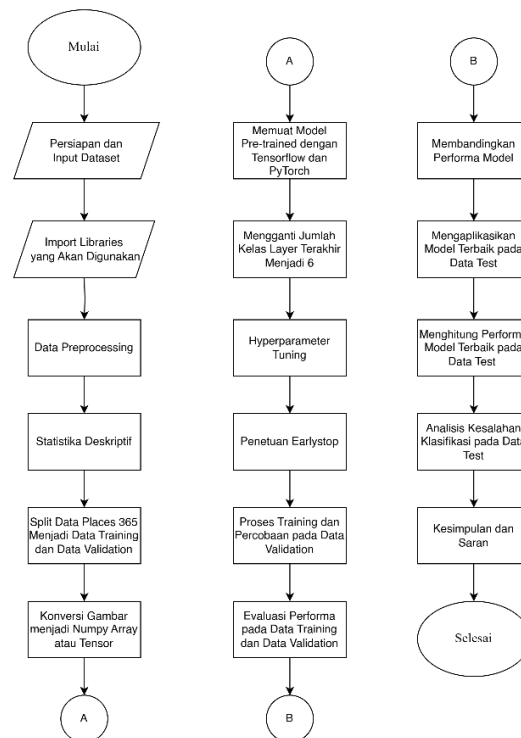
Langkah analisis yang dilakukan dalam penelitian ini untuk melakukan klasifikasi citra foto objek pariwisata adalah sebagai berikut:

1. Mempersiapkan dataset yang akan digunakan. Data *training* dan data *validation* didapatkan dengan mengunduh *subset* data Places365 yang sesuai dengan kategori objek wisata. Sedangkan untuk data *test* didapatkan dengan mengunduh gambar melalui Google Images dengan kata kunci "(kategori objek pariwisata) di Indonesia".
2. Mengimpor *libraries* yang dibutuhkan untuk proses *training* model.
3. Melakukan data *preprocessing* dengan mengatur ulang ukuran (*resize*) gambar menjadi 224 x 224 piksel agar sesuai dengan *input* model. Selanjutnya dilakukan *normalize* pada data gambar agar rentang intensitas berada antara 0 – 1 untuk semua piksel.
4. Statistika deskriptif dengan menampilkan beberapa sampel data gambar serta histogram *color channel* dari beberapa sampel gambar.
5. Membagi dataset Places365 menjadi data *training* dan data *validation* dengan proporsi data *training* 0,8 dan data *validation* 0,2.
6. Konversi data gambar menjadi NumPy Array untuk pemrosesan dengan model yang diimpor dengan *library* Tensorflow, serta konversi data gambar menjadi Tensor untuk pemrosesan dengan model yang diimpor menggunakan *library* PyTorch.

7. Melakukan klasifikasi citra dengan metode *Convolutional Neural Network* (CNN) menggunakan arsitektur GoogLeNet, Alexnet, dan VGG16 dengan langkah-langkah sebagai berikut:
 - a) Memuat model *pre-trained* yang akan digunakan melalui API Tensorflow atau PyTorch.
 - b) Mengganti jumlah *layer* output dari 1000 kategori klasifikasi menjadi 6 kategori klasifikasi objek pariwisata.
 - c) Menentukan kondisi *freeze* atau *unfreeze* pada *layer* terakhir model.
 - d) Melakukan inisiasi *hyperparameter* berupa pengaturan *learning rate*, *batch size*, dan *dropout*.
 - e) Menentukan *earlystop* sebelum proses *training* yang bertujuan agar didapatkan *epoch* optimum dan mencegah terjadinya *overfitting* pada model.
 - f) Proses *training* model dilakukan pada data *training*, lalu model dicoba pada data *validation*.
8. Menghitung kebaikan model yang terbentuk dengan nilai akurasi dan *loss* baik pada data *training* maupun data *validation*.
9. Membandingkan hasil kinerja klasifikasi antara berbagai kombinasi *hyperparameter* pada model GoogLeNet, Alexnet, dan VGG16.
10. Mengaplikasikan model dengan performa terbaik pada data *test*.
11. Menghitung performa model pada data *test* dengan akurasi.
12. Melakukan analisis kesalahan klasifikasi model pada data *test*.
13. Menarik kesimpulan dan saran.

3.5 Diagram Alir

Berdasarkan Langkah-langkah penelitian yang sudah dipaparkan, maka dapat disusun diagram alir untuk penelitian seperti pada **Gambar 3.1** berikut.



Gambar 3.1 Diagram Alir Penelitian

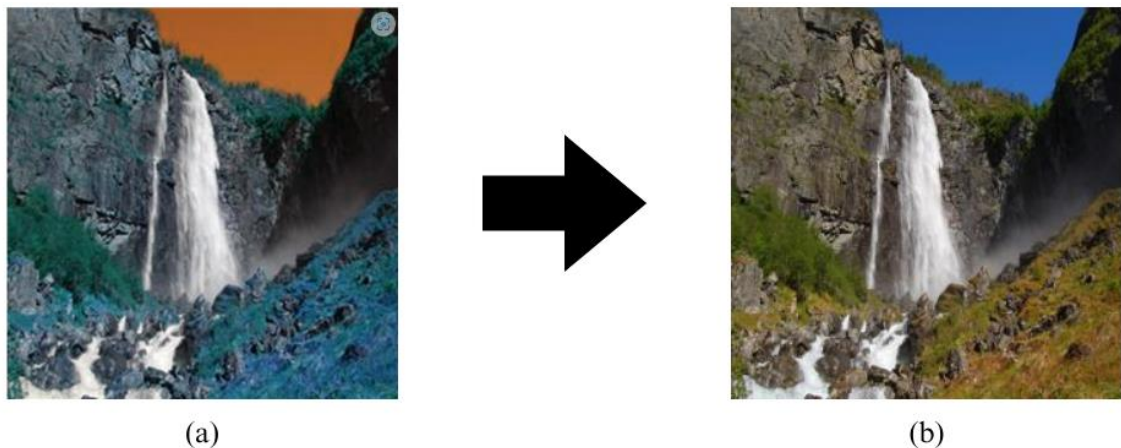
(Halaman Ini Sengaja Dikosongkan)

BAB 4 HASIL DAN PEMBAHASAN

Penelitian ini melakukan klasifikasi gambar objek pariwisata yang terdiri dari enam kategori, dimana pada setiap kategori terdapat 1000 data gambar objek pariwisata, sehingga total data yang digunakan yaitu 6000 data gambar. Selain itu, digunakan data *testing* sebanyak 600 data. Metode klasifikasi yang digunakan pada penelitian ini yaitu *pre-trained* model CNN yang terdiri dari tiga arsitektur yang berbeda yakni VGG16, GoogLeNet, dan AlexNet. Kebaikan hasil klasifikasi dari masing-masing model diukur dengan nilai akurasi dan *loss*.

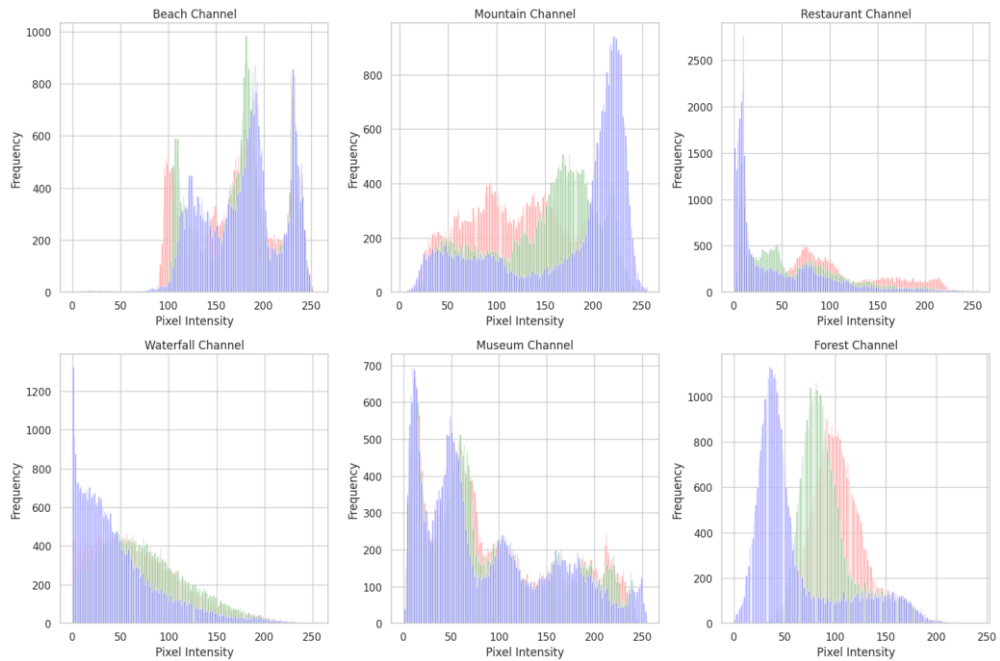
4.1 Data Preprocessing

Tahapan pertama yang dilakukan pada penelitian ini yaitu data *preprocessing*. Data didapatkan dari situs <http://places2.csail.mit.edu/download.html>. Pada penelitian ini digunakan dua jenis *library machine learning* yaitu TensorFlow dan PyTorch. Dalam memuat data gambar, kedua *library* tersebut memiliki prinsip yang berbeda. Pada PyTorch, data dimuat menggunakan *library* dari Torch yaitu DataLoader. Ketika data gambar dimuat dengan DataLoader, maka data tersebut strukturnya akan menjadi Tensor yang merupakan struktur dasar dari *library* PyTorch, selain itu gambar juga sudah langsung dimuat dengan *color channel* RGB. Sedangkan pada *library* TensorFlow, data gambar dimuat menggunakan *library* OpenCV. Ketika data dimuat menggunakan *library* OpenCV, maka data harus memiliki struktur ndarray. Oleh karena itu pada *pre-trained* model yang menggunakan *library* TensorFlow yaitu VGG16, maka data gambar perlu dikonversi dahulu menjadi ndarray menggunakan *library* numpy. Data yang dimuat menggunakan *library* OpenCV otomatis akan memiliki *color channel* BGR (*Blue, Green, Red*). *Color channel* tersebut perlu diubah terlebih dahulu menjadi RGB agar sesuai dengan *input color channel* dari masing-masing model arsitektur CNN. Contoh konversi *color channel* BGR ke RGB terdapat pada **Gambar 4.1** berikut.



Gambar 4.1 Konversi *Color Channel* Gambar dari BGR (a) ke RGB (b)

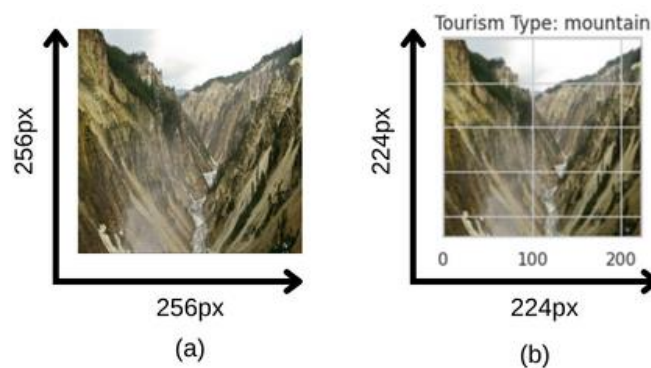
Lalu, setelah data dimuat dan disesuaikan *color channel*-nya, maka selanjutnya dapat dilihat intensitas dari masing-masing warna di *color channel* RGB untuk satu sampel gambar dari masing-masing kategori, yang ditunjukkan dengan histogram yang ditampilkan pada **Gambar 4.2** berikut.



Gambar 4.2 Histogram *Color Channel* RGB dari Satu Gambar pada Setiap Kategori

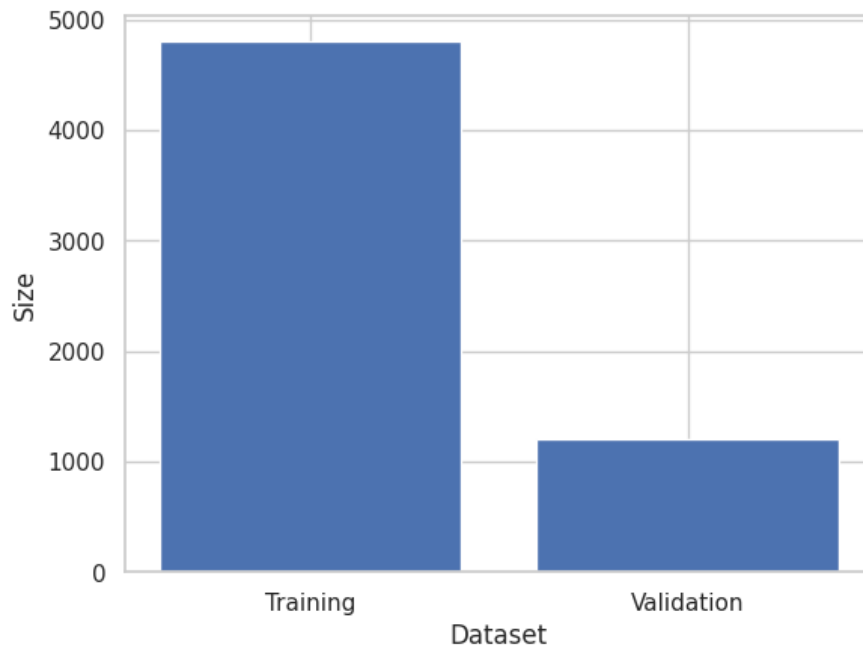
Sumbu horizontal pada histogram tersebut menunjukkan intensitas piksel gambar untuk masing-masing warna pada *color channel* RGB, dimana semakin tinggi nilainya maka menunjukkan intensitas warna merah, hijau, dan biru yang semakin tinggi pada gambar. Lalu, sumbu vertikal pada histogram menunjukkan frekuensi piksel yang memiliki nilai intensitas warna tertentu. Berdasarkan contoh sampel dari masing-masing kategori objek pariwisata pada histogram tersebut, dapat diketahui bahwa frekuensi piksel pada kategori pantai dan gunung cenderung didominasi dengan nilai intensitas warna biru dan hijau yang relatif tinggi dibandingkan dengan nilai intensitas warna merahnya. Sedangkan untuk kategori lainnya, frekuensi piksel dengan nilai intensitas warna biru yang rendah relatif lebih banyak dibandingkan dengan dua kategori sebelumnya.

Tahap preprocessing selanjutnya yaitu dilakukan *Image standardizing (resize)* yaitu proses penyesuaian ukuran gambar menjadi 224x224 piksel. Proses *resize* dilakukan untuk menyesuaikan ukuran gambar dengan ukuran input yang diterima oleh masing-masing model CNN. Contoh gambar yang sudah dilakukan proses *resize* dapat dilihat pada **Gambar 4.3** berikut.



Gambar 4.3 Contoh Hasil dari Proses *Resize* (a) Gambar Asli (b) Gambar *Resize*

Setelah dilakukan proses *resize* pada gambar, selanjutnya gambar dinormalisasi (*normalize*) pada rentang nilai piksel-piksennya. Gambar yang awalnya memiliki nilai piksel antara 0-255 dinormalisasikan nilainya menjadi antara rentang 0-1. Setelah dilakukan normalisasi, maka data sudah siap untuk dilakukan proses *training* pada masing-masing *pre-trained* model. Namun, sebelum proses training tersebut dimulai, data terlebih dahulu dipisah menjadi dua bagian, yaitu data *training* dan data *validation*. Proporsi untuk masing-masing data *training* dan data *validation* yaitu sebesar 0,8 dan 0,2. Hasil dari pemisahan dataset menjadi data *training* dan *testing* dapat dilihat pada **Gambar 4.4** berikut.



Gambar 4.4 Proporsi Data *Training* dan Data *Validation*

4.2 Evaluasi Performa Model

Pre-trained model yang diukur performanya pada penelitian ini yaitu VGG16, GoogLeNet, dan AlexNet. Performa model pada penelitian ini diukur menggunakan akurasi dan *loss* baik itu dari performa pada data *training* maupun data *validation*.

4.2.1 Performa Model VGG16

Model VGG16 yang digunakan pada penelitian ini, merupakan model yang sudah dilakukan *pre-training* pada dataset ImageNet. Model VGG16 dimuat menggunakan *library* TensorFlow dengan API Keras. Dikarenakan pada kasus ini jumlah kelas dari *pre-trained* model dan data yang digunakan tidak sama, maka *fully connected layer* bawaan pada VGG16 yang didapatkan dari *library* TensorFlow tidak diikutsertakan. Oleh karena itu, perlu ditambahkan *fully connected layer* sendiri yang jumlah outputnya disesuaikan dengan kelas pada dataset yang digunakan pada penelitian ini yaitu enam kelas. **Tabel 4.1** merupakan arsitektur VGG16 yang sudah ditambahkan *fully connected layer* yang baru. Penambahan *fully connected layer* pada arsitektur VGG16 tersebut juga turut menambah total parameter yang terdapat pada model, yang awalnya hanya berkisar 14,7 juta parameter, menjadi sekitar 33,6 juta parameter.

Tabel 4.1 Arsitektur VGG16 dari Tensorflow

<i>Layer Type</i>	<i>Output Shape</i>	<i>Parameter</i>
input_2 (input layer)	[(batch size, 224, 224, 3)]	0
blok1_conv1 (Conv2D)	(batch size, 224, 224, 64)	1792
blok1_conv2 (Conv2D)	(batch size, 224, 224, 64)	36928
blok1_pool (MaxPooling2D)	(batch size, 112, 112, 64)	0
blok2_conv1 (Conv2D)	(batch size, 112, 112, 128)	73856
blok2_conv2 (Conv2D)	(batch size, 112, 112, 128)	147584
blok2_pool (MaxPooling2D)	(batch size, 56, 56, 128)	0
blok3_conv1 (Conv2D)	(batch size, 56, 56, 256)	295168
blok3_conv2 (Conv2D)	(batch size, 56, 56, 256)	590080
blok3_conv3 (Conv2D)	(batch size, 56, 56, 256)	590080
blok3_pool (MaxPooling2D)	(batch size, 28, 28, 256)	0
blok4_conv1 (Conv2D)	(batch size, 28, 28, 512)	1180160
blok4_conv2 (Conv2D)	(batch size, 28, 28, 512)	2359808
blok4_conv3 (Conv2D)	(batch size, 28, 28, 512)	2359808
blok4_pool (MaxPooling2D)	(batch size, 14, 14, 512)	0
blok5_conv1 (Conv2D)	(batch size, 14, 14, 512)	2359808
blok5_conv2 (Conv2D)	(batch size, 14, 14, 512)	2359808
blok5_conv3 (Conv2D)	(batch size, 14, 14, 512)	2359808
blok5_pool (MaxPooling2D)	(batch size, 7, 7, 512)	0
global_avg_pooling2d	(batch size, 512)	0
dense_3 (Dense)	(batch size, 4096)	2101248
dense_4 (Dense)	(batch size, 4096)	16781312
dropout	(batch size, 4096)	0
dense (5)	(none, 6)	24582
Total Parameter		33621830

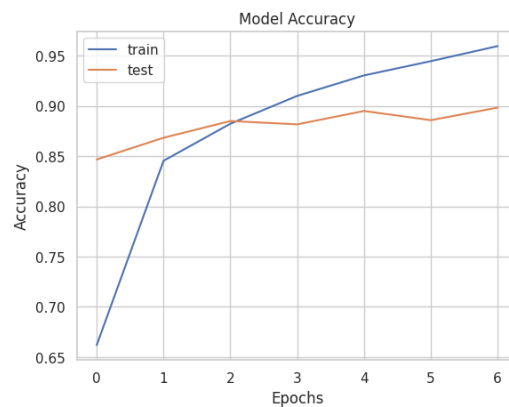
Proses selanjutnya setelah model dimuat dan disesuaikan dengan dataset pada penelitian ini, yaitu proses *training*. *Training* pada model VGG16 dilakukan sebanyak lima kali dengan kombinasi *hyperparameter* yang berbeda-beda agar didapatkan kombinasi yang terbaik. Kemudian, setelah dilakukan proses *training* sebanyak lima kali, didapatkan hasil performa untuk masing-masing kombinasi seperti pada **Tabel 4.2** berikut.

Tabel 4.2 Performa Berbagai Kombinasi Pengujian untuk VGG16

No	Freeze	<i>Hyperparameter</i>				Hasil			
		<i>Epoch</i>	<i>Learning Rate</i>	<i>Batch</i>	<i>Dropout</i>	<i>Akurasi Data Training</i>	<i>Loss Data Training</i>	<i>Akurasi Data Validation</i>	<i>Loss Data Validation</i>
1	Tidak	6	0,0001	32	0,5	0,9752	0,0767	0,8725	0,4558
2	Ya	5	0,0001	32	0,5	0,9638	0,1121	0,8867	0,4848
3	Ya	8	0,00001	32	0,5	0,9879	0,0482	0,89	0,3799
4	Ya	7	0,00001	64	0,5	0,9596	0,1188	0,8983	0,3156
5	Ya	8	0,00001	64	0,3	0,9804	0,0662	0,8933	0,3586

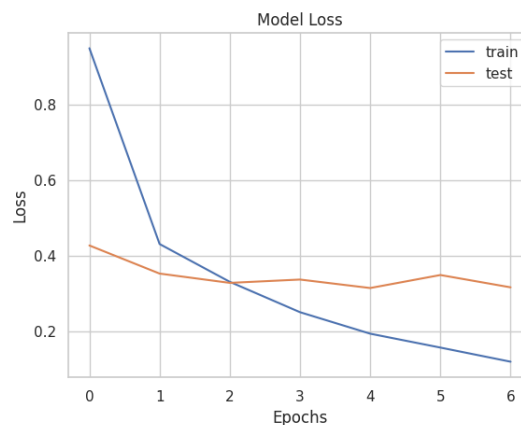
Berdasarkan performa dari masing-masing kombinasi pengujian model VGG16 pada **Tabel 4.1** diketahui bahwa pengaturan atau *tuning* yang dilakukan pada *hyperparameter* memiliki pengaruh yang beragam pada hasil *training* model yang direpresentasikan dalam nilai akurasi dan nilai *loss*. Dari hasil yang sudah didapatkan, kombinasi pada pengujian keempat menunjukkan hasil yang paling baik dibandingkan dengan kombinasi yang lainnya. Hal

tersebut dapat dilihat berdasarkan nilai akurasi data *validation* yang paling tinggi serta nilai *loss* data *validation* yang paling rendah dibandingkan dengan kombinasi pengujian yang lainnya. Walaupun nilai akurasi untuk data *training* merupakan yang terendah dan nilai *loss* data *training* merupakan yang tertinggi pada kombinasi *hyperparameter* keempat, namun hasil tersebut, masih tergolong relatif baik. Oleh karena itu, hasil pada data *validation* lebih dipertimbangan dalam menentukan model VGG16 dengan performa terbaik, karena dianggap lebih menggambarkan performa model dalam data yang tidak diketahui sebelumnya dalam proses *training*. Kombinasi pengujian keempat tersebut diantaranya *layer convolutional* pada *block 5* dan *fully connected layer* pada arsitektur model disetel pada kondisi *unfreeze* sehingga model dapat mempelajari data baru pada penelitian ini. Selanjutnya learning rate diatur sebesar 0.00001, *batch size* sebesar 64, *dropout* sebesar 0.5, kemudian *epoch* setelah dilakukan *early stopping* sebanyak 7 *epoch*. **Gambar 4.5** dan **Gambar 4.6** berikut merupakan gambaran performa model VGG16 dengan kombinasi yang keempat.



Gambar 4.5 Akurasi Model VGG16 dengan Kombinasi Pengujian Keempat

Hasil nilai akurasi yang didapatkan dari proses *training* yaitu sebesar 0,9596 untuk data *training* dan 0,8983 untuk data *validation*. Proses *training* model VGG16 dengan kombinasi pengujian keempat dilakukan sebanyak 7 *epoch*. Selanjutnya nilai *loss* ditampilkan pada **Gambar 4.6** berikut.



Gambar 4.6 Nilai *Loss* Model VGG16 dengan Kombinasi Pengujian Keempat

Nilai *loss* yang didapatkan pada proses *training* model VGG16 dengan kombinasi pengujian keempat yaitu sebesar 0,1188 untuk data *training* dan 0,3156 untuk data *validation*.

4.2.2 Performa Model GoogLeNet

Model GoogLeNet yang digunakan pada penelitian ini, merupakan model yang sudah dilakukan *pre-training* pada dataset ImageNet. Model GoogLeNet dimuat menggunakan *library* PyTorch. **Tabel 4.3** berikut merupakan arsitektur dari model GoogLeNet yang didapatkan dari *library* tersebut.

Tabel 4.3 Arsitektur GoogLeNet dari *Library* PyTorch

<i>Layer Type</i>	<i>Output Shape</i>	<i>Parameter</i>
Conv2d-1	[-1, 64, 112, 112]	9536
MaxPool2d	[-1, 64, 56, 56]	0
Conv2d-2	[-1, 64, 56, 56]	4224
Conv2d-3	[-1, 192, 56, 56]	110976
MaxPool2d	[-1, 192, 28, 28]	0
Inception 3a	[-1, 256, 28, 28]	155872
Inception 3b	[-1, 480, 28, 28]	337224
Inception 4a	[-1, 512, 14, 14]	364512
Inception 4b	[-1, 512, 14, 14]	425232
Inception 4c	[-1, 512, 14, 14]	486192
Inception 4d	[-1, 528, 14, 14]	573312
Inception 4e	[-1, 832, 14, 14]	803840
Inception 5a	[-1, 832, 7, 7]	978944
Inception 5b	[-1, 1024, 7, 7]	1347040
Dropout	[-1, 1024]	0
Linear	[-1, 6]	6150
Total Parameter		5603054

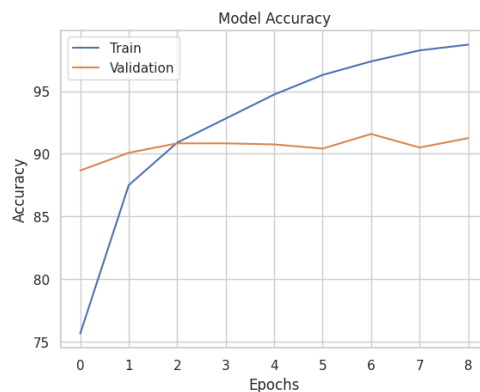
Model *pre-trained* GoogLeNet yang sudah diimpor dari *library* PyTorch masih memiliki *layer* terakhir dimana jumlah kelas untuk klasifikasinya masih sesuai dengan kelas bawaan dari data yang digunakan untuk melakukan *training* pada GoogLeNet yaitu ImageNet, sejumlah 1000 kelas. Oleh karena itu, *layer* terakhir pada GoogLeNet diubah sesuai dengan jumlah kelas pada data yang digunakan pada penelitian ini yaitu sebanyak 6 kelas. Setelah *fully connected layer* diubah jumlah *outputnya*, parameter yang ada pada *pre-trained* model GoogLeNet berubah menjadi sekitar 5,6 juta parameter.

Proses selanjutnya setelah model dimuat dan disesuaikan dengan dataset pada penelitian ini, yaitu proses *training*. *Training* pada model GoogLeNet dilakukan sebanyak lima kali dengan kombinasi pengujian yang berbeda-beda agar didapatkan kombinasi yang terbaik. Kemudian, setelah dilakukan proses *training* sebanyak lima kali, didapatkan hasil performa untuk masing-masing kombinasi yaitu ditampilkan pada **Tabel 4.4** berikut.

Tabel 4.4 Performa Berbagai Kombinasi Pengujian untuk GoogLeNet

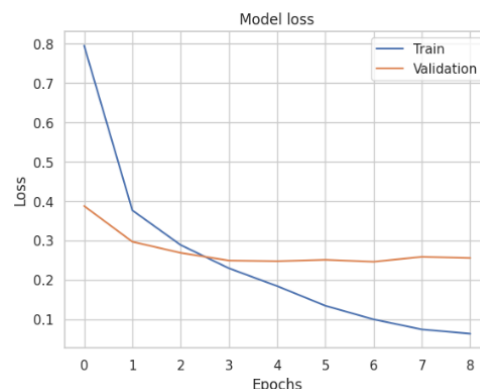
No	Freeze	<i>Hyperparameter</i>				Hasil			
		<i>Epoch</i>	<i>Learning Rate</i>	<i>Batch</i>	<i>Dropout</i>	<i>Akurasi Data Training</i>	<i>Loss Data Training</i>	<i>Akurasi Data Validation</i>	<i>Loss Data Validation</i>
1	Tidak	36	0,0001	32	0,5	0,8471	0,4368	0,8767	0,355
2	Ya	9	0,0001	32	0,5	0,9871	0,0631	0,9125	0,2533
3	Ya	31	0,00001	32	0,5	0,9338	0,2099	0,905	0,2651
4	Ya	9	0,0001	64	0,5	0,9883	0,0712	0,9092	0,2571
5	Ya	6	0,0001	32	0,3	0,969	0,114	0,9017	0,2614

Berdasarkan performa dari masing-masing kombinasi pengujian model GoogLeNet pada **Tabel 4.4** diketahui bahwa kombinasi pada pengujian kedua menunjukkan hasil yang paling baik dibandingkan dengan kombinasi yang lainnya. Hal tersebut dapat dilihat berdasarkan nilai akurasi data *validation* yang paling tinggi serta nilai *loss* data *validation* yang paling rendah dibandingkan dengan kombinasi yang lainnya. Selain itu nilai akurasi pada data *training* pada kombinasi *hyperparameter* kedua juga merupakan yang tertinggi kedua setelah kombinasi keempat, sedangkan nilai *loss* pada data *training* juga merupakan yang paling rendah jika dibandingkan dengan kombinasi pengujian yang lainnya. Kombinasi pengujian kedua tersebut diantaranya *block layer inception 5a*, *inception 5b* dan *fully connected layer* pada arsitektur model disetel pada kondisi *unfreeze* sehingga model dapat mempelajari data baru pada penelitian ini. Selanjutnya learning rate diatur sebesar 0.0001, *batch size* sebesar 32, *dropout* sebesar 0.5, kemudian *epoch* setelah dilakukan *early stopping* sebanyak 9 *epoch*. **Gambar 4.7** dan **Gambar 4.8** berikut merupakan hasil training dari kombinasi pengujian kedua.



Gambar 4.7 Akurasi Model GoogLeNet dengan Kombinasi Pengujian Kedua

Hasil *training* model GoogLeNet dengan kombinasi *hyperparameter* kedua menghasilkan nilai akurasi untuk data *training* sebesar 0,9871 sedangkan untuk data *validation* sebesar 0,9125. Proses *training* model GoogLeNet dengan kombinasi pengujian kedua dilakukan sebanyak 9 *epoch*. Selanjutnya untuk nilai *loss* ditampilkan pada **Gambar 4.8** berikut.



Gambar 4.8 Nilai *Loss* Model GoogLeNet dengan Kombinasi Pengujian Kedua

Nilai *loss* yang didapatkan pada proses *training* model GoogLeNet dengan kombinasi pengujian kedua yaitu sebesar 0,0631 untuk data *training* dan 0,2533 untuk data *validation*.

4.2.3 Performa Model AlexNet

Model AlexNet yang digunakan pada penelitian ini, merupakan model yang sudah dilakukan *pre-training* pada dataset ImageNet. Model AlexNet dimuat menggunakan *library* PyTorch. **Tabel 4.5** berikut merupakan arsitektur dari model AlexNet yang didapatkan dari *library* tersebut.

Tabel 4.5 Arsitektur AlexNet dari *Library* PyTorch

<i>Layer Type</i>	<i>Output Shape</i>	<i>Parameter</i>
Conv2d	[-1, 64, 55, 55]	23296
ReLU	[-1, 64, 55, 55]	0
MaxPool2d	[-1, 64, 27, 27]	0
Conv2d	[-1, 192, 27, 27]	307392
ReLU	[-1, 192, 27, 27]	0
MaxPool2d	[-1, 192, 13, 13]	0
Conv2d	[-1, 384, 13, 13]	663936
ReLU	[-1, 384, 13, 13]	0
Conv2d	[-1, 256, 13, 13]	884992
ReLU	[-1, 256, 13, 13]	0
Conv2d	[-1, 256, 13, 13]	590080
ReLU	[-1, 256, 13, 13]	0
MaxPool2d	[-1, 256, 6, 6]	0
AdaptiveAvgPool2d	[-1, 256, 6, 6]	0
Dropout	[-1, 9216]	0
Linear	[-1, 4096]	37752832
ReLU	[-1, 4096]	0
Dropout	[-1, 4096]	0
Linear	[-1, 4096]	16781312
ReLU	[-1, 4096]	0
Linear	[-1, 6]	24582
Total Parameter		57028422

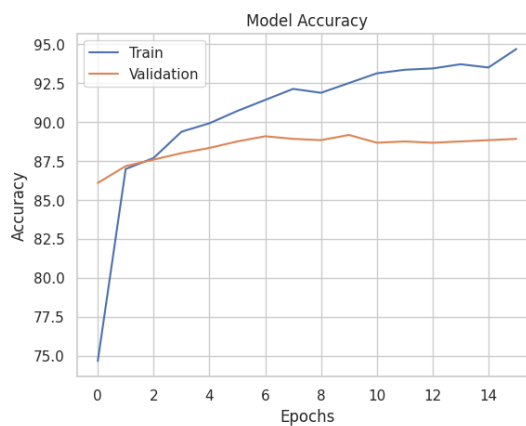
Model *pre-trained* AlexNet yang sudah diimpor dari *library* PyTorch masih memiliki jumlah parameter yang ada pada *pre-trained* model AlexNet sekitar 57 juta parameter.

Proses selanjutnya setelah model dimuat dan disesuaikan dengan dataset pada penelitian ini, yaitu proses *training*. *Training* pada model AlexNet dilakukan sebanyak lima kali dengan kombinasi *hyperparameter* yang berbeda-beda agar didapatkan kombinasi yang terbaik. Kemudian, setelah dilakukan proses training sebanyak lima kali, didapatkan hasil performa untuk masing-masing kombinasi yang ditampilkan pada **Tabel 4.6** berikut.

Tabel 4.6 Performa Berbagai Kombinasi Pengujian untuk AlexNet

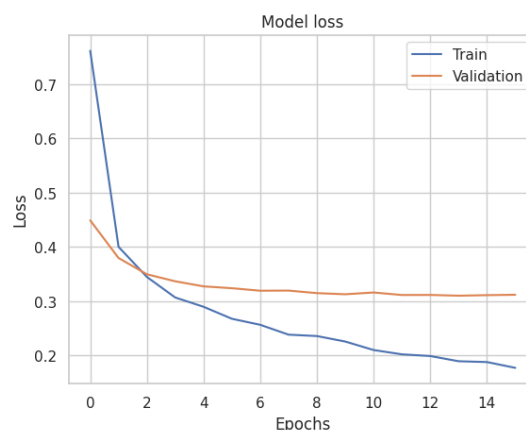
No	Freeze	<i>Hyperparameter</i>				Hasil			
		<i>Epoch</i>	<i>Learning Rate</i>	<i>Batch</i>	<i>Dropout</i>	<i>Akurasi Data Training</i>	<i>Loss Data Training</i>	<i>Akurasi Data Validation</i>	<i>Loss Data Validation</i>
1	Tidak	16	0,0001	32	0,5	0,9469	0,1773	0,8892	0,312
2	Ya	3	0,0001	32	0,5	0,9673	0,1021	0,88	0,317
3	Tidak	50	0,00001	32	0,5	0,9108	0,2656	0,8833	0,3214
4	Tidak	16	0,0001	64	0,5	0,9302	0,2078	0,8883	0,3107
5	Tidak	13	0,0001	32	0,3	0,9544	0,1592	0,8883	0,3146

Berdasarkan performa dari masing-masing kombinasi pengujian model AlexNet pada **Tabel 4.6** diketahui bahwa pengaturan atau *tuning* yang dilakukan pada *hyperparameter* memiliki pengaruh yang beragam pada hasil *training* model yang direpresentasikan dalam nilai akurasi dan nilai *loss*. Dari hasil yang sudah didapatkan, kombinasi pada pengujian pertama menunjukkan hasil yang paling baik dibandingkan dengan kombinasi yang lainnya. Hal tersebut dapat dilihat berdasarkan nilai akurasi data *validation* yang paling tinggi serta nilai *loss* data *validation* yang terendah kedua setelah kombinasi *hyperparameter* yang keempat. Kombinasi pengujian pertama tersebut diantaranya seluruh *layer* AlexNet diatur pada kondisi *freeze* sehingga model tidak dapat mempelajari data baru pada penelitian ini, selanjutnya learning rate diatur sebesar 0.00001, *batch size* sebesar 64, *dropout* sebesar 0.5, kemudian *epoch* setelah dilakukan *early stopping* sebanyak 8 *epoch*. **Gambar 4.9** dan **Gambar 4.10** berikut merupakan hasil *training* model AlexNet dengan kombinasi *hyperparameter* pertama.



Gambar 4.9 Akurasi Model AlexNet dengan Kombinasi Pengujian Pertama

Hasil *training* model AlexNet dengan kombinasi *hyperparameter* pertama menghasilkan nilai akurasi untuk data *training* sebesar 0,9469 sedangkan untuk data *validation* sebesar 0,8892. Proses *training* model AlexNet dengan kombinasi pengujian pertama dilakukan sebanyak 16 *epoch*. Selanjutnya untuk nilai *loss* ditampilkan pada **Gambar 4.10** berikut.



Gambar 4.10 Nilai Loss Model AlexNet dengan Kombinasi Pengujian Pertama

Nilai *loss* yang didapatkan pada proses *training* model AlexNet dengan kombinasi pengujian pertama yaitu sebesar 0,1773 untuk data *training* dan 0,312 untuk data *validation*.

4.3 Pemilihan Model dengan Performa Terbaik

Proses *training* pada ketiga model *pre-trained* dilakukan sebanyak lima kali untuk masing-masing model dengan kombinasi *hyperparameter* yang berbeda-beda. Pada masing-masing model didapatkan kombinasi *hyperparameter* dengan performa terburuk dan performa terbaik. Kombinasi dengan performa terbaik untuk masing-masing model kemudian dibandingkan untuk mengetahui model *pre-trained* terbaik untuk penelitian ini. Perbandingan model *pre-trained* dengan kombinasi terbaik untuk masing-masing model ditampilkan pada **Tabel 4.7** berikut.

Tabel 4.7 Performa Terbaik dari Masing-masing Model

No	Model	Kombinasi Ke-	Akurasi Data Training	Loss Data Training	Akurasi Data Validation	Loss Data Validation
1	VGG16	4	0,9596	0,1188	0,8983	0,3156
2	GoogLeNet	2	0,9871	0,0631	0,9125	0,2533
3	AlexNet	1	0,9469	0,1773	0,8892	0,312

Perbandingan performa untuk kombinasi terbaik dari setiap model pada **Tabel 4.7** menunjukkan bahwa GoogLeNet memiliki performa yang paling baik dibandingkan dengan model lainnya. Hasil tersebut didapatkan karena beberapa faktor, yang pertama yaitu dari arsitektur GoogLeNet. Layer *inception* pada GoogLeNet memiliki kelebihan yaitu kemampuan untuk memproses data gambar dengan ukuran *convolutional layer* yang bervariasi yang kemudian diagregatkan menjadi satu *output* untuk *layer* selanjutnya. Hal tersebut memungkinkan *layer* selanjutnya untuk mendapatkan fitur-fitur penting dari gambar melalui skala yang berbeda-beda dalam waktu yang bersamaan. Selain itu, struktur Layer *inception* juga memiliki peran lainnya dalam mendapatkan performa terbaik seperti pada **Tabel 4.7**. Adanya *convolutional layer* berukuran 1x1 setelah *convolutional layer parallel* berukuran 3x3 dan 5x5 membantu untuk reduksi parameter model. Reduksi parameter tersebut menyebabkan GoogLeNet memiliki total parameter sebesar 5,6 juta parameter. Jumlah parameter tersebut merupakan yang paling kecil dibandingkan dengan VGG16 yang memiliki parameter sebanyak 33,6 juta dan AlexNet yang memiliki parameter sebanyak 57 juta. Jumlah parameter yang sedikit memiliki keunggulan yaitu meminimalisir terjadinya *overfitting* ketika proses *training* dilakukan pada jumlah data yang sedikit. Hal tersebut sesuai dengan kondisi pada penelitian ini, dimana jumlah data yang digunakan yaitu sebanyak 6000 data gambar. Lebih sedikit dibandingkan dengan jumlah parameter dari model-model *pre-trained* yang digunakan. Selain itu, jumlah parameter yang sedikit juga memiliki keuntungan lain yaitu penggunaan sumber daya yang sedikit pula untuk proses *training* modelnya.

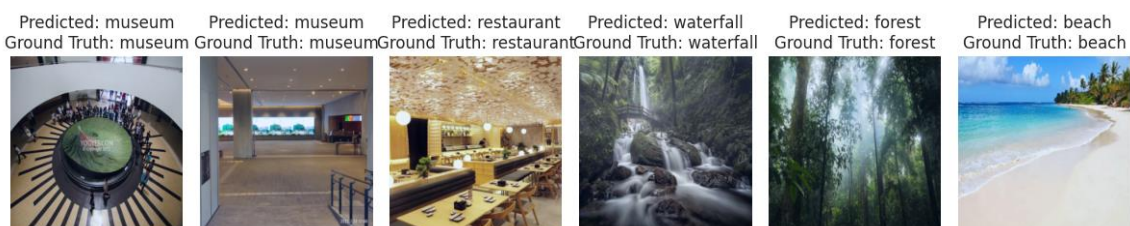
4.4 Implementasi Model Terbaik pada Data Test

Model dengan performa terbaik yang sudah didapatkan yaitu GoogLeNet selanjutnya diaplikasikan pada dataset yang tidak digunakan dalam proses *training* yaitu data *test*. Data *test* ini merupakan data yang didapatkan dari Google Images, dimana gambar objek pariwisata yang didapatkan mayoritas merupakan objek pariwisata yang terdapat di Indonesia. Data tersebut dipilih agar nantinya dapat diketahui performa model terhadap data di dunia nyata yaitu terkait pariwisata di Indonesia. Setelah data dimuat selanjutnya model GoogLeNet diaplikasikan untuk memprediksi kategori dari data gambar tersebut. Hasil dari prediksi yang sudah dilakukan ditampilkan pada *confusion matrix* pada **Gambar 4.11** berikut.

		Confusion Matrix					
True Labels	beach	92	4	3	0	0	1
	forest	0	96	4	0	0	0
	mountain	1	1	98	0	0	0
	museum	0	0	0	98	2	0
	restaurant	0	0	0	6	94	0
	waterfall	0	9	1	0	0	90
		beach	forest	mountain	museum	restaurant	waterfall
		Predicted Labels					

Gambar 4.11 *Confusion Matrix* Prediksi Model GoogLeNet pada Data Test

Confusion matrix tersebut menunjukkan bahwa model GoogLeNet memiliki kemampuan prediksi yang beragam untuk berbagai kategori objek pariwisata, namun secara umum performa klasifikasi GoogLeNet sudah baik dengan klasifikasi yang tepat paling tidak lebih dari atau sama dengan 90 gambar dari 100 gambar pada setiap kategorinya. Performa prediksi terbaik terdapat pada kategori *mountain* dan *museum* dengan dengan hasil prediksi yang tepat yaitu sebanyak 98 gambar dari 100 gambar pada kategori tersebut. Sedangkan performa prediksi terburuk terjadi pada *waterfall*, dalam hal ini model berhasil memprediksikan dengan tepat sebanyak 90 gambar dari 100 gambar pada kategori tersebut. Contoh beberapa sampel *output* prediksi model GoogLeNet pada data *test* ditampilkan pada **Gambar 4.12** berikut.



Gambar 4.12 Contoh *Output* Prediksi Model GoogLeNet pada Data Test

4.5 Analisis Prediksi yang Tidak Tepat terhadap Data Test

Prediksi model GoogLeNet pada data test yang ditunjukkan pada *confusion matrix* di **Gambar 4.11** menghasilkan akurasi sebesar 0.9467. Nilai akurasi tersebut lebih tinggi dibandingkan dengan hasil pada data *validation* yaitu sebesar 0.9125. Selain itu didapatkan pula nilai *loss* pada data test sebesar 0.1158. Nilai *loss* tersebut lebih rendah daripada nilai *loss* pada data *validation* sebesar 0.2533. Namun walaupun nilai akurasi dan nilai *loss* pada data test tersebut sudah relatif lebih baik dibandingkan pada data *validation*, masih terdapat beberapa gambar yang masih dikategorikan salah oleh model GoogLeNet dengan detail kesalahan

klasifikasi yang ditunjukkan pada *confusion matrix* tersebut. Oleh karena itu berikut dilakukan analisis pada beberapa sampel gambar yang dikategorikan salah oleh model GoogLeNet. Dalam melakukan analisis kesalahan klasifikasi, perlu diketahui probabilitas dari suatu gambar untuk diklasifikasikan pada setiap kategori. Hal tersebut bertujuan agar analisis lebih objektif sehingga dapat diketahui letak fitur-fitur pada gambar yang menyebabkan model salah dalam mengklasifikasikan gambar.

1. Kesalahan Klasifikasi pada Kategori *Beach*

Kesalahan klasifikasi pada data gambar kategori *beach* terjadi pada 8 data. Sebanyak 4 data diklasifikasikan pada kategori *forest*, selanjutnya 3 data diklasifikasikan kedalam kategori *mountain*, dan 1 gambar diklasifikasikan dalam kategori *waterfall*. Kesalahan klasifikasi data gambar kategori *beach* menjadi kategori *forest* ditampilkan pada **Gambar 4.13** berikut.



Gambar 4.13 Kesalahan Klasifikasi Kategori *Beach* menjadi *Forest*

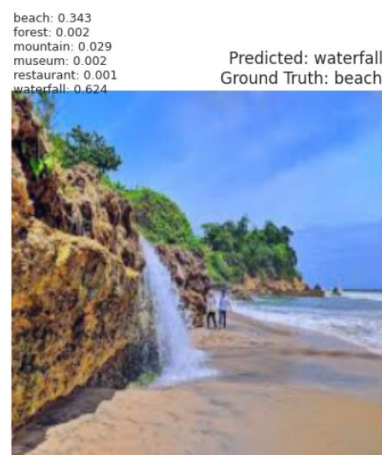
Tiga kesalahan klasifikasi kategori *beach* yang terjadi pada **Gambar 4.13** (a), (c), dan (d), dikarenakan adanya fitur pada gambar tersebut yang sering muncul pada kategori *forest* yaitu pepohonan. Pada **Gambar 4.13** (a) terdapat beberapa batang pohon yang menutupi pantai sehingga model membaca gambar tersebut sebagai hutan. Hal tersebut didukung dengan probabilitas klasifikasi pada kategori *forest* sebesar 0,745. Namun, model masih dapat mengenali fitur pantai pada gambar tersebut, ditunjukkan dengan probabilitas klasifikasi pada kategori *beach* sebesar 0,193. Selanjutnya **Gambar 4.13** (c) dan (d) memiliki permasalahan yang serupa, dimana fitur pohon terlihat dominan pada gambar, sedangkan fitur pantai atau laut terlihat sangat sedikit bahkan warnanya cenderung menyatu dengan langit. Hal tersebut menyebabkan probabilitas klasifikasi untuk kategori *forest* pada kedua gambar tersebut sangat tinggi yaitu diatas 0,95, sedangkan untuk kategori *beach* probabilitas klasifikasinya sangat rendah yaitu sebesar 0,037 untuk **Gambar 4.13** (c) dan 0,014 untuk **Gambar 4.13** (d). Sedangkan untuk **Gambar 4.13** (b), permasalahan yang menyebabkan terjadinya kesalahan

klasifikasi yaitu dominannya fitur warna hijau pada gambar yang disebabkan oleh adanya persawahan. Fitur tersebut tidak terdapat pada data *training* dikarenakan mayoritas data *training* pada kategori pantai memiliki fitur yang dominan yaitu berupa pasir. Hal tersebut menyebabkan probabilitas klasifikasi gambar pada kategori *forest* sebesar 0,927, sedangkan untuk kategori *beach* sebesar 0,057. Selanjutnya kesalahan klasifikasi untuk kategori *beach* menjadi *mountain* ditampilkan pada **Gambar 4.14** berikut.



Gambar 4.14 Kesalahan Klasifikasi Kategori *Beach* menjadi *Mountain*

Kesalahan klasifikasi pada **Gambar 4.14** (a) dan (b) disebabkan karena fitur perbukitan tampak dominan pada gambar, sehingga model membaca gambar tersebut menjadi kategori *mountain*. Hal tersebut didukung dengan nilai probabilitas klasifikasi pada kategori gunung yaitu sebesar 0,667 pada **Gambar 4.14** (a) dan 0,662 pada **Gambar 4.14** (b). Namun pada kedua gambar tersebut, fitur pantai yaitu laut dan pasir masih cukup dikenali oleh model, ditunjukkan dengan nilai probabilitas kategori *beach* sebesar 0,318 pada **Gambar 4.14** (a) dan 0,184 pada **Gambar 4.14** (b). Kemudian, pada **Gambar 4.14** (c), kesalahan klasifikasi lebih kompleks, karena pada gambar tersebut fitur seperti perbukitan tidak begitu tampak. Namun kesalahan klasifikasi dapat disebabkan karena dominannya fitur bebatuan pada gambar, selain itu pada **Gambar 4.14** (c) lautan juga tidak memiliki ombak dan bahkan terlihat refleksi dari langit, sehingga menyebabkan lautan dan langit seperti satu bagian yang sama. Berdasarkan fitur tersebut, model mengklasifikasikan **Gambar 4.14** (c) pada kategori *mountain* dengan probabilitas 0,689, sedangkan probabilitas pada kategori *beach* sebesar 0,130. Selanjutnya, kesalahan klasifikasi kategori *beach* menjadi *waterfall* ditunjukkan pada **Gambar 4.15** berikut.



Gambar 4.15 Kesalahan Klasifikasi Kategori *Beach* menjadi *Waterfall*

Kesalahan klasifikasi pada **Gambar 4.15** disebabkan karena pada gambar tersebut terdapat air terjun yang langsung menuju area pantai. Fitur air terjun tampak jelas pada gambar, yang menyebabkan model mengklasifikasikan **Gambar 4.15** pada kategori *waterfall* dengan probabilitas sebesar 0,624. Namun, model masih mengenali fitur-fitur pada kategori pantai, ditunjukkan dengan probabilitas pada kategori *beach* sebesar 0,343.

2. Kesalahan Klasifikasi pada Kategori *Forest*

Kesalahan klasifikasi pada kategori *forest* terjadi pada 4 data, keempat data tersebut diklasifikasikan pada kategori *mountain*. **Gambar 4.16** berikut menampilkan kesalahan klasifikasi tersebut.



Gambar 4.16 Kesalahan Klasifikasi Kategori *Forest* menjadi *Mountain*

Kesalahan klasifikasi untuk keempat data pada **Gambar 4.16** disebabkan oleh permasalahan yang serupa yaitu munculnya fitur perbukitan yang dominan pada data, hal tersebut dikarenakan hutan dipotret dari jarak yang jauh sehingga yang lebih menonjol adalah bentuk perbukitan dibandingkan dengan fitur kategori *forest* seperti pepohonan. Oleh karena itu, model mengklasifikasikan keempat data tersebut kedalam kategori *mountain*. Probabilitas klasifikasi pada kategori *mountain* untuk keempat data tersebut yaitu sebesar 0,922 untuk **Gambar 4.16** (a), kemudian 0,550 untuk **Gambar 4.16** (b), 0,506 untuk **Gambar 4.16** (c), dan 0,803 untuk **Gambar 4.16** (d).

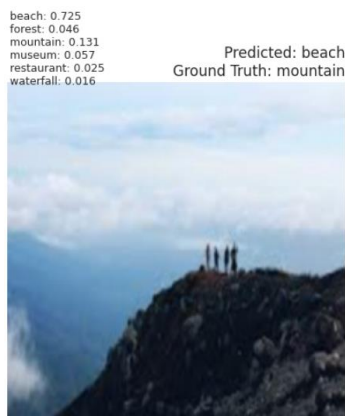
3. Kesalahan Klasifikasi pada Kategori *Mountain*

Klasifikasi yang dilakukan model pada kategori *mountain* mengalami dua kali kesalahan, kedua data terklasifikasikan pada kategori yang berbeda yaitu *beach* dan *forest*. Data yang terklasifikasikan kedalam kategori *forest* ditampilkan pada **Gambar 4.17** berikut.



Gambar 4.17 Kesalahan Klasifikasi Kategori *Mountain* menjadi *Forest*

Gambar 4.17 menunjukkan bahwa gambar memiliki probabilitas sebesar 0,476 untuk diklasifikasikan kedalam kategori *forest*, hal tersebut disebabkan fitur kategori *mountain* yaitu siluet perbukitan tidak begitu terlihat pada gambar. Selain itu, ukuran pohon yang cukup besar pada area pojok kiri bawah cukup mendominasi bidang gambar, hal tersebut memungkinkan model membaca gambar termasuk dalam kategori *forest*. Selain itu, peluang kategori *beach* juga cukup besar yaitu 0,330. Hal tersebut kemungkinan disebabkan oleh adanya fitur kabut pada gambar yang terbaca sebagai fitur ombak oleh model, sehingga kategori *beach* memiliki peluang yang cukup besar, sedangkan kategori *mountain* memiliki peluang relatif kecil yaitu 0,104. Selanjutnya kesalahan klasifikasi kategori *mountain* menjadi *beach* ditunjukkan oleh **Gambar 4.18** berikut.



Gambar 4.18 Kesalahan Klasifikasi Kategori *Mountain* menjadi *Beach*

Kesalahan klasifikasi kategori *mountain* menjadi *beach* yang ditampilkan pada **Gambar 4.18** disebabkan karena fitur pegunungan di area latar belakang gambar cukup sulit dibedakan dan cenderung menyatu dengan langit. Selain itu, palet warna kebiruan pada pegunungan disertai warna putih pada awan menyebabkan model membaca data tersebut sebagai fitur-fitur pada kategori *beach* daripada kategori *mountain*. Hal tersebut didukung dengan probabilitas kategori *beach* sebesar 0.725, sedangkan probabilitas kategori *mountain* yaitu sebesar 0.131.

4. Kesalahan Klasifikasi pada Kategori *Museum*

Kesalahan klasifikasi pada kategori *museum* terjadi sebanyak dua kali pada kategori yang sama yaitu *restaurant*. Kedua data yang salah diklasifikasikan ditampilkan pada **Gambar 4.19** berikut.



Gambar 4.19 Kesalahan Klasifikasi Kategori *Museum* menjadi *Restaurant*

Kesalahan klasifikasi yang ditunjukkan **Gambar 4.19** dapat disebabkan karena pada kedua gambar tersebut terdapat fitur yang serupa yaitu lingkaran berwarna putih. Fitur tersebut merupakan salah satu fitur yang banyak terdapat pada data *training* pada kategori *restaurant* yaitu berupa piring. Oleh karena itu, model mengklasifikasikan kedua gambar tersebut kedalam kategori *restaurant* dengan probabilitas sebesar 0.929 untuk **Gambar 4.19** (a) dan 0.763 untuk **Gambar 4.19** (b).

5. Kesalahan Klasifikasi pada Kategori *Restaurant*

Klasifikasi yang dilakukan model GoogLeNet pada kategori *restaurant* mengalami kesalahan sebanyak enam kali pada kategori yang sama yaitu *museum*. Beberapa kesalahan klasifikasi tersebut ditampilkan pada **Gambar 4.20** berikut.



Gambar 4.20 Kesalahan Klasifikasi Kategori *Restaurant* menjadi *Museum*

Kesalahan klasifikasi yang ditampilkan pada **Gambar 4.20** hanya ditampilkan empat contoh, hal tersebut yaitu dikarenakan kesalahan klasifikasi pada dua gambar yang lainnya memiliki penyebab yang sama yaitu kurang terlihatnya fitur restoran pada gambar. Seperti pada **Gambar 4.20 (a)** dan **Gambar 4.20 (b)** dimana fitur restoran seperti piring, gelas, dsb. Tidak begitu terlihat, namun fitur seperti kaca besar yang sering terdapat pada kategori *museum* justru terlihat dominan pada kedua gambar tersebut. Kemudian pada **Gambar 4.20 (c)** eksterior restoran yang mirip seperti gedung museum dapat menjadi penyebab model mengklasifikasikan gambar tersebut kedalam kategori *museum*. Lalu, pada **Gambar 4.20 (d)** fitur kategori *restaurant* juga tidak terlalu tampak. Kursi dan meja menyatu dengan warna keseluruhan restoran yaitu coklat, sehingga fitur-fitur tersebut tidak menonjol dan sulit dikenali sebagai fitur kategori *restaurant*. Probabilitas masing-masing gambar pada kategori *museum* yaitu sebesar 0,994 pada **Gambar 4.20 (a)**, kemudian 0,9 pada **Gambar 4.20 (b)**, lalu 0,893 pada **Gambar 4.20 (c)**, dan 0,724 pada **Gambar 4.20 (d)**.

6. Kesalahan Klasifikasi pada Kategori *Waterfall*

Kesalahan klasifikasi pada kategori *waterfall* terjadi pada 10 data, 9 data diklasifikasikan pada kategori *forest*, sedangkan satu data diklasifikasikan pada kategori *mountain*. **Gambar 4.21** berikut menampilkan kesalahan klasifikasi pada kategori *forest*.



Gambar 4.21 Kesalahan Klasifikasi Kategori *Waterfall* menjadi *Forest*

Kesalahan klasifikasi yang ditampilkan pada **Gambar 4.21** hanya 4 contoh, dikarenakan gambar lainnya memiliki penyebab kesalahan yang sama dengan contoh gambar diatas, yaitu fitur hutan yang mendominasi gambar. Fitur air terjun pada keempat gambar kurang mendominasi, sehingga model mengklasifikasikan gambar-gambar tersebut kedalam kategori *forest*. Probabilitas klasifikasi pada kategori *forest* untuk masing-masing gambar yaitu sebesar 0,681 untuk **Gambar 4.21 (a)**, 0,882 untuk **Gambar 4.21 (b)**, 0,529 untuk **Gambar 4.21 (c)**,

dan 0,693 untuk **Gambar 4.21** (d). Selanjutnya kesalahan klasifikasi kategori *waterfall* menjadi *mountain* ditampilkan pada **Gambar 4.22** berikut.



Gambar 4.22 Kesalahan Klasifikasi Kategori *Waterfall* menjadi *Mountain*

Kesalahan klasifikasi yang terjadi pada **Gambar 4.22** disebabkan karena fitur gunung yang tampak menonjol pada gambar, sedangkan fitur air terjun tampak relatif kecil dan kurang mendominasi. Pengambilan gambar dari jarak jauh menjadi salah satu penyebab fitur air terjun tidak terlalu terlihat, oleh karena itu model GoogLeNet mengklasifikasikan gambar tersebut kedalam kategori *mountain* dengan probabilitas sebesar 0,518, sedangkan kategori *waterfall* memiliki probabilitas sebesar 0,014.

BAB 5 PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang sudah dilakukan didapatkan beberapa kesimpulan sebagai berikut:

1. Penelitian ini menggunakan tiga *pre-trained* model CNN yaitu VGG16, GoogLeNet, dan AlexNet untuk melakukan klasifikasi 6000 foto objek pariwisata yang terbagi kedalam 6 kategori yaitu *beach*, *mountain*, *forest*, *museum*, *restaurant*, dan *waterfall*. Model-model yang akan digunakan dimuat dengan *library* TensorFlow dan PyTorch. Selanjutnya dilakukan *hyperparameter tuning* untuk mendapatkan kombinasi parameter terbaik dari setiap model. Setelah itu model dilakukan evaluasi performa dengan menggunakan metrik akurasi, *loss* baik pada data *training* maupun *data validation*. Setelah didapatkan model dengan perfroma terbaik, selanjutnya model tersebut diuji kembali pada data *test*.
2. Model *Pre-trained CNN* dengan performa terbaik pada penelitian ini yaitu GoogLeNet dengan kombinasi *hyperparameter* yaitu *layer inception block* 5 dalam kondisi *unfreeze*, *learning rate* sebesar 0,0001, *batch size* 32, dan *dropout* 0,5. Kombinasi *hyperparameter* tersebut menghasilkan model dengan akurasi data *training* sebesar 0,9871, *loss* data *training* sebesar 0,0631, akurasi data *validation* sebesar 0,9125, dan *loss* data *validation* sebesar 0,2533. Performa terbaik yang dihasilkan model tersebut salah satunya dikarenakan adanya *inception layer* yang membuat model dapat menangkap fitur pada gambar dengan ukuran *convolutional layer* yang berbeda-beda kemudian disatukan menjadi *output* untuk *layer* selanjutnya. Performa model diuji kembali pada data *test* sebanyak 600 data yang terbagi seimbang pada 6 kategori objek pariwisata. Pada data *test*, model mendapatkan akurasi sebesar 0,9467 dan *loss* sebesar 0,1158. Hasil klasifikasi pada data *test* menunjukkan model GoogLeNet dapat mengklasifikasikan secara tepat sebanyak 568 data dan klasifikasi kurang tepat sebanyak 32 data.

5.2 Saran

Berdasarkan penelitian yang sudah dilakukan, dapat dirumuskan saran untuk beberapa pihak yaitu sebagai berikut:

1. Penelitian ini masih memiliki kekurangan-kekurangan yang dapat diperbaiki untuk penelitian selanjutnya, seperti keterbatasan jumlah data. Penggunaan data yang lebih banyak akan memberikan keuntungan dikarenakan semakin banyak data yang digunakan, maka model akan mempelajari lebih banyak informasi. Selain itu, kualitas data juga perlu menjadi perhatian untuk penelitian selanjutnya agar model dapat melakukan klasifikasi dengan lebih baik untuk data-data pada dari foto wisatawan seperti contohnya terkait pelabelan gambar yang lebih tepat agar model tidak mempelajari fitur gambar yang tidak sesuai dengan kategorinya. Selanjutnya, dalam melakukan proses *training* model, dapat dipertimbangkan untuk menentukan nilai *hyperparameter epoch* dengan nilai yang sama untuk setiap model, hal tersebut bertujuan agar perbandingan performa model menjadi lebih objektif. Kemudian, penelitian selanjutnya dapat mempertimbangkan menggunakan model yang lebih

baru seperti ResNet, CoCa, ModelSoup, dsb. untuk mengetahui model-model dengan performa klasifikasi yang lebih baik dalam kasus foto objek pariwisata.

2. Model klasifikasi dengan performa terbaik yang sudah didapatkan pada penelitian ini dapat menjadi dasar dalam pembentukan sistem rekomendasi objek pariwisata berbasis *fuzzy cluster* bagi pihak-pihak yang bergerak pada bidang pariwisata untuk memberikan rekomendasi objek pariwisata yang lebih tepat bagi para wisatawan.

DAFTAR PUSTAKA

- Afaq, S., & Rao, S. (2020). Significance Of Epochs On Training A Neural Network. *Journal Of Scientific & Technology Research*, 9(06), 485–488. www.ijstr.org
- Agarap, A. F. (2018). *Deep Learning using Rectified Linear Units (ReLU)*. <http://arxiv.org/abs/1803.08375>
- Ali, N., Neagu, D., & Trundle, P. (2019). Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Applied Sciences*, 1(12). <https://doi.org/10.1007/s42452-019-1356-9>
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1). <https://doi.org/10.1186/s40537-021-00444-8>
- Antunes, A., Ferreira, B., Marques, N., & Carriço, N. (2023). Hyperparameter Optimization of a Convolutional Neural Network Model for Pipe Burst Location in Water Distribution Networks. *Journal of Imaging*, 9(3). <https://doi.org/10.3390/jimaging9030068>
- Aszemi, N. M., & Dominic, P. D. D. (2019). Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 10, Issue 6). www.ijacsa.thesai.org
- Brusch, I. (2022). Identification of Travel Styles by Learning from Consumer-generated Images in Online Travel Communities. *Information and Management*, 59(6). <https://doi.org/10.1016/j.im.2022.103682>
- Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2021). *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. <http://arxiv.org/abs/2109.14545>
- Ho, Y., & Wookey, S. (2020). *The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling*.
- Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., & Azim, M. A. (2022). Transfer learning: a friendly introduction. *Journal of Big Data*, 9(1). <https://doi.org/10.1186/s40537-022-00652-w>
- Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. <http://arxiv.org/abs/1502.03167>
- Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. <http://arxiv.org/abs/1412.6980>
- Koo, K. M., & Cha, E. Y. (2017). Image recognition performance enhancements using image normalization. *Human-Centric Computing and Information Sciences*, 7(1). <https://doi.org/10.1186/s13673-017-0114-5>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. <http://code.google.com/p/cuda-convnet/>
- Kurama, V. (2020). *A Review of Popular Deep Learning Architectures: AlexNet, VGG16, and GoogleNet*. <https://Paperspace.Com/Popular-Deep-Learning-Architectures-Alexnet-Vgg-Googlenet/>.







- Kusumanto, R. D., & Tompunu, A. N. (2011). Pengolahan Citra Digital untuk Mendeteksi Obyek menggunakan Pengolahan Warna Model Normalisasi RGB. In *Seminar Nasional Teknologi Informasi & Komunikasi Terapan*.
- Li, F.-F., Johnson, J., & Yeung, S. (2017). <https://www.section.io/engineering-education/image-preprocessing-in-python/>.
- Md Anwar Hossain, B., Shahriar Alam Sajib, M., Anwar Hossain α , M., & Shahriar Alam Sajib σ , M. (2019). *Classification of Image using Convolutional Neural Network (CNN)*.
- MENPAN RB. (2022, December 27). *Hingga Oktober 2022, Jumlah Wisman ke Indonesia Capai 3,92 Juta Orang*. <https://www.menpan.go.id/site/berita-terkini/berita-daerah/hingga-oktober-2022-jumlah-wisman-ke-indonesia-capai-3-92-juta-orang#:~:Text=Sandiaga%20mengungkapkan%2C%20dengan%20data%2Ddata,Yang%20hanya%20%2C40%20persen.>
- Muhammad, A. R., Utomo, H. P., Hidayatullah, P., & Syakrani, N. (2022). Early Stopping Effectiveness for YOLOv4. *Journal of Information Systems Engineering and Business Intelligence*, 8(1), 11–20. <https://doi.org/10.20473/jisebi.8.1.11-20>
- Ponnusamy, R. (2017). A Review of Image Classification Approaches and Techniques. *International Journal of Recent Trends in Engineering and Research*, 3(3), 1–5. <https://doi.org/10.23883/ijrter.2017.3033.xts7z>
- Pramoditha, R. (2021, December 5). *How RGB and Grayscale Images Are Represented in NumPy Arrays*. <https://towardsdatascience.com/exploring-the-mnist-digits-dataset-7ff62631766a>.
- Ramadijanti, N., Karlita, T., Basuki, A., Shabrina, U. I., Afrianto, F., Adiputra, A. A., & Dzalhaqi, M. (2022). Scenery Classification Using Convolutional Neural Network Towards Indonesia Tourism. In *Proceedings of the International Conference on Applied Science and Technology on Social Science 2022 (iCAST-SS 2022)* (pp. 656–660). Atlantis Press SARL. https://doi.org/10.2991/978-2-494069-83-1_114
- Rochman, F., & Junaedi, H. (2020). Implementasi Transfer Learning untuk Identifikasi Ordo Tumbuhan Melalui Daun. *Journal Syntax Admiration*, 1(6), 672–679.
- Sa'idah, S., Suparta, I., & Suhartono, E. (2022). Modifikasi Convolutional Neural Network Arsitektur GoogLeNet dengan Dull Razor Filtering untuk Klasifikasi Kanker Kulit. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, 11(2), 148–153.
- Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation Functions in Neural Networks. In *International Journal of Engineering Applied Sciences and Technology* (Vol. 4). <http://www.ijeast.com>
- Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. <http://arxiv.org/abs/1409.1556>
- Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *Journal of Machine Learning Research* (Vol. 15).
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions*. <http://arxiv.org/abs/1409.4842>

- Tamma, S. (2019). Transfer Learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications*, 9(10), 143–150. <https://doi.org/10.29322/IJSRP.9.10.2019.p9420>
- UNWTO. (2019). International Tourism Highlights, 2019 Edition. In *International Tourism Highlights, 2019 Edition*. World Tourism Organization (UNWTO). <https://doi.org/10.18111/9789284421152>
- Xiang, Z., & Fesenmaier Editors, D. R. (2017). *Tourism on the Verge Analytics in Smart Tourism Design Concepts and Methods*. <http://www.springer.com/series/13605>
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). *How transferable are features in deep neural networks?* <http://arxiv.org/abs/1411.1792>







(Halaman Ini Sengaja Dikosongkan)

LAMPIRAN

Lampiran 1. Data *Training* dan Data *Validation* dari Places365

No.	Foto (X)	Kategori Objek Wisata (Y)
1		Beach
2		Forest
3		Mountain
4		Restaurant
5		Museum
⋮	⋮	⋮
6000		Beach

Lampiran 2. Data *Testing* dari Google Images

No.	Foto (X)	Kategori Objek Wisata (Y)
1		Forest
2		Museum
3		Waterfall
4		Mountain
5		Museum
:	:	:
600		Beach

Lampiran 3. Syntax untuk *Import Libraries* yang Dibutuhkan

```
#ignore the warning (mostly unrequired info)
import warnings
warnings.filterwarnings('ignore')

#data visualization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import style

#configure matplotlib (set to inline & display graph below corresponding cell)
%matplotlib inline
style.use('default')
sns.set(style='whitegrid', color_codes=True)

#model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import
accuracy_score, precision_score, recall_score, confusion_matrix, roc_curve, roc_auc_sc
ore
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder

#preprocess
from keras.preprocessing.image import ImageDataGenerator

#dynamically loaded library
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.utils import to_categorical
from keras.callbacks import ReduceLROnPlateau

#specific for cnn
from keras.layers import Dropout, Flatten, Activation
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
import tensorflow as tf
import random as rn

#specific for manipulating images
import cv2
import os
from tqdm import tqdm
from random import shuffle
from PIL import Image

#VGG16
from keras.applications.vgg16 import VGG16
```

Lampiran 4. Syntax untuk Membaca Dataset

```
#mounted to Gdrive
from google.colab import drive
drive.mount('/content/drive')

#list of data categories
print(os.listdir('/content/drive/MyDrive/TA2/DataMedium'))

#get training and validation set from images
X=[]
y=[]
IMG_SIZE=224
beach_dir = '/content/drive/MyDrive/TA2/DataMedium/beach'
mountain_dir = '/content/drive/MyDrive/TA2/DataMedium/mountain'
restaurant_dir = '/content/drive/MyDrive/TA2/DataMedium/restaurant'
waterfall_dir = '/content/drive/MyDrive/TA2/DataMedium/waterfall'
museum_dir = '/content/drive/MyDrive/TA2/DataMedium/museum'
forest_dir = '/content/drive/MyDrive/TA2/DataMedium/forest'

#function to add image label
def assign_label(img,tourism_type):
    return tourism_type

#function to add dataset for training
def make_train_data(tourism_type, DIR):
    for img in tqdm(os.listdir(DIR)):
        label = assign_label(img, tourism_type)
        path = os.path.join(DIR, img)
        img = cv2.imread(path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        X.append(np.array(img))
        y.append(str(label))

#train data for beach
make_train_data('beach', beach_dir)
print(len(X))
.....
.....
.....
.....

#train data for forest
make_train_data('forest', forest_dir)
print(len(X))
```


Lampiran 5. Syntax untuk Statistika Deskriptif

```
#histogram of the sample images
categories = ['beach', 'mountain', 'restaurant', 'waterfall', 'museum',
'forest']

plt.figure(figsize=(15, 10))

for i, category in enumerate(categories):
    dir_path = eval(category + '_dir')
    img_paths = [os.path.join(dir_path, img) for img in os.listdir(dir_path)]

    img_path = rn.choice(img_paths)
    label = assign_label(img_path, category)

    img = cv2.imread(img_path, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    X.append(np.array(img))
    y.append(str(label))

#display RGB histograms
plt.subplot(2, 3, i+1)
plt.hist(img[:, :, 0].ravel(), bins=256, color='red', alpha=0.5)
plt.hist(img[:, :, 1].ravel(), bins=256, color='green', alpha=0.5)
plt.hist(img[:, :, 2].ravel(), bins=256, color='blue', alpha=0.5)
plt.title(category.capitalize() + ' Channel')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

#show some sample images
if len(X) >= 6:
    fig, ax = plt.subplots(2, 3, figsize=(8, 8))

    for i in range(2):
        for j in range(3):
            l = rn.randint(0, len(y) - 1)
            ax[i, j].imshow(X[l])
            ax[i, j].set_title('Tourism Type: ' + y[l])
else:
    print("Insufficient images loaded to create subplots.")
```

Lampiran 6. Syntax untuk Data Preprocessing dan Transformasi Data

```
#data normalization
X = [x / 255.0 for x in X]
#convert dataset to NumPy arrays
#define the tourism types
tourism_types = ['beach', 'mountain', 'restaurant', 'waterfall', 'museum',
'forest']

#convert X and y to NumPy arrays
X = np.array(X)
y = np.array(y)

#convert y to numeric labels using integer encoding
label_mapping = {label: index for index, label in enumerate(tourism_types)}
y_encoded = np.array([label_mapping[label] for label in y])

#convert y to one-hot encoded format
y_one_hot = to_categorical(y_encoded, num_classes=len(tourism_types))
```

Lampiran 7. Syntax untuk Split Dataset menjadi Data Train dan Data Validation

```
#split the data into training and validation sets
X_train, X_test, y_train, y_test = train_test_split(X, y_one_hot, test_size=0.2,
random_state=42)

#calculate the sizes
train_size = len(X_train)
test_size = len(X_test)

#create the bar chart
labels = ['Training', 'Validation']
sizes = [train_size, test_size]

plt.bar(labels, sizes)
plt.xlabel('Dataset')
plt.ylabel('Size')

#show the chart
plt.show()
```

Lampiran 8. Syntax untuk Memuat *Pre-Trained CNN Model*

```
#define the model
VGG16_model = VGG16(include_top=False, weights='imagenet', input_shape=(224,
224, 3), pooling='avg')

#model summary
VGG16_model.summary()

#add fully connected layers
model=Sequential()
model.add(VGG16_model)
model.add(Dense(4096, activation='relu'))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(6, activation='softmax'))
```

Lampiran 9. Syntax untuk *Hyperparameter Tuning*

```
#set batch size
batch_size=64

#set the VGG model to be untrainable.
VGG16_model.trainable=False

#check trainable layers
for layer in VGG16_model.layers:
    print(layer.name, layer.trainable)

#compile model, import adam, also set the learning rate
model.compile(optimizer=Adam(lr=1e-4), loss='categorical_crossentropy',
metrics=['accuracy'])

#early stopping criteria
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

#add dropout layer
model.add(Dropout(0.5))
```

Lampiran 10. *Syntax* untuk Model *Training* serta Evaluasi Performa Model

```
#training and evaluate model
VGG16_classify = model.fit(X_train, y_train, batch_size=batch_size, epochs=50,
validation_data=(X_test, y_test), callbacks=[early_stopping])

#plot the accuracy result
train_accuracy = VGG16_classify.history['accuracy']
val_accuracy = VGG16_classify.history['val_accuracy']

plt.plot(train_accuracy)
plt.plot(val_accuracy)
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()

#plot thw loss result
train_loss = VGG16_classify.history['loss']
val_loss = VGG16_classify.history['val_loss']

plt.plot(train_loss)
plt.plot(val_loss)
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```

Lampiran 11. *Syntax* untuk Memuat dan Mengaplikasikan Model Terbaik

```
#path to the model
model_path = '/content/drive/MyDrive/TA2/Code/SavedModel/best_GoogLeNet.pt'

#load the saved model
model = googlenet()
#modify the last fully connected layer
num_classes = 6
dropout_rate = 0.5
model.dropout.p = dropout_rate
model.fc = nn.Linear(1024, num_classes)

#move the model to GPU
model.to(device)

model.load_state_dict(torch.load(model_path), strict=False)
model.eval()

#create a data loader for the new dataset
test_dataset_loader = torch.utils.data.DataLoader(dataset, batch_size=32,
shuffle=False)

#initialize lists to collect predicted labels and ground truth labels
predicted_labels = []
ground_truth_labels = []
predicted_probabilities = []

with torch.no_grad():
    for inputs, labels in test_dataset_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        #compute predicted probabilities using softmax
        softmax = nn.Softmax(dim=1)
        probabilities = softmax(outputs)

        predicted_labels.extend(predicted.cpu().numpy())
        ground_truth_labels.extend(labels.cpu().numpy())
        predicted_probabilities.extend(probabilities.cpu().numpy())

#convert predicted_probabilities to a numpy array
predicted_probabilities = np.array(predicted_probabilities)

#generate the confusion matrix
confusion_mat = confusion_matrix(ground_truth_labels, predicted_labels)

print(confusion_mat)
```

Lampiran 12. Syntax evaluasi Performa Model Terbaik

```
#make the confusion matrix to a plot
import seaborn as sns

#define the class labels
class_labels = ['beach', 'forest', 'mountain', 'museum', 'restaurant',
                'waterfall']

#call the confusion matrix
cf = confusion_mat

#create a figure and axis
fig, ax = plt.subplots(figsize=(8, 6))

#create the heatmap using seaborn
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues', cbar=False, ax=ax)

#set axis labels and title
ax.set_xlabel('Predicted Labels')
ax.set_ylabel('True Labels')
ax.set_title('Confusion Matrix')

#set x-axis tick labels
ax.set_xticklabels(class_labels, rotation=45, ha='right')
ax.set_yticklabels(class_labels, rotation=0)

#show the plot
plt.show()

#performance in Data Testing
from sklearn import metrics
print("Precision, Recall, Confusion matrix, in training\n")
print(metrics.classification_report(predicted_labels, ground_truth_labels,
                                     digits=4))
```


Lampiran 13. *Syntax* untuk Menampilkan Klasifikasi yang Tidak Tepat

```
#find wrong prediction
wrong_prediction_indices = np.where(np.array(predicted_labels) !=
np.array(ground_truth_labels))[0]

#randomize the wrong predictions
np.random.shuffle(wrong_prediction_indices)

#select random 6 wrong predictions
num_samples = min(6, len(wrong_prediction_indices))
random_indices = wrong_prediction_indices[:num_samples]

#create subplot 2x3
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))

#iteration over wrong images
for i, idx in enumerate(random_indices):
    #get the image, label, predicted label, ground truth label, and
    #probabilities for the current index
    image, label = dataset[idx]
    predicted_label = predicted_labels[idx]
    ground_truth_label = ground_truth_labels[idx]
    probabilities = predicted_probabilities[idx]

    #convert the image tensor to a numpy array and transpose it
    image = image.permute(1, 2, 0).numpy()
    image = (image - np.min(image)) / (np.max(image) - np.min(image))

    #plot the image in the corresponding subplot
    row = i // 3
    col = i % 3
    ax = axes[row, col]
    ax.imshow(image)
    ax.set_title(f'Predicted: {label_map[predicted_label]}\nGround Truth:
{label_map[ground_truth_label]}', loc='right')
    ax.axis('off')

    #create a string for probabilities and labels
    labels_with_probabilities = [f'{label_map[j]}: {prob:.3f}' for j, prob in
enumerate(probabilities)]
    labels_with_probabilities = '\n'.join(labels_with_probabilities)

    #display the predicted label and probabilities
    ax.text(1.02, 0.5, labels_with_probabilities, fontsize=9,
verticalalignment='baseline')

#adjust the spacing between subplots
plt.tight_layout()

#show the plot
plt.show()
```

Lampiran 14. Surat Pernyataan Data

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FSAD ITS:

Nama mahasiswa : Akhmad Miftakhul Ilmi

NRP : 06211940000062

menyatakan bahwa data yang digunakan dalam Tugas Akhir dengan Judul “ Klasifikasi Objek Pariwisata Melalui Unggahan Foto di Media Sosial dengan Metode Pre-Trained Convolutional Neural Network” ini merupakan data sekunder yang diambil dari ~~penelitian/buku/Tugas Akhir/Thesis/~~ publikasi lainnya* yaitu:

Sumber : Data *website*

1. <http://places2.csail.mit.edu/download.html>
2. <https://images.google.com/>

Keterangan :

1. Data gambar objek pariwisata di seluruh dunia diambil 6 kategori.
2. Data gambar objek pariwisata di Indonesia diambil 6 kategori.

Surat Pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data maka saya siap menerima sanksi sesuai aturan yang berlaku.

Surabaya, 01 Juli 2023

Mengetahui,
Dosen Pembimbing Tugas Akhir

Mahasiswa



Dr. Dra. Kartika Fithriasari, M.Si.
NIP. 19691212 199303 2 002



Akhmad Miftakhul Ilmi
NRP. 06211940000062

*(coret yang tidak perlu)

BIODATA PENULIS



Penulis dilahirkan di Malang, pada 19 Maret 1999, merupakan anak pertama dari dua bersaudara. Penulis berasal dari Desa Pandesari, Kecamatan Pujon, Kabupaten Malang. Penulis mengawali pendidikan formal di SDN Pandesari 03, kemudian dilanjutkan ke jenjang menengah pertama di MTs.TMI Pujon, lalu penulis melanjutkan ke jenjang menengah atas di MAN Kota Batu. Setelah itu penulis melanjutkan pendidikan ke jenjang S1 di Departemen Statistika ITS pada tahun 2019 melalui jalur SBMPTN.

Selama berkuliah di ITS, penulis aktif dalam berbagai kegiatan baik kegiatan akademik maupun kegiatan non-akademik. Aktivitas yang diikuti penulis dalam kategori akademik diantaranya kompetisi dengan beberapa prestasinya yaitu Juara I Business Plan Competition oleh Universitas Negeri Padang, Juara 3 Lomba Karya Tulis Ilmiah oleh Univet Sukoharjo, Gold Medal dan Bronze Medal Paper Award FTP Universitas Brawijaya, Gold Medal ASEAN Innovative Science Environmental and Entrepreneur Fair (AISEEF), Silver Medal World Science Environment and Engineering Competition (WSEEC), serta Juara I Track AML Hackathon PPATK 2023. Sedangkan dalam bidang non-akademik penulis pernah tergabung dalam organisasi Badan Eksekutif Mahasiswa Fakultas Sains dan Analitika Data periode 2021-2022 sebagai staff divisi riset aplikatif, departemen ristek. Kemudian penulis juga pernah menjadi panitia lomba Data Analisis Competition tahun 2021 menjabat sebagai wakil ketua divisi acara. Selain itu, penulis juga pernah mengikuti program pertukaran pelajar secara virtual dengan Asia University Taiwan pada tahun 2021. Bagi pembaca yang ingin berdiskusi, memberikan kritik, serta masukan terkait Tugas Akhir ini dapat menghubungi penulis melalui email: takhulilmi99@gmail.com