TESIS – ES235401

INTEGRASI K-MEANS CLUSTERING PADA LIME EXPLAINABLE AI UNTUK MENINGKATKAN INTERPRETABILITAS MODEL DEEP LEARNING PADA KLASIFIKASI BOX PALETTE

EVAN RADITYA
6026231037

DOSEN PEMBIMBING
Dr. Rarasmaya Indraswari, S.Kom.
1995202012057

DEPARTEMEN SISTEM INFORMASI
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2024

THESIS – ES235401

INTEGRATION OF K-MEANS CLUSTERING IN LIME EXPLAINABLE AI TO IMPROVE THE INTERPRETABILITY OF DEEP LEARNING MODELS IN BOX PALETTE CLASSIFICATION

EVAN RADITYA
6026231037

SUPERVISOR
Dr. Rarasmaya Indraswari, S.Kom.
1995202012057

DEPARTMENT OF INFORMATION SYSTEMS
FACULTY OF INTELLIGENT ELECTRICAL AND INFORMATICS TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
2024

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sistem Informasi (M.Kom.)
di
Institut Teknologi Sepuluh Nopember

Oleh:
Evan Raditya
NRP: 6026231037

Tanggal Ujian: 26 Juli 2024
Periode Wisuda ITS: 130

Disetujui oleh:
Pembimbing:

Dr. Rarasmaya Indraswari, S.Kom
NIP: 1995202012057

Penguji:

Prof. Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D
NIP: 198201202005012001

Dr. Ir. Aris Tjahyanto, M.Kom
NIP: 196503101991021001

Surabaya, 02 Agustus 2024
Kepala Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas

LP/P/24/184
DSI-130-F-23-267/9

Dr. Mudjahidin, ST, MT
NIP: 197010102003121001

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP	:	Evan Raditya / 6026231037
Program studi	:	S2 Sistem Informasi
Dosen Pembimbing / NIP	:	Dr. Rarasmaya Indraswari, S.Kom / 1995202012057

dengan ini menyatakan bahwa Tesis dengan judul "INTEGRASI K-MEANS CLUSTERING PADA LIME EXPLAINABLE AI UNTUK MENINGKATKAN INTERPRETABILITAS MODEL DEEP LEARNING PADA KLASIFIKASI BOX PALETTE" 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 Agustus 2024

Mengetahui

Dosen Pembimbing

Dr. Rarasmaya Indraswari, S.Kom
NIP. 1995202012057

Mahasiswa

Evan Raditya
NRP. 6026231037

INTEGRASI K-MEANS CLUSTERING PADA LIME EXPLAINABLE AI UNTUK MENINGKATKAN INTERPRETABILITAS MODEL DEEP LEARNING PADA KLASIFIKASI BOX PALETTE

Nama Mahasiswa : Evan Raditya
NRP : 6026231037
Dosen Pembimbing 1 : Dr. Rarasmaya Indraswari, S.Kom.

ABSTRAK

Dalam sistem produksi bahan pangan, penyusunan box palette secara manual sering mengalami kesalahan, seperti pola susunan yang salah, jumlah box yang tidak sesuai, dan campuran varian produk yang tidak seharusnya. Masalah ini terjadi pada perusahaan produksi kecap yang memiliki keterbatasan infrastruktur, sehingga dua varian produk dengan ukuran box yang sama tercampur dalam satu jalur produksi. Hal ini mengakibatkan gangguan pada alur produksi dan potensi kerugian yang signifikan.

Penelitian ini mengusulkan solusi dengan menggunakan deep learning untuk mendeteksi pola susunan box palette. Metode *Convolutional Neural Network* (CNN) dipilih karena telah terbukti efektif dalam klasifikasi gambar. Selain itu, penelitian ini juga akan mengimplementasikan *Explainable AI* (XAI) untuk memberikan penjelasan terkait hasil klasifikasi, meningkatkan kepercayaan pengguna terhadap sistem. Teknik Local Interpretable Model-agnostic Explanations (LIME) akan digunakan untuk memberikan interpretasi, namun dengan penambahan *K-Means Clustering* untuk meningkatkan stabilitas hasil penjelasan.

Dari penelitian ini didapatkan bahwa integrasi K-Means Clustering pada LIME dapat meningkatkan skor *fidelity* dan menjaga *stability* dari penjelasan LIME.

Kata kunci: sistem deteksi; image classification; convolutional neural network; explainable ai; box palette; k-means clustering

INTEGRATION OF K-MEANS CLUSTERING IN LIME EXPLAINABLE AI TO IMPROVE THE INTERPRETABILITY OF DEEP LEARNING MODELS IN BOX PALETTE CLASSIFICATION

Student Name : Evan Raditya
NRP : 6026231037
Advisor : Dr. Rarasmaya Indraswari, S.Kom.

ABSTRACT

In the food production system, manual box palette stacking often encounters errors such as incorrect stacking patterns, mismatched box quantities, and the mixing of product variants that should not be combined. This issue occurs in a soy sauce production company with limited infrastructure, where two product variants with the same box size are mixed in a single production lane. This results in production flow disruptions and significant potential losses.

This research proposes a computer-based solution using deep learning to detect box palette stacking patterns. The Convolutional Neural Network (CNN) method is chosen due to its proven effectiveness in image classification. Additionally, this study will implement Explainable AI (XAI) to provide explanations related to classification results, enhancing user trust in the system. The Local Interpretable Model-agnostic Explanations (LIME) technique will be used for interpretation, with the addition of K-Means Clustering to improve the stability of the explanations.

From this research, it was found that the integration of K-Means Clustering in LIME can increase the fidelity score and maintain the stability of the LIME explanation.

Keywords: detection system; image classification; convolutional neural networks; explainable ai; palette box; k-means clustering

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dalam sebuah sistem produksi sebuah pabrik, biasanya dilakukan penyusunan hasil produksi dari pabrik tersebut dalam bentuk susunan box. Hal ini biasa disebut dengan *palettizing* atau penyusunan *box palette*. Penyusunan box palette biasanya dilakukan pada berbagai macam sistem produksi, misalnya sistem produksi bahan pangan. Saat ini sistem penyusunan *box palette* dapat dilakukan secara manual atau secara otomatis. Pada sebuah sistem produksi produk bahan pangan yang menggunakan tenaga manual, sering terdapat kesalahan dalam penyusunan box. Umumnya, masalah yang sering terjadi adalah salah pola susunan, jumlah box kurang/tidak sesuai, dan varian lain yang terselip dalam susunan box. Kesalahan-kesalahan seperti ini dapat menghambat alur dari sistem produksi. Oleh karena itu, diperlukan sebuah solusi yang dapat membantu mengatasi kesalahan pada penyusunan box.

Pada studi kasus penelitian ini, perusahaan produksi kecap memiliki permasalahan serupa, yaitu sering terjadi kesalahan pada penyusunan *box palette*. Permasalahan ini muncul karena dua varian produk, yaitu varian 60 ml dan 63 ml tercampur pada lane produksi. Hal ini disebabkan karena pada perusahaan ini, terdapat keterbatasan infrastruktur sehingga hanya terdapat satu jalur (lane) produksi. Selama ini, perusahaan melakukan penyaringan untuk tiap varian produknya dengan menggunakan balok besi untuk varian dengan ukuran box berbeda. Sedangkan dua varian 60 ml dan 63 ml tersebut, memiliki ukuran box yang sama sehingga tersaring menjadi satu. Karena hal ini, sering terjadi kesalahan manusia pada proses penyusunan box palette. Akibatnya, perusahaan produksi kecap mengalami hambatan pada sistem produksinya karena harus menata ulang susunan box palette ketika terjadi kesalahan. Hambatan ini jika tidak diatasi maka dapat mengakibatkan kerugian yang cukup signifikan bagi perusahaan.

Salah satu solusi yang dapat dilakukan adalah dengan membuat sistem deteksi berbasis komputer untuk mendeteksi pola susunan pada box susunan. Sistem deteksi ini dapat dibuat dengan menggunakan *image classification*. *Image*

classification digunakan dalam sistem karena dapat digunakan untuk mengklasifikasikan perbedaan dari tiap pola susunan yang ada. Salah satu metode image classification yang dapat digunakan dalam permasalahan ini adalah dengan menggunakan deep learning.

Saat ini, deep learning dalam image classification sudah banyak digunakan dan dikembangkan. Selain itu, karena perkembangannya sudah banyak, deep learning memiliki performa yang akurat dalam melakukan image classification. Contohnya adalah penggunaan metode deep learning dengan arsitektur Convolutional Neural Network (CNN) dalam melakukan klafisikasi gambar. Hal ini diperkuat dengan adanya berbagai penelitian yang mengimplementasikan CNN untuk berbagai tugas klasifikasi. Salah satu contoh adalah implementasi CNN untuk tugas klasifikasi mineral (Liu *et al.*, 2023). Penelitian tersebut membahas berbagai aplikasi dan penelitian terkait klasifikasi gambar menggunakan CNN untuk eksplorasi mineral. CNN memberikan cara baru yang kuat untuk melakukan analisis terhadap gambar-gambar terkait eksplorasi mineral. Selain itu, CNN memiliki potensi untuk tugas lain yang berhubungan, seperti pada industri pertambangan dan deteksi objek mineral. Contoh lainnya adalah penggunaan CNN untuk tugas klasifikasi sel darah putih (Bhatia *et al.*, 2023). Pada penelitian ini digunakan dua dataset yaitu BCCD dan Raabin-WBC. Dari penelitian ini, didapatkan bahwa klasifikasi menggunakan model CNN DenseNet121 menghasilkan performa yang baik, yaitu dengan nilai akurasi 95,87% pada dataset BCCD dan nilai akurasi 98,59%, pada dataset Raabin-WBC.

Namun, dalam perkembangannya, deep learning bekerja secara black box. Oleh karena itu muncul perkembangan baru yaitu penggunaaan explainable AI (XAI) pada algoritma deep learning yang digunakan untuk image classification. XAI merupakan salah satu perkembangan dalam machine learning yang bertujuan untuk menjelaskan alasan dan memberikan konteks terkait output sebuah model deep learning. Contohnya adalah implementasi XAI dalam bentuk *Local Interpretable Model-agnostic Explanations* (LIME) pada klasifikasi sel darah putih yang menggunakan *deep learning* (Bhatia *et al.*, 2023). Pada penelitian ini, dilakukan perbandingan lima model deep learning pada dua dataset sel darah putih. Hasil dari penelitian ini menunjukkan bahwa XAI (LIME) mampu menjelaskan

alasan dari hasil prediksi klasifikasi model, karena model dengan akurasi yang tinggi belum tentu memberikan hasil prediksi yang tepat. Oleh karena itu, dengan hasil XAI, dapat ditentukan bagian mana dari sebuah gambar yang menjadi acuan dari sebuah model untuk menghasilkan prediksinya. Namun, dalam implementasinya, LIME juga memiliki masalah hasil penjelasan yang kurang stabil dan konsisten. Untuk mengatasi hal ini, penelitian oleh (Zafar and Khan, 2019) membahas salah satu solusi mengatasi ketidakstabilan LIME dengan menggunakan *Agglomerative Hierarchical Clustering* untuk pengelompokan dataset yang digunakan LIME. Hasil penelitian ini menunjukkan bahwa stabilitas meningkat dengan metode clustering yang diajukan. Namun jumlah sampel pada dataset mempengaruhi cluster yang dihasilkan, sehingga juga mempengaruhi stabilitas dan akurasi prediksi LIME. Penelitian ini juga tidak meneliti pengaruh *clustering* pada metrik *fidelity* yang menandakan kemampuan model LIME dalam menginterpretasi model asli (Shi, Du and Fan, 2020). Selain itu penelitian dengan *clustering* baru diterapkan pada data tabular, dan belum diterapkan pada data lain seperti data gambar atau teks.

Saat ini telah banyak penelitian yang membahas terkait implementasi deep learning untuk mengklasifikasi gambar pola. Namun, penelitian yang khusus membahas tentang klasifikasi pola susunan box pada sebuah sistem produksi, khususnya produksi bahan pangan cukup jarang ditemukan. Oleh karena itu, penelitian ini akan berkontribusi dalam pengembangan sistem deteksi untuk mengklasifikasi pola susunan box palette dengan menggunakan CNN. Selain itu penelitian ini juga akan menerapkan *explainable AI* (XAI) sebagai salah satu cara untuk meningkatkan kepercayaan kepada sistem, khususnya pada sistem yang banyak berinteraksi dengan pengguna sistem. Dengan adanya XAI, maka hasil klasifikasi dapat diberi penjelasan sehingga pengguna awam sekalipun dapat mengetahui alasan dan konteks dari hasil klasifikasi yang muncul. Selain itu, pada penelitian ini juga dilakukan upaya untuk meningkatkan interpretabilitas dan *fidelity* dari hasil penjelasan LIME dengan menggunakan *K-Means Clustering* sebagai alternatif dari metode *Agglomerative Hierarchical Clustering* yang telah diusulkan pada penelitian terdahulu. Penelitian ini juga dilakukan dengan menggunakan data gambar. Dengan penelitian ini, diharapkan dapat dihasilkan

sebuah metode yang dapat dipercaya dan dapat mengklasifikasikan susunan pola box untuk mengatasi kesalahan pada penyusunan box. Selain itu, penelitian ini juga diharapkan dapat memberikan pengetahuan baru terkait implementasi XAI pada sebuah sistem klasifikasi yang menggunakan CNN.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang, rumusan masalah yang ada pada penelitian tesis ini adalah:

1. Bagaimana melakukan klasifikasi box palette menggunakan model deep learning?
2. Bagaimana mengintegrasikan k-means clustering pada LIME XAI?
3. Bagaimana melakukan penilaian interpretabilitas model deep learning menggunakan LIME XAI yang diintegrasikan dengan K-means clustering?

1.3 Tujuan Penelitian

Tujuan akhir dari penelitian tugas akhir ini adalah:

1. Mengetahui bagaimana penambahan *K-Means Clustering* dapat meningkatkan performa penjelasan LIME.
2. Mengetahui arsitektur model CNN mana yang memiliki performa terbaik untuk klasifikasi susunan box palette.
3. Mengetahui interpretabilitas melalui nilai metrik *fidelity* dan *stability* dari LIME yang sudah dimodifikasi.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian tesis adalah sebagai berikut:

1. Bagi peneliti, untuk mengetahui cara pengembangan model CNN dengan XAI yang dapat digunakan untuk melakukan deteksi pola susunan box palette pada sebuah sistem produksi.
2. Bagi perusahaan produksi, untuk mendapat solusi ramah biaya berupa sistem deteksi susunan box untuk mengatasi permasalahan yang dialami ketika menggunakan tenaga manual pada sistem produksi.

3. Bagi masyarakat, untuk memberikan pengetahuan terkait penerapan ilmu *deep learning (CNN)* serta XAI dalam sistem deteksi pola susunan box.

1.5 Batasan Penelitan

Batasan yang ada pada penelitian ini adalah sebagai berikut:

1. Dataset yang digunakan adalah data pola susunan box yang dikumpulkan dari sebuah perusahaan produksi kecap.
2. Dataset terdiri dari 600 gambar.
3. Sistem klasifikasi pola akan mengklasifikasikan pola menjadi dua varian, yaitu varian kemasan 60 ml dan varian kemasan 63 ml
4. Sistem deteksi yang dikembangkan berfokus untuk mengklasifikasi pola susunan berdasarkan variasi produk.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan dijelaskan terkait kajian pustaka yang merupakan rangkuman dari penelitian-penelitian terkait sebelumnya. Selain itu, akan dijelaskan juga terkait dasar teori untuk menunjang pengerjaan tugas akhir ini.

2.1 Kajian Pustaka

Judul	**Dataset**	**Metode**	**Hasil**
Integrating explainability into deep learning-based models for white blood cells classification (Bhatia *et al.*, 2023)	BCCD dataset Raabin-WBC dataset	-Klasifikasi menggunakan model DenseNet121, ResNet50, Xception, MobileNetV2, and VGG16 -Evaluasi XAI dengan menggunakan LIME -Metrik pengukuran: *accuracy*, *precision*, *recall*, *specificity* dan F1-score	-Mengintegrasikan teknik AI yang dapat dijelaskan (LIME) dengan model klasifikasi, yang dapat memberikan hasil yang dapat ditafsirkan dan transparan bagi pengguna, sehingga meningkatkan kepercayaan diri dan kepercayaan mereka terhadap diagnosis otomatis. -Hasil LIME dapat menyempurnakan model yang telah dilatih sebelumnya pada set data spesifik domain yang lebih kecil, yang dapat mengurangi sumber daya komputasi dan waktu yang diperlukan untuk melatih model dari awal.
Skin cancer classification using explainable artificial intelligence on	Dataset PH2, yang berisi 200 gambar dermoskopi lesi kulit	-Menggunakan SHAP sebagai metode XAI -Klasifikasi menggunakan model	-Terdapat tiga bagian dalam hasil XAI: *permutation importance*, plot dependensi parsial (PDP), hasil *Shapley*

pre-extracted image features (Khater *et al.*, 2023)	dengan tujuh fitur input dan satu fitur output.	XGBoost, *decision tree, random forest,* dan KNN -evaluasi performa model dan pentingnya fitur pada gambar dengan menggunakan SHAP	-Permutation importance untuk mengukur pentingnya sebuah fitur gambar dengan mengevaluasi perubahan pada error prediksi model setelah permutasi. -Plot dependensi parsial (PDP) untuk menampilkan pengaruh marjinal variabel prediktor, yaitu pengaruh rata-ratanya di seluruh kumpulan data, terhadap variabel target. -Hasil Shapley yang menunjukkan bahwa SHAP *(Shapley Additive Explanations)* dapat menghasilkan penjelasan lokal dengan melokalisasi model menggunakan model yang lebih kecil dan mengganggu data input untuk mengamati bagaimana output berubah. -Selain itu, XAI menunjukkan bahwa fitur asimetri dan jaringan pigmen adalah fitur

			paling penting dalam klasifikasi kanker kulit.
XAI for myo-controlled prosthesis: Explaining EMG data for hand gesture classification (Gozzi *et al.*, 2022)	EMG Dataset	-Menggunakan kombinasi antara metode *machine learning* klasikal dan *deep learning*. -Untuk metode klasikal, diekstrak 15 fitur statistik dan fitur berbasis domain dari setiap sinyal EMG dan melatih algoritma pembelajaran mesin yang berbeda, seperti LDA, SVM, dan XRT. -Untuk deep learning, digunakan model CNet1D dan CNet2D, yang mengambil sinyal EMG mentah sebagai input dan melakukan pembelajaran fitur dan klasifikasi. -Metode XAI yang digunakan pada	-Berhasil mengurangi jumlah elektroda yang perlu dipasang pada tubuh sehingga sistemnya menjadi lebih nyaman untuk digunakan. -XAI yang diimplementasikan berhasil mengidentifikasi error dan inkonsistensi pada peletakkan elektroda dan kualitas data. -Dengan menggunakan hasil XAI, peneliti berhasil melakukan pengurangan fitur untuk meningkatkan efisiensi performa klasifikasi.

		penelitian ini adalah SHAP untuk model machine learning klasikal dan Grad-CAM untuk model *Convolutional Neural Network* berbasis sinyal EMG	
Example-based explainable AI and its application for remote sensing image classification (Ishikawa *et al.*, 2023)	Dataset EuroSAT dari satelit Sentinel-2	-Diusulkan sebuah metode XAI bernama "What I Know" (WIK) untuk memverifikasi reliabilitas dari sebuah model *deep learning*. -WIK mengevaluasi kemiripan berdasarkan fitur yang diekstrak oleh model, bukan data input, untuk menentukan apakah data pelatihan cukup mewakili data target. -Kemiripan diukur dengan *L2-norm* dari vektor-vektor dari lapisan terakhir	-Penelitian menunjukkan bahwa WIK sebagai salah satu contoh metode XAI memiliki aplikasi di beberapa area utama ketika penjelasan yang dapat diandalkan sangat dibutuhkan untuk menghindari kesalahan dalam pengambilan keputusan. Hal ini membantu pengguna memahami proses pengambilan keputusan model AI dan menumbuhkan kepercayaan pada sistem AI.

		sebelum lapisan keluaran model, yang mewakili fitur-fitur yang diekstraksi untuk tugas klasifikasi	
Logistics box recognition in robotic industrial de-palletising procedure with systematic RGB-D image processing supported by multiple deep learning methods (Yoon, Han and Nguyen, 2023)	Dataset yang digunakan terdiri dari 435 kotak dengan dimensi yang bervariasi, yang diambil dalam 2075 gambar.	-Diajukan metode deteksi susunan box berbasis *Mask R-CNN* yang didukung dengan *Cycle Generative Adversarial Network* (GAN). -Cycle-GAN untuk mengoptimalkan permukaan luar kotak dengan secara otomatis menghapus tag, stiker, label, dan simbol yang ada pada kotak -Kemudian gambar diproses menggunakan Mask R-CNN	-Performa deteksi box menjadi meningkat. Peningkatkan ini divalidasi dengan menggunakan masing-masing 200 kasus pengaturan kotak yang teratur dan tidak teratur. -Evaluasi juga dilakukan dengan mengukur *mean absolute error* (MAE) antara titik pengambilan yang diprediksi dan nilai *ground truth* untuk kasus uji coba dalam proses implementasi.
Optimizing LIME Explanations Using REVEL	CIFAR10	-Klasifikasi dengan menggunakan model efficientnetb2	-Hasil evaluasi REVEL dapat menentukan parameter LIME yang optimal agar hasil LIME

Metrics (Sevillano-Garcia, Luengo and Herrera, 2023)		-Menggunakan LIME sebagai metode XAI -Menggunakan REVEL untuk evaluasi hasil LIME -REVEL metrik terdiri dari 5 metrik: local fidelity, local concordance, prescriptivity, conciseness, robustness	akurat dengan model yang diintepretasikan. -Nilai parameter yang optimal adalah ketika nilai metrik local fidelity, local concordance, dan prescriptivenya seimbang.
DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems (Zafar and Khan, 2019)	UCI	-Menggunakan Hierarchical Clustering untuk clustering data training LIME -Menghilangkan langkah sampling dari LIME	-Penggunaan clustering dapat meningkatkan stabilitas prediksi LIME -Jumlah sampel data untuk dataset yang digunakan LIME mempengaruhi stabilitas prediksi LIME
OptiLIME: Optimized LIME Explanations for Diagnostic Computer	NHANES I	-Menggunakan model Survival Gradient Boost untuk klasifikasi.	-OptiLIME dapat memberikan nilai parameter *kernel width* yang optimal agar *adherence* dan *stability* dari hasil LIME seimbang.

Algorithms (Visani, Bagli and Chesani, 2020)		-Mengganti model *ridge penalty* pada LIME dengan model *simple linear regression* -Menggunakan bayesian optimization untuk menentukan parameter kernel width yang optimal	-LIME memiliki hasil lebih baik ketika menggunakan Linear Regression sebagai model untuk penjelasan. -Nilai *kernel width* yang lebih kecil memberikan penjelasan lokal yang lebih realistis

2.2 Dasar Teori

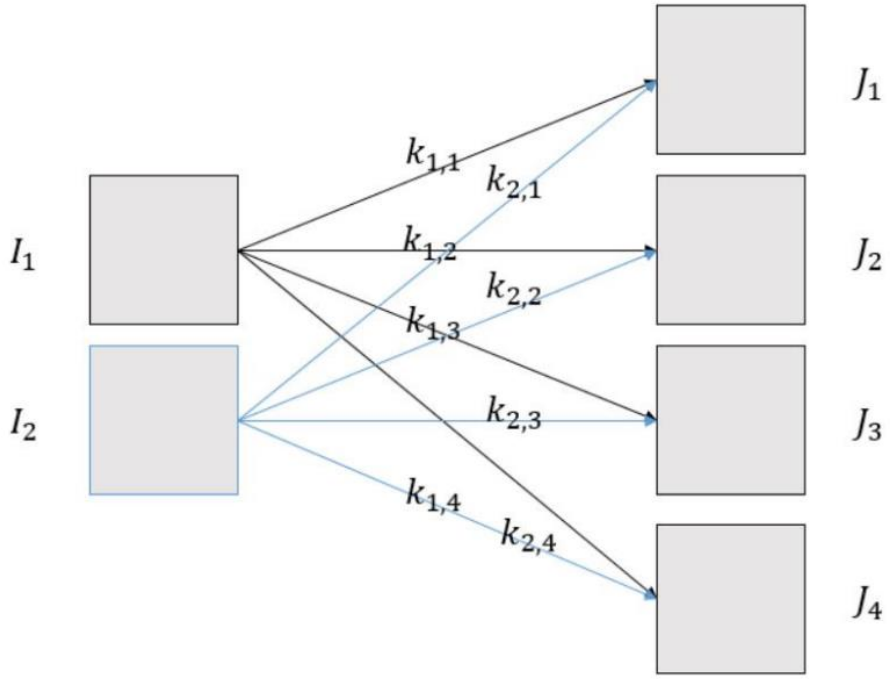
2.2.1 Convolutional Neural Network

Dalam tugas mengklasifikasi gambar, secara umum terdapat dua metode, yaitu dengan *machine learning* konvensional dan dengan *deep learning*. Secara garis besar, kedua metode ini memiliki cara kerja yang berbeda. *Machine learning* konvensional bekerja dengan mengubah *input* gambar menjadi angka secara manual, sehingga terdapat beberapa proses yang perlu dilakukan secara manual, seperti segmentasi, ekstraksi fitur, dan klasifikasi. Sedangkan pada *deep learning*, *input* gambar tidak perlu diubah menjadi angka karena *deep learning* secara otomatis mengubah gambar menjadi angka dan melakukan proses ekstraksi fitur. Oleh karena itu peran manusia pada *deep learning* tidak sebanyak ketika menggunakan *machine learning* konvensional.

Dengan kemajuan kecerdasan buatan, *deep learning* menjadi semakin populer dalam klasifikasi gambar, pengenalan gambar, deteksi objek, dan lain sebagainya. *Convolutional Neural Networks* (CNNs) secara khusus dibangun untuk tugas pemrosesan gambar(O'Shea and Nash, 2015). CNN dapat digunakan untuk melakukan tugas klasifikasi gambar. Performa CNN dapat menyamai atau melampaui kinerja manusia dalam beberapa kasus(Islam *et al.*, 2023). Model CNN sebelumnya telah menunjukkan kinerja yang luar biasa dalam hal klasifikasi,

segmentasi, dan ekstraksi gambar (Bhandare *et al.*, 2016). Kelemahan dari CNN adalah kebutuhan komputasi tinggi dan dibutuhkan dataset yang cukup besar agar didapatkan performa yang baik.

Secara umum, Convolutional Neural Networks (CNN) terdiri dari beberapa lapisan atau layer. Layer pertama, yang merupakan layer konvolusi, menggunakan Rectified Linear Units (ReLU) sebagai fungsi aktivasi dalam proses konvolusi 2D. Biasanya, ReLU ini diterapkan dengan kernel yang telah ditentukan. Setiap proses konvolusi menghasilkan beberapa channel atau feature map (Agarap, 2018). Selanjutnya, dilakukan filtering antara input channel menggunakan kernel tertentu. Hasil filtering dari tiap input channel kemudian dijumlahkan untuk menghasilkan output channel dari proses konvolusi. Ilustrasi dari layer konvolusi dapat ditemukan pada gambar 2.1 (Indraswari, Herulambang and Rokhana, 2022). Jika proses konvolusi melibatkan dua channel input (Ii; i={1,2}) dan menghasilkan empat channel output (Jj; j={1,2,3,4}), maka total kernel yang digunakan untuk konvolusi adalah delapan. Untuk menghasilkan output channel Jj, digunakan persamaan $Jj = \sum Ii * ki,j$; dengan N adalah total channel input dan * menunjukkan operasi konvolusi. Hasil dari konvolusi kemudian diaktivasi menggunakan fungsi ReLU. Oleh karena itu, output dari layer konvolusi, yang awalnya merupakan x, diubah menjadi f(x) menggunakan fungsi ReLU $f(x) = \max(0, x)$ (Agarap, 2018).



Gambar 2.1 Ilustrasi *layer* konvolusi

Selanjutnya, terdapat *layer* kedua yang disebut *layer pooling*. Pada *layer* ini, dilakukan proses kompresi pada *input* gambar untuk menghasilkan fitur-fitur utama dengan menggunakan ukuran *window* tertentu. Proses *pooling* membagi *input* gambar menjadi beberapa bagian sesuai dengan ukuran *window* yang ditentukan. Pada setiap bagian, satu piksel yang diambil untuk mewakili nilai statistik sesuai dengan jenis *pooling* yang digunakan. Terdapat beberapa jenis *pooling*, contohnya *max pooling* dan *average pooling*. *Max pooling*, mengambil nilai tertinggi dalam window, sedangkan *average pooling* mengambil rata-rata nilai dalam window. Setelah proses konvolusi dan *pooling* selesai, hasil kompresi dari masing-masing proyeksi dengan *feature map* ditumpuk. Hasil tumpukan kemudian diproses pada konvolusi lagi untuk menghasilkan *feature map* dari hasil kompresi.

Terakhir adalah layer fully-connected, juga dikenal sebagai dense layer. Pada *layer* ini, *output* dihasilkan dengan mengalikan data numerik dengan bobot yang sesuai. *Input* untuk *layer* ini berupa matriks satu dimensi dan dipetakan ke sejumlah *node*. *Output* dari *layer* ini kemudian dipetakan ke *node* sesuai dengan jumlah kelas dalam dataset. *Node* yang menghasilkan probabilitas tertinggi akan menentukan kelas dari data input. Selanjutnya, dilakukan penghitungan error antara

hasil klasifikasi dan label yang telah ditentukan menggunakan *cost function*. Nilai error ini digunakan untuk memperbarui nilai *filter* konvolusi dalam jaringan..

2.2.2 Image Clasification

Image classification atau klasifikasi gambar adalah salah satu tugas dalam *computer vision* yang menggunakan machine learning dengan tujuan menganalisa dan membedakan objek, hewan, atau orang dalam suatu gambar berdasarkan label atau tag kelas yang sudah diberikan untuk suatu gambar tertentu. Terdapat beberapa jenis klasfikasi gambar seperti binary, multiclass, atau hierarchical.

2.2.3 Explainable AI (XAI)

Explainable AI (XAI) adalah bidang studi yang berupaya menciptakan sistem AI yang mudah diinterpretasikan oleh manusia. Teknik dan metode yang digunakan dalam XAI bertujuan untuk meningkatkan transparansi dan interpretasi model AI dan proses pengambilan keputusan bagi pengguna akhir, termasuk pembuat kebijakan, pakar domain, dan pemangku kepentingan lainnya. Terdapat dua kategori utama metode XAI: metode *model-specific* dan metode *model-agnostic.* Metode *model-specific* berfokus untuk menjelaskan bagaimana sebuah model AI tertentu mencapai keputusannya dengan menganalisis cara kerja dan parameter internalnya (Bhatia *et al.*, 2023). Sedangkan metode *model-agnostic* berfokus untuk menjelaskan proses pengambilan keputusan model AI apapun, terlepas dari struktur internalnya, dengan mengevaluasi input dan output dari model (Bhatia *et al.*, 2023).

2.2.3.1 LIME

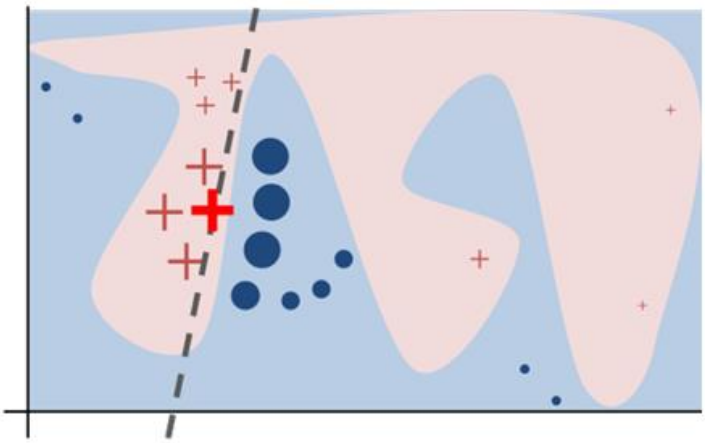
Local Interpretable Model-agnostic Explanations (LIME) merupakan salah satu metode penggunaan XAI. LIME digunakan untuk intepretasi pada *instance-level*. LIME menggunakan model sederhana untuk memperkirakan model yang lebih kompleks. LIME bekerja dengan membuat *perturbed instance* (variasi acak dari *instance* yang diminati) dan mendapatkan hasil prediksi untuk *perturbed instance* ini dari model kompleks yang akan dijelaskan. Kemudian, model sederhana dipasang dengan menggunakan *perturbed instance* sebagai *input* dan prediksi model kompleks untuk *perturbed instance* ini sebagai *output*. Dalam proses ini, *perturbed instance* dibobotkan berdasarkan kedekatannya dengan

instance asli. Fitur-fitur yang signifikan dari model sederhana yang telah disesuaikan akan mendorong prediksi model kompleks untuk *instance* ini (Zhang, Cho and Vasarhelyi, 2022).

Secara garis besar, cara kerja LIME terdiri dari tiga langkah, yaitu (1) mengambil sampel contoh di sekitar prediksi, (2) mendapatkan prediksi untuk contoh-contoh ini menggunakan model asli, dan (3) mempelajari model linier berbobot pada set data ini sebagai penjelasan (Ribeiro, Singh and Guestrin, 2016). Pada langkah pertama dilakukan pembuatan dataset dari contoh yang mirip dengan yang sedang dijelaskan, dengan membuat perubahan (*perturbance*) kecil pada contoh asli sambil tetap menjaga labelnya (yaitu prediksi model kompleks) tetap sama. Sebagai contoh, jika contohnya adalah gambar, LIME dapat mengubah warna atau kecerahan beberapa piksel. Jika contohnya adalah sebuah teks, LIME dapat mengganti beberapa kata dengan sinonim atau menghapus beberapa kata. Tujuannya adalah untuk menghasilkan sekumpulan contoh yang beragam dan representatif yang mendekati contoh asli dalam ruang fitur.

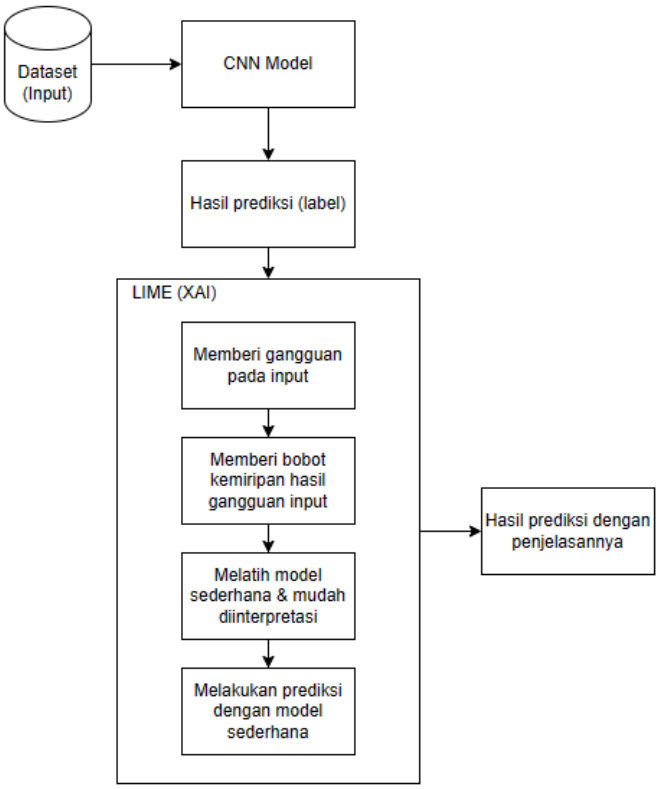
Kemudian untuk langkah kedua, digunakan model *black box* untuk mendapatkan prediksi untuk contoh yang diambil. LIME memperlakukan model sebagai *"black box"* atau kotak hitam, yang berarti tidak perlu mengetahui bagaimana model bekerja secara internal, hanya bagaimana mendapatkan prediksinya. Prediksi tersebut digunakan sebagai variabel target untuk model penjelasan lokal.

Pada langkah ketiga, dilakukan pembelajaran model yang sederhana dan dapat diinterpretasikan yang mendekati perilaku model kompleks di sekitar contoh yang sedang dijelaskan. LIME menggunakan model linier sebagai model penjelasan, yang setiap fiturnya memiliki bobot yang mengindikasikan tingkat kepentingannya untuk prediksi. LIME juga memberikan bobot pada contoh yang diambil berdasarkan kemiripannya dengan contoh asli, dengan menggunakan fungsi kernel. LIME kemudian mengoptimalkan model linier untuk meminimalkan kerugian kuadrat tertimbang antara prediksi model kompleks dan model penjelasan. Model linier yang dihasilkan digunakan untuk menghasilkan penjelasan atas keputusan model kompleks.



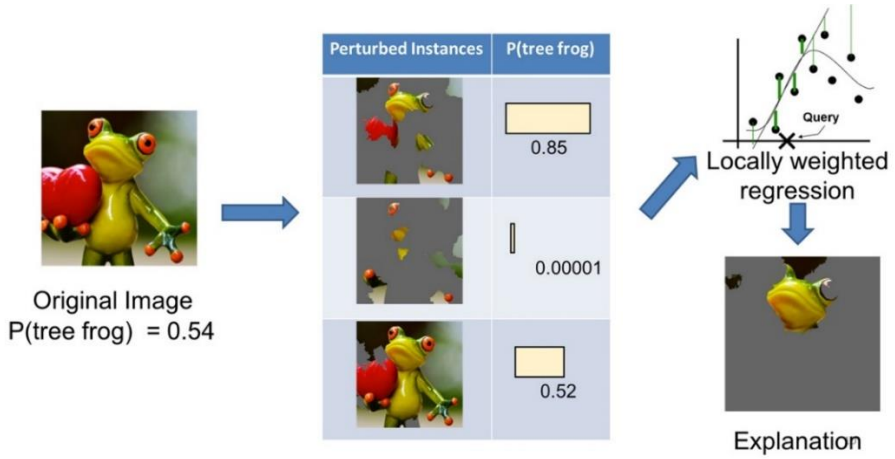
Gambar 2.2 Contoh Sederhana Cara Kerja Fungsi LIME

Gambar 2.2 menunjukkan bagaimana LIME memproses hasil prediksi dari model *black box*. Latar belakang biru dan merah muda mewakili fungsi keputusan yang kompleks dari model *black box*, yang tidak diketahui oleh LIME. Tanda silang merah tebal adalah contoh yang sedang dijelaskan. LIME mengambil sampel contoh di sekitar tanda silang merah, mendapatkan prediksi menggunakan fungsi prediksi model *black box*, dan menimbangnya berdasarkan kedekatannya dengan tanda silang merah (diwakili oleh ukuran lingkaran). Garis putus-putus adalah penjelasan yang telah dipelajari yang sesuai dengan fungsi model *black box* secara lokal, tetapi tidak secara global. Ini adalah model linier yang mendekati fungsi model *black box* di sekitar palang merah. Gambar tersebut mengilustrasikan intuisi di balik LIME: meskipun model asli mungkin terlalu rumit untuk dijelaskan secara global, LIME dapat memberikan penjelasan yang sederhana dan sesuai untuk prediksi individu.



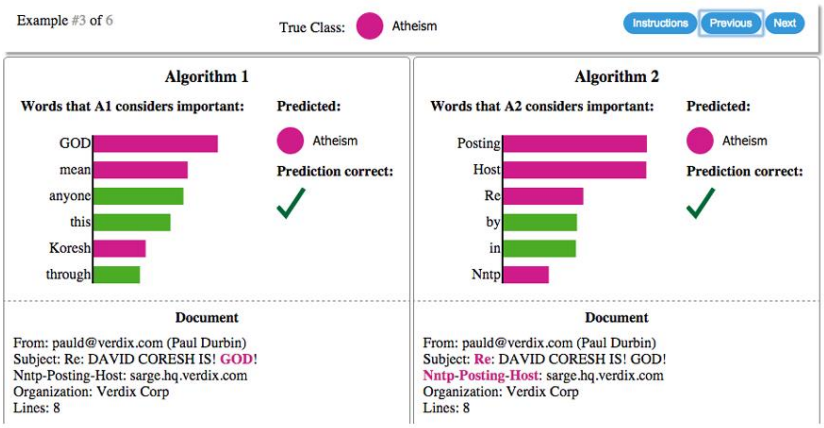
Gambar 2.3 Alur Kerja LIME

Pada gambar 2.3 ditunjukkan alur kerja LIME untuk melengkapi hasil prediksi yang dilakukan oleh model CNN yang bersifat *black box*. Pada gambar ditunjukkan langkah-langkah yang dilakukan pada LIME. Awalnya, dilakukan prediksi pada data input dengan menggunakan CNN Model sehingga menghasilkan prediksi beserta labelnya. Kemudian, setelah didapatkan hasil prediksi dan labelnya, mulai dilakukan proses LIME. Pertama-tama, diberi gangguan kecil pada instansi *(data point)* tertentu pada input dengan mempertahanhkan label hasil prediksi yang telah didapat agar tetap sama. Kemudian diberikan pembobotan pada instansi yang mirip berdasarkan hasil input yang telah diberi gangguan. Semakin mirip suatu instansi maka bobotnya akan semakin tinggi. Berikutnya dilakukan pelatihan model kembali menggunakan model lokal yang lebih sederhana agar dapat lebih mudah diinterpretasikan. Terakhir, dilakukan prediksi dengan menggunakan model lokal yang telah dilatih sebelumnya. Kemudian sebagai *output*, LIME akan memberikan hasil prediksi beserta penjelasannya berdasarkan model lokal yang telah dilatih.



Gambar 2.4 Contoh Penggunaan Lime pada Data Gambar

Pada gambar 2.4, ditunjukkan contoh cara kerja XAI. Pada contoh tersebut, digunakan input berupa gambar katak. Kemudian dibuat dataset baru yang berisi *input* yang diberi gangguan (*perturbed instances*) dengan mematikan beberapa komponen pada *input* awal. Untuk setiap *input* yang diberi gangguan, didapatkan probabilitas katak pada gambar (P(tree frog)). Kemudian dilakukan pelatihan model sederhana (linier) pada dataset baru ini, yang berbobot lokal. Kemudian, disajikan hasil *input* yang diberi gangguan dengan bobot positif tertinggi sebagai penjelasan, dengan mengabu-abukan yang lainnya. Penjelasan tersebut mengungkapkan bahwa pengklasifikasi terutama berfokus pada wajah katak sebagai penjelasan untuk kelas yang diprediksi. Hal ini juga menjelaskan mengapa "meja biliar" memiliki probabilitas yang tidak nol: tangan dan mata katak memiliki kemiripan dengan bola biliar, terutama pada latar belakang hijau. Demikian pula, jantungnya memiliki kemiripan dengan balon merah.



Gambar 2.5 Contoh Penggunaan LIME pada Data Teks

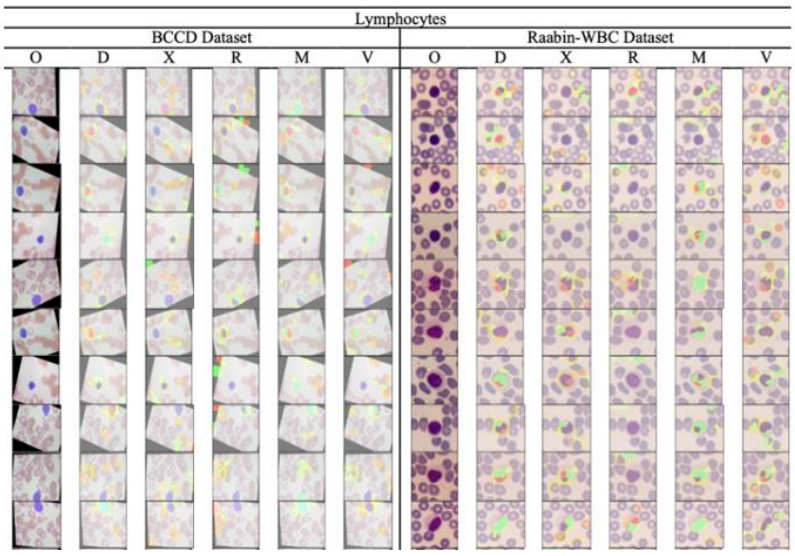
Kemudian, LIME juga dapat digunakan pada tipe data lain. Gambar 2.5 menunjukkan contoh penggunaan LIME pada data teks. Pada contoh tersebut, dilakukan klasifikasi teks dengan menggunakan data 20 Newsgroup dataset. Klasifikasi dilakukan dengan menggunakan *Support Vector Machine* (SVM) dengan RBF kernel. Diambil dua kelas yang sulit dibedakan karena memiliki banyak kesamaan, yaitu *christianity* dan *atheism*. Meskipun pengklasifikasi ini mencapai akurasi 94%, hasil penjelasan menunjukkan bahwa prediksi dibuat dengan alasan yang cukup rancu (kata "*Posting"*, "*Host"*, dan "*Re"* tidak ada hubungannya dengan Kristen atau Atheis). Kata "*Posting"* muncul muncul dalam 22% contoh dalam set pelatihan, 99% di antaranya dalam kelas kelas *Atheism*. Bahkan jika tajuk dihapus, nama-nama yang tepat dari poster-poster yang produktif di newsgroup asli dipilih oleh pengklasifikasi, yang juga tidak akan menggeneralisasi (Ribeiro, Singh and Guestrin, 2016). Setelah mendapatkan wawasan dari penjelasan tersebut, jelas bahwa dataset ini memiliki masalah yang serius (yang tidak terbukti jika hanya mempelajari data mentah atau prediksi), dan bahwa pengklasifikasi ini, atau evaluasi yang dilakukan, tidak dapat dipercaya. Lalu, permasalahan menjadi jelas sehingga dapat ditentukan langkah-langkah yang dapat diambil untuk memperbaiki masalah ini dan melatih pengklasifikasi yang lebih dapat dipercaya.

(a) Husky classified as wolf (b) Explanation

Gambar 2.6 Contoh *Output* LIME untuk Klasifikasi anjing *Husky* dan serigala

Berikutnya pada gambar 3.4 ditunjukkan contoh lain penggunaan LIME. Pada contoh ini, model salah memprediksi anjing Husky sebagai serigala, lalu ditunjukkan penjelasan yang menyebabkan prediksi menjadi wolf. Pada gambar 3.4b dapat dilihat bahwa fitur salju pada latar belakang anjing menjadi alasan model

memprediksi gambar tersebut sebagai serigala. Oleh karena itu, pada kasus ini LIME memberikan wawasan bahwa model yang digunakan kurang dapat dipercaya karena menggunakan fitur latar belakang salju sebagai untuk melakukan prediksi, sehingga menghasilkan prediksi klasifikasi yang salah.



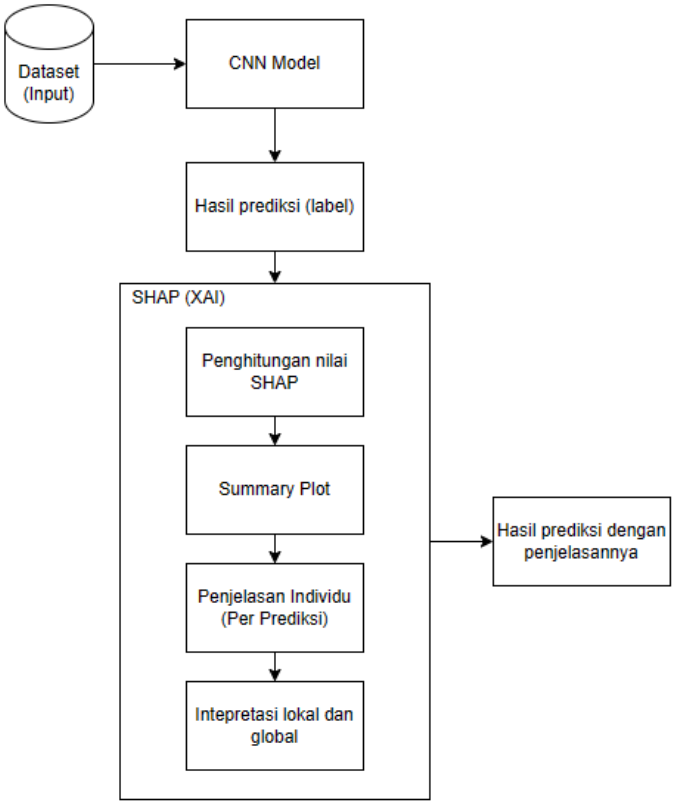
Gambar 2.7 Output LIME untuk Klasifikasi Sel Darah Putih dengan Berbagai Model

Pada gambar 3.5, ditunjukkan contoh lain penggunaan LIME untuk klasifikasi sel darah putih (limfosit). Pada contoh tersebut, dibandingkan gambar asli serta hasil prediksi dari lima model black box. Model yang digunakan adalah DenseNet121 (D), Xception (X), ResNet50 (R), MobileNetV2 (M) dan VGG16 (V). Dengan menganalisis gambar, kita dapat mengidentifikasi piksel atau fitur yang membantu model dalam membuat keputusan yang benar dan salah. LIME dapat digunakan untuk memvalidasi kebenaran prediksi model ketika memprediksi hasil yang benar.

2.2.3.2 SHAP

SHapley Additive exPlanations (SHAP) merupakan salah satu metode XAI yang berdasar pada nilai Shapley dari *game theory*. Metode ini bekerja dengan menggunakan nilai Shapley untuk melakukan pengukuran nilai kontribusi dari seluruh fitur pada sebuah dataset, terhadap hasil prediksi sebuah model (Lundberg and Lee, 2017). Karena hal ini, SHAP memberikan penjelasan terhadap hasil prediksi model secara global, karena mempertimbangkan keseluruhan fitur data. Hal ini yang menjadi pembeda dengan LIME yang memberikan penjelasan hasil

prediksi secara lokal. Secara garis besar, SHAP berfokus pada pentingnya fitur pada keseluruhan dataset (global), jika dibandingkan dengan LIME yang berfokus pada pentingnya fitur pada sebuah prediksi tertentu (lokal).



Gambar 2.8 Diagram Cara Kerja SHAP

Secara garis besar, pada SHAP terdapat beberapa tahapan. Tahap pertama, dilakukan penghitungan nilai Shapley (SHAP) setelah dilakukan prediksi dengan CNN model. Kemudian, dihasilkan summary plot berdasarkan hasil penghitungan nilai Shapley. Plot ini memvisualisasikan pengaruh keseluruhan dari tiap fitur terhadap *output* model pada keseluruhan dataset. Kemudian dihasilkan penjelasan individu. Maksud dari penjelasan individu adalah untuk tiap hasil prediksi, dihitung nilai Shapley untuk memberi penjelasan mengapa model menghasilkan outputnya. Kemudian dibuat intepretasi lokal dan global. Intepretasi lokal adalah penjelasan secara lokal untuk menggambarkan pengaruh tiap fitur terhadap input tertentu. Sedangkan intepretasi global adalah hasil analisa pola pentingnya fitur secara global pada keseluruhan dataset. Secara garis besar, perbedaan metode SHAP

dengan LIME adalah penggunaan nilai Shapley dan hasil intepretasi/penjelasan secara lokal dan global.

2.2.4 Fidelity

Fidelity merupakan salah satu metrik yang dapat digunakan untuk mengukur sebuah metode, seperti model CNN pada kasus ini. *Fidelity* dapat didefinisikan sebagai perbedaaan yang diharapkan antara dua istilah: (a) hasil kali titik *(dot product)* dari gangguan input dengan penjelasan (*output*) dan (b) gangguan *output*, yaitu perbedaan nilai fungsi setelah gangguan yang signifikan pada input (Yeh *et al.*, 2019). *Fidelity* memiliki persamaan sebagai berikut

$$\mathrm{INFD}(\Phi,\mathbf{f},\mathbf{x})=\mathrm{E}_{\mathbf{I}\sim\mu_I}[(\mathbf{I}^T\Phi(\mathbf{f},\mathbf{x})-(\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{x}-\mathbf{I})))^2] \quad (2.2)$$

Pada persamaan tersebut, Φ adalah fungsi penjelasan, f adalah model *black-box*, x adalah *input* (gambar), I adalah gangguan (*pertubation*) signifikan. Pada *fidelity* akan digunakan gangguan signifikan. Menurut penelitian oleh Yeh *et al.*, terdapat dua opsi utama untuk menentukan gangguan signifikan ini. Pertama adalah dengan menggunakan *noisy baseline* yang menggunakan vektor acak Gaussian. Kedua adalah dengan menggunakan *square removal*, yaitu dengan cara menghilangkan subset piksel secara acak. Dari kedua opsi ini, opsi dengan menggunakan *noisy baseline* lebih mudah dari sisi implementasi dan penggunaan. Ketika menggunakan *noisy baseline*, hal yang perlu dilakukan adalah membuat vektor *noise* dengan bentuk yang sama seperti input, dengan menggunakan distribusi Gaussian. Untuk perhitungannya, dapat dibagi menjadi 3 tahap berdasarkan rumus 2.2. Pertama adalah bagian $(\mathbf{I}^T\Phi(\mathbf{f},\mathbf{x})$. Pada tahap ini, dilakukan perhitungan atribusi menggunakan *input* asli dan mengalikannya dengan vektor *noise* yang telah ditransposisi. Hal ini cukup sederhana, namun perlu diperhatikan bahwa jika vektor *noise* memiliki beberapa nilai 0, maka nilai atribusi yang ditempatkan pada posisi yang ditransposisi dari nilai-nilai tersebut akan dihapus (dikalikan dengan 0).

0.00	0.00	0.00
0.00	0.40	0.00
0.00	0.00	0.00

Gambar 2.9 Contoh noise dengan hanya satu nilai tidak bernilai 0

Gambar 2.7 menunjukkan contoh ketika hanya terdapat satu titik yang bernilai tidak 0, sehingga hanya nilai atribusi dari titik tengah (2,2) yang akan digunakan pada perhitungan skor akhir.

Tahap kedua, yaitu $f(x)-f(x-I)$; dilakukan pengurangan nilai atribusi setelah gangguan dari nilai sebelum gangguan dari *output* model *black-box*. Untuk tahap ketiga, yaitu $([...])2$; dilakukan penguadratan terhadap hasil akhir agar tidak terdapat nilai negatif.

2.2.5 Stability

Salah satu aspek penting dalam metode explainble AI adalah stabilitas penjelasan ketika input diberi gangguan. Stabilitas memastikan bahwa masukan yang serupa menghasilkan penjelasan yang serupa, yang merupakan kunci untuk menghasilkan penjelasan yang bermakna di berbagai model. Untuk mengukur stabilitas, didefinisikan perhitungan dengan rumus berikut (Alvarez-Melis and Jaakkola, 2018).

$$L(x_i) = \underset{x_j \in B_\in(x_i)}{argmax} \frac{||f_{expl}(x_i) - f_{expl}(x_j)||_2}{||h(x_i) - h(x_j)||_2} \qquad (2.3)$$

Pada rumus tersebut, $B_\in(x_i)$ menggambarkan ∈ *neighborhood* di sekitar x_i. Kemudian, f_{expl} merupakan model penjelasan untuk input x. Lalu $h(x)$ merupakan fungsi pemetaan, dan || . ||2 merupakan *euclidean norm* (L2 norm). Pada skor *stability*, skor yang lebih kecil atau semakin mendekati nol berarti stabilitas dari penjelasan semakin baik. Hal ini dikarenakan semakin mendekati nol, maka penjelasan dari input asli dengan input yang diberi gangguan tidak berbeda jauh dan berlaku juga sebaliknya.

2.2.6 K-Means Clustering

K-Means clustering adalah salah satu metode dalam melakukan *clustering* data yang berbasis *centroid* atau titik pusat massa. K-Means Clustering adalah algoritma *Unsupervised Machine Learning*, yang mengelompokkan kumpulan data yang tidak berlabel ke dalam cluster yang berbeda. Cara kerja K-Means adalah dengan mempartisi kumpulan data tertentu ke dalam k cluster menggunakan jarak dari setiap titik data ke k pusat massa (*centroid*) atau sarana (*means)* yang berbeda (Ganganath, Cheng and Tse, 2014).

Algoritma *K-means* membagi sekumpulan N sampel X ke dalam K cluster C yang terpisah, masing-masing dijelaskan oleh rata-rata μ_j sampel dalam cluster. *Means* tersebut umumnya disebut cluster *centroids*. Namun, secara umum *centroid* bukanlah titik dari X, meskipun mereka berada di ruang yang sama. *Algoritma K-means* bertujuan untuk memilih pusat massa yang meminimalkan inersia, atau kriteria jumlah kuadrat dalam cluster. Sehingga jika diformulasikan, algoritma K-Means memiliki rumus $\sum_{i=0}^{n} \underset{\mu_j \epsilon C}{min}(\left|\left|x_i - \mu_j\right|\right|^2)$.

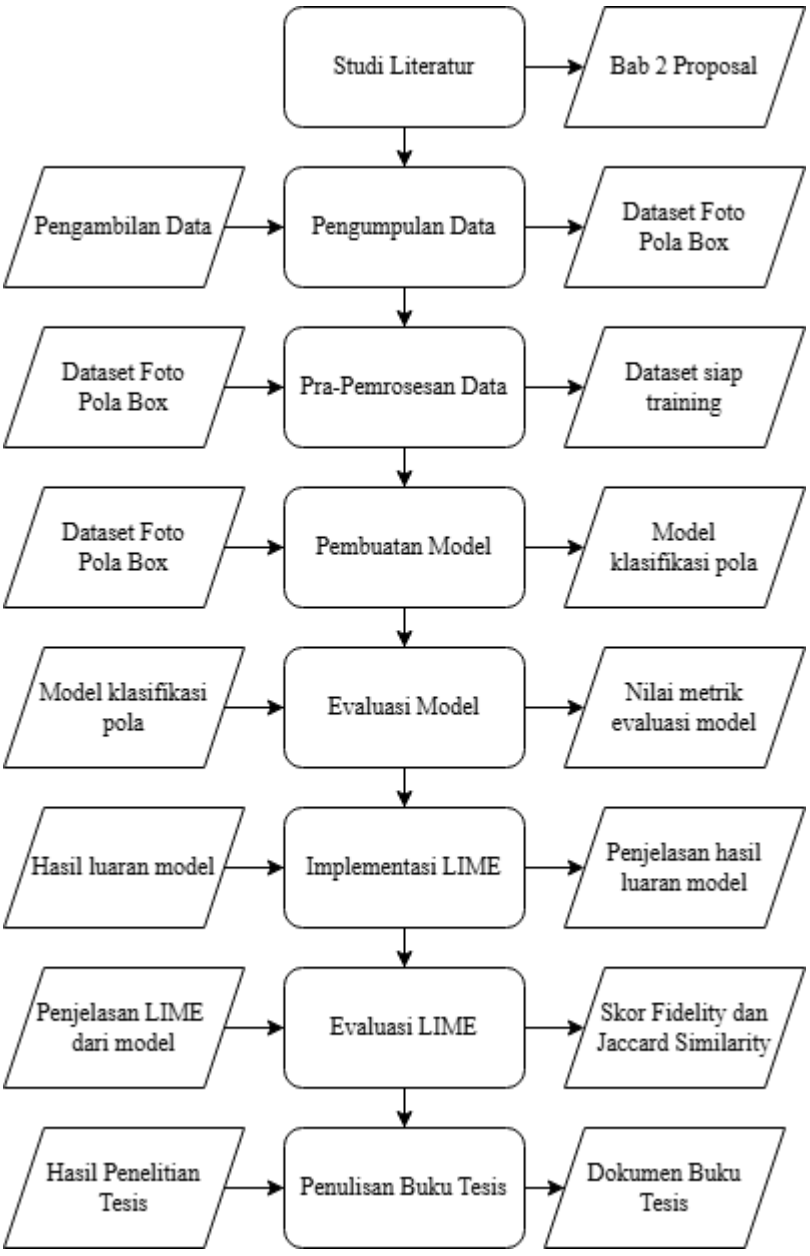
Secara garis besar, algoritma *K-means* memiliki tiga langkah. Langkah pertama adalah memilih *centroid* awal, dengan metode paling dasar adalah memilih K sampel dari kumpulan data X. Setelah inisialisasi, *K-means* melakukan pengulangan antara dua langkah berikutnya. Langkah pertama ini menugaskan setiap sampel ke *centroid* terdekat. Langkah kedua adalah membuat *centroid* baru dengan mengambil nilai rata-rata dari seluruh sampel yang ditugaskan pada setiap *centroid* sebelumnya. Perbedaan antara *centroid* lama dan baru dihitung dan algoritma mengulangi dua langkah terakhir hingga nilai ini kurang dari ambang batas. Hal ini kemudian berulang hingga *centroid* tidak bergerak secara signifikan.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai gambaran metode serta rencana tahapan dari pengerjaan tugas akhir ini. Metodologi berguna agar pengerjaan tugas akhir dapat dilakukan secara sistematis, jelas dan terarah

3.1 Diagram Metodologi Penelitian

Pada sub bab ini, ditunjukkan metodologi penelitian yang akan dilakukan. Diagram metodologi adalah sebagai berikut.



Gambar 3.1 Diagram Metodologi

Diagram metodologi yang ditunjukkan pada gambar 3.1 merupakan alur secara keseluruhan dari pengerjaan tesis ini, mulai dari tahap identifikasi kebutuhan hingga tahap penyusunan buku tugas akhir.

3.2 Uraian Metodologi Penelitian

3.2.1 Studi Literatur

Dilakukan studi literatur terkait teori dan metode yang akan digunakan pada tahap metodologi. Studi literatur dilakukan dengan menggunakan sumber yang berkaitan dengan klasifikasi gambar pola dengan menggunakan arsitektur *deep learning*. Selain itu juga dikumpulkan sumber literatur terkait penggunaan XAI dalam klasifikasi gambar serta contoh implementasi sistem klasifikasi gambar dalam bentuk aplikasi.

3.2.2 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan dataset primer dan sekunder. Dataset primer yang digunakan pada penelitian ini adalah data foto susunan box. Foto diambil secara manual menggunakan kamera dengan posisi pengambilan dari atas susunan box. Objek yang dikumpulkan menjadi data foto adalah foto susunan box untuk dua varian produk, yaitu box untuk varian 60 ml dan box untuk varian 63 ml. Untuk tiap varian, terdapat dua pola yang digunakan dalam satu *box palette*. Kemudian untuk tiap pola terdapat dua kelas, yaitu data untuk susunan benar dan data untuk susunan salah. Secara keseluruhan, terdapat delapan kelas data. Untuk tiap kelas pola susunan benar, terdapat 50 foto. Sedangkan untuk tiap kelas pola susunan salah terdapat 100 foto. Sehingga, total terdapat 600 foto yang dikumpulkan untuk menjadi dataset. Data diambil dengan menggunakan *smartphone* dari sisi atas susunan *box palette.* Dataset memiliki resolusi 1600x900.

Tabel 3.1 Pembagian Kelas Dataset

Kelas	Jumlah Data
60ml_Pola1_Benar	50
60ml_Pola1_Salah	100
60ml_Pola2_Benar	50
60ml_Pola2_Salah	100
63ml_Pola1_Benar	50

63ml_Pola1_Salah	100
63ml_Pola2_Benar	50
63ml_Pola2_Salah	100
Jumlah Data Keseluruhan	**600**

Tabel 3.2 Contoh Data untuk Tiap Kelas

Kelas	**Contoh Data**
60ml_Pola1_Benar	
60ml_Pola1_Salah	
60ml_Pola2_Benar	
60ml_Pola2_Salah	
63ml_Pola1_Benar	

63ml_Pola1_Salah	
63ml_Pola2_Benar	
63ml_Pola2_Salah	

Tabel 3.3 Metadata EXIF

Nama	**Keterangan**
Camera Maker	Xiaomi
Camera Model Name	Redmi Note 5
File Type	JPEG
ISO	640, 800
Flash	Off, did not fire
White Balance	Auto
Megapixel	8
Image Size	4000x2000
Bit Depth	24
Color Representation	sRGB
F-stop	f/1.9
Exposure time	1/20 sec
Focal Length	4mm
EXIF Version	0220

Kemudian pada tabel 3.3 ditampilkan beberapa isi data EXIF dari foto yang digunakan untuk penelitian ini.

3.2.3 Pra-Pemrosesan Data

Sebelum dilanjutkan pada tahap pembuatan model, data yang telah dikumpulkan perlu melalui tahap pra-proses terlebih dulu. Pada tahap ini dilakukan pengelompokkan untuk tiap kelas data. Tiap kelas dipisahkan dengan folder yang berbeda. Secara keseluruhan terdapat delapan kelas, sehingga data dibagi menjadi delapan folder. Kemudian, dilakukan augmentasi data untuk mengakomodasi keterbatasan jumlah dataset. Augmentasi bertujuan untuk mengubah data pelatihan dengan satu atau lebih operasi yang telah ditentukan. Augmentasi yang dilakukan adalah rotasi, flip, zoom, shift terhadap dataset yang ada. Data yang sudah melalui proses augmentasi akan menjadi *input network* untuk tiap iterasi pelatihan model. Pada tiap iterasi, akan digunakan set data yang berbeda dengan jumlah yang sama, agar model nantinya dapat memiliki kemampuan generalisasi yang baik. Semakin baik kemampuan generalisasi dari sebuah model, maka model tersebut dapat menangani data uji yang belum pernah dilatih dengan lebih baik.

3.2.4 Pembuatan Model

Pada tahap ini dilakukan pembuatan model deep learning dengan menggunakan arsitektur base model ResNet50V2, MobileNetV2, VGG16, dan InceptionV3. Model-model ini merupakan model *transfer learning*. *Transfer learning* merupakan salah satu metode *machine learning* yang modelnya dilatih terlebih dulu dengan dataset lain sehingga proses pelatihan pada dataset baru dapat dipersingkat. Model yang sudah dilatih terlebih dulu disebut dengan *pre-trained model*. Proses ini disebut dengan *transfer learning* karena model dapat membawa informasi yang diperoleh dari pelatihan dengan dataset pertama ke pelatihan dengan dataset kedua / dataset studi kasus. Arsitektur - arsitektur model ini digunakan karena berdasarkan penelitian yang sudah ada, memiliki performa yang sangat baik dalam tugas klasifikasi pola dan merupakan beberapa contoh dari model *transfer learning* yang populer digunakan.

Pada proses transfer learning, *layer* paling atas pada pre-trained model dimodifikasi dengan menambahkan *layer* konvolusi atau *pooling*. Layer pertama

ini disebut sebagai head model. Kemudian *layer* berikutnya akan di-*freeze* agar bobot yang ada dan proses *feed forward* tidak berubah. Layer kedua ini disebut sebagai base model. Setelah proses modifikasi ini, kemudian baru dilakukan pelatihan dengan menggunakan dataset kedua dan penyetelan parameter.

Pada penelitian ini akan digunakan beberapa arsitektur base model. Pertama, ResNet50V2 adalah salah satu varian dari arsitektur ResNet (*Residual Network*) dalam pengolahan citra. Fitur utama ResNet50V2 adalah penggunaan fitur *skip connections.* Fitur ini berguna untuk mempelajari perbedaan antara *input* dan *output* pada setiap lapisan *network* model. Fitur *skip connections* berguna untuk mengatasi masalah penurunan kinerja (*vanishing gradient*) yang terjadi ketika gradien menjadi sangat kecil saat proses *backpropagation* ketika pelatihan.

Kemudian, VGG16 adalah salah satu arsitektur CNN yang diusulkan oleh *Visual Geometry Group* dari Universitas Oxford. Arsitektur ini terdiri dari beberapa lapisan konvolusi yang diikuti oleh lapisan pooling, dan diakhiri dengan lapisan fully connected. Penamaan VGG16 didasarkan pada jumlah lapisan yang ada dalam setiap arsitektur. VGG16 terdiri dari 16 lapisan, dan terdapat varian lainnya yaitu VGG19 yang terdiri dari 19 lapisan. Kelebihan utama dari arsitektur VGG adalah kemampuannya dalam mengekstraksi fitur-fitur kompleks dari gambar, yang memungkinkan pengenalan objek yang lebih baik. Namun, kelemahan dari VGG adalah jumlah parameter yang besar, yang membuat arsitektur ini membutuhkan sumber daya komputasi yang cukup tinggi untuk dilatih dan dijalankan.

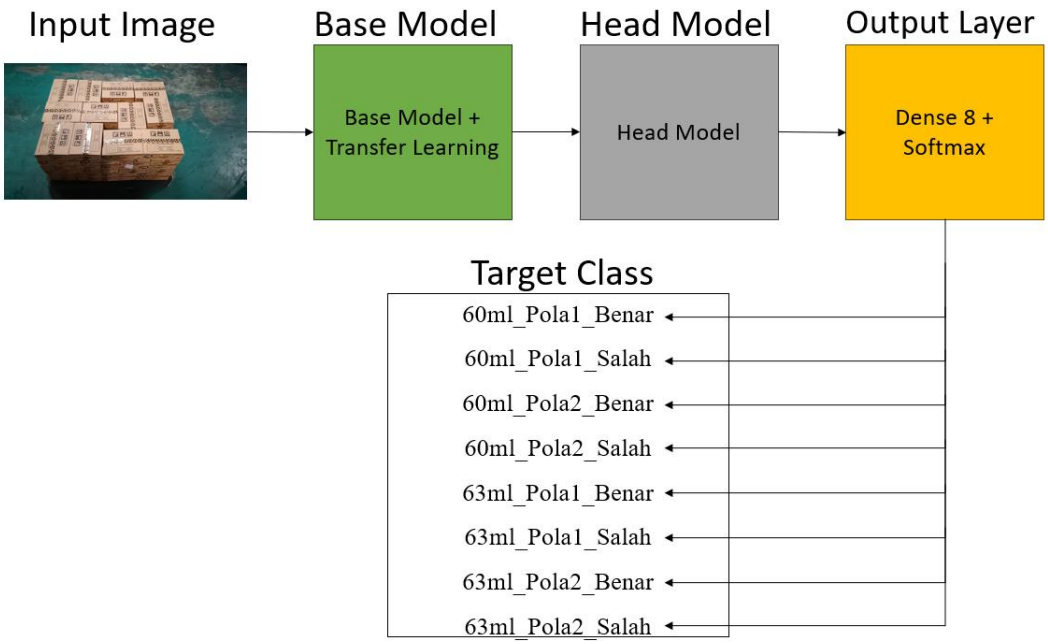
InceptionV3 adalah arsitektur CNN yang dirancang untuk melakukan klasifikasi citra. Arsitektur ini memiliki beberapa fitur yang membuatnya efektif dalam melakukan klasifikasi. Salah satu fitur utamanya adalah penggunaan campuran konvolusi dengan ukuran 1 x 1, 3 x 3, dan 5 x 5. Campuran konvolusi ini digunakan untuk menangkap fitur dengan skala yang berbeda dalam citra. Dengan menggunakan campuran konvolusi ini, InceptionV3 dapat menangkap dan mempelajari fitur-fitur dengan berbagai ukuran dalam citra. Selain itu, InceptionV3 juga menggunakan pengklasifikasi tambahan dalam network untuk meningkatkan kinerja pelatihan dan menghindari overfitting. Pengklasifikasi tambahan ini, yang dikenal sebagai regularizer, membantu membatasi kompleksitas model dan

mendorong model untuk mempelajari fitur-fitur yang lebih umum dan relevan secara keseluruhan serta mengurangi risiko overfitting.

Terakhir, adalah MobileNetV2 yang merupakan arsitektur CNN yang dikembangkan untuk perangkat seluler yang memiliki sumber daya terbatas. Arsitektur ini dirancang dengan pertimbangan sumber daya komputasi, memori, dan kecepatan perangkat. Model ini memiliki keunggulan yaitu penggunaakn depthwise separable convolutions. Metode ini berguna untk memcah operasi konvolusi menjadi dua, yaitu depthwise convolution dan pointwise convolution. Depthwise menghasilkan fitur-fitur yang lebih ringan melalui konvolusi pada tiap saluran input terpisah, sedangkan pointwise menggabungkan fitur-fitur tersebut dengan konvolusi 1x1 pada tiap saluran. Hal ini membuat MobileNetV2 menjadi lebih efisien dalam penggunaan sumber daya.

MobileNetV2 adalah arsitektur CNN yang dikembangkan khusus untuk digunakan pada perangkat seluler dan lingkungan dengan sumber daya terbatas. Arsitektur ini telah dirancang dengan mempertimbangkan keterbatasan daya komputasi, memori, dan kecepatan pada perangkat tersebut. Salah satu keunggulan utama dari MobileNetV2 adalah penggunaan depthwise separable convolutions. Metode ini memecah operasi konvolusi menjadi dua tahap, yaitu depthwise convolution dan pointwise convolution. Depthwise convolution menghasilkan fitur-fitur yang lebih ringan dengan melakukan konvolusi pada setiap saluran input secara terpisah, sedangkan pointwise convolution menggabungkan fitur-fitur tersebut dengan melakukan konvolusi 1x1 pada semua saluran. Pendekatan ini membantu mengurangi jumlah parameter yang dibutuhkan dan kompleksitas komputasi secara signifikan, sehingga MobileNetV2 menjadi lebih efisien dalam penggunaan sumber daya. Meskipun MobileNetV2 memiliki operasi yang lebih ringan, arsitektur ini tetap mampu mempertahankan tingkat akurasi yang tinggi dalam tugas-tugas seperti klasifikasi citra dan 24 pendeteksian objek

Kemudian, arsitektur network juga ditentukan pada tahap ini. Rencana arsitektur head model serta base model yang digunakan pada penelitian ini dapat dilihat pada gambar 3.2



Gambar 3.2 Rencana Arsitektur Model

3.2.5 Uji Coba dan Evaluasi Model CNN

Pada tahap ini dilakukan beberapa uji coba terkait pelatihan model. Pertama dilakukan uji coba arsitektur *base model* ResNet50V2, MobileNetV2, dan VGG16, dan InceptionV3 pada model untuk klasifikasi pola susunan box palette. Kemudian dilakukan juga uji coba metode optimasi yang menetukan nilai minimum pada fungsi pada model klasifikasi. Percobaan akan dilakukan dengan menggunakan 50 *epochs*. *Epoch* menenentukan banyaknya proses pelatihan terhadap data pelatihan dan batch size menentukan jumlah sampel pada tiap batch.

Kemudian pada tahap ini dilakukan evaluasi model dengan menggunakan metrik pengukuran *accuracy, precision, recall* dan *F1 Score.*

- *Accuracy*

  Rasio prediksi benar (positif dan negatif) terhadap keseluruhan data dengan rumus penghitungan sebagai berikut.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \qquad (3.1)$$

- *Precision*

  Rasio prediksi benar positif terhadap keseluruhan hasil yang diprediksi positif dengan rumus penghitungan sebagai berikut

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

- *Recall*

  Rasio prediksi benar positif terhadap terhadap keseluruhan data yang benar positif dengan rumus penghitungan sebagai berikut.

$$Recall = \frac{TP}{TP+FN} \quad (3.3)$$

- *F1-Score*

  Perbandingan rata-rata precision dan recall yang dibobotkan dengan rumus penghitungan sebagai berikut.

$$F1\ Score = 2 \times \frac{Recall \times Precision}{Recall+Precision} \quad (3.4)$$

3.2.6 Implementasi LIME

Pada tahap ini dilakukan implementasi XAI dengan menggunakan LIME *(Local Interpretable Model-agnostic Explanations*). LIME digunakan untuk memberi penjelasan terhadap hasil klafisikasi model. Pada penelitian ini, digunakan LIME karena kedua metode XAI tersebut merupakan salah satu metode yang populer digunakan pada implementasi XAI. LIME merupakan metode XAI yang bersifat model-agnostic, yang artinya LIME tidak bergantung pada model tertentu dan dapat digunakan untuk berbagai model. Implementasi LIME dilakukan melalui beberapa langkah.

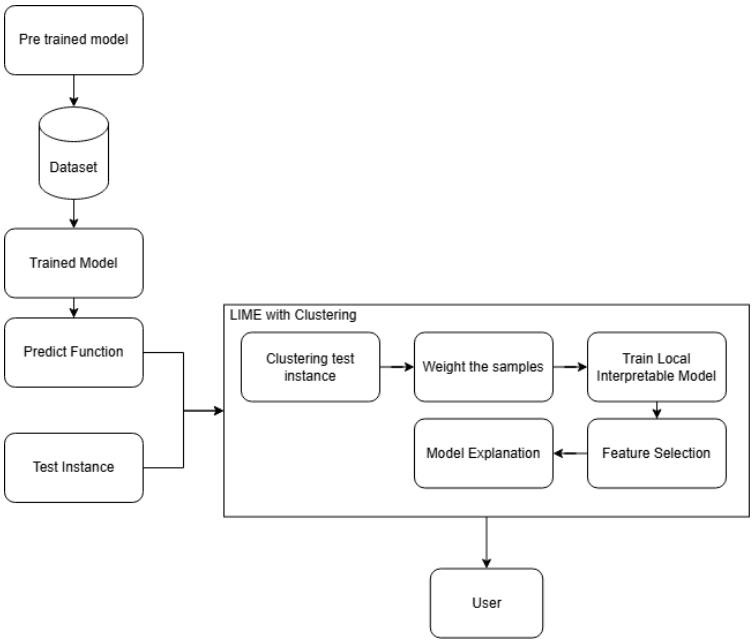
Pertama, dibutuhkan instansi yang menjadi target penjelasan XAI. Dalam kasus ini, instansi yang diinginkan adalah gambar susunan box. Instansi ini akan direpresentasikan dalam bentuk *feature vector*. Kemudian, ditentukan fungsi prediksi dari model yang ingin digunakan. Instansi dan fungsi prediksi ini yang akan menjadi *input* untuk LIME. Kemudian LIME akan melakukan *generate perturbations*, yaitu memberikan gangguan pada instansi yang sudah ditentukan sebelumnya. Hal ini bertujuan untuk memperkirakan perilaku model terhadap instansi tersebut, sehingga dapat diperkirakan juga fitur-fitur apa saja yang penting pada instansi bagi model.

Kemudian dilakukan prediksi dengan model yang sudah ditentukan terhadap instansi yang sudah diganggu. LIME kemudian menghitung kemiripan antara setiap instansi yang terganggu dan contoh asli menggunakan metrik jarak

(misalnya *Euclidean Distance*). LIME memberikan bobot pada instansi yang terganggu berdasarkan kemiripannya dengan instansi asli. Kemudian, LIME melakukan pelatihan dan *fitting* dengan menggunakan *local intepretable model* sederhana seperti model regresi linear, terhadap instansi yang sudah diganggu beserta prediksinya. Tujuannya adalah untuk memperkirakan perilaku model asli yang bersifat *black box.*

Langkah selanjutnya, LIME mengukur pentingnya tiap fitur pada model lokal yang sudah dilatih sebelumnya. Hal ini biasanya dilakukan dengan mengamati koefisien dari *local intepretable model* yang jika semakin besar koefisiennya menunjukkan bahwa fitur tersebut semakin penting.

Setelah mendapatkan skor pentingnya fitur, dibuat penjelasan dari hasil prediksi model asli yang bersifat *black box* terhadap instansi yang dipilih. Penjelasan ini menyoroti fitur mana yang paling banyak berkontribusi pada prediksi dan bagaimana fitur tersebut memengaruhi hasilnya. Penjelasan akan ditampilkan dalam bentuk daftar fitur yang diurutkan berdasarkan tingkat kepentingannya atau visualisasi yang menyoroti fitur-fitur yang berpengaruh.



Gambar 3.3 Usulan Modifikasi LIME dengan Menggunakan Clustering

Kemudian, pada penelitian ini diusulkan modifikasi pada LIME dengan menggunakan K Means Clustering untuk menghasilkan sampel dataset yang digunakan untuk melatih model LIME. K Means clustering digunakan untuk

menggantikan langkah bawaan pada LIME yang menggunakan teknik segmentasi dan memberi gangguan acak pada data input untuk menghasilkan sampel dataset. Dengan menggunakan K Means Clustering diharapkan dapat meningkatkan interpretabilitas model dengan penjelasan yang dihasilkan serta memberi penjelasan LIME lebih stabil dan konsisten. Alur kerja LIME yang dimodifikasi ini dapat dilihat pada gambar 3.3

Dalam alur kerjanya, modifikasi LIME yang dilakukan mirip dengan LIME biasa. Setelah menghasilkan sampel dataset, sampel ini kemudian diberi pembobotan. Kemudian dilakukan pelatihan terhadap *local interpretable model.* dengan menggunakan sampel yang sudah diberi bobot sebelumnya. Model yang dilatih adalah model sederhana seperti regresi linier atau *decision trees*. Digunakan model yang sederhana agar model lebih mudah diintepretasikan sehingga dapat dimengerti oleh manusia. Tujuannya pelatihan model di sini adalah untuk menyesuaikan wilayah lokal di sekitar target tes seakurat mungkin. Kemudian LIME akan melakukan seleksi fitur untuk memilih fitur yang penting. Setelah itu, pengguna diberikan penjelasan model yang dapat diintreptasi. Dalam penelitina ini, penjelasan diberikan dalam bentuk visualisasi fitur penting pada target tes. Penjelasan ini menyoroti fitur mana yang paling berpengaruh dalam prediksi model untuk contoh pengujian tertentu.

3.2.7 Uji coba dan Evaluasi LIME

Kemudian terdapat beberapa parameter yang dapat diuji coba pada penelitian ini. Uji coba yang akan dilakukan adalah sebagai berikut.

- Uji coba LIME biasa pada berbagai dataset
- Uji coba LIME dengan clustering pada berbagai dataset
- Uji coba jumlah sampel pada tiap metode

Setelah LIME menghasilkan *output*, kemudian dilakukan evaluasi terhadap penjelasan yang dihasilkan. Untuk mengukur interpretabilitas model dari penjelasan LIME, dilakukan evaluasi dilakukan dengan menggunakan metrik pengukuran *fidelity* (Velmurugan *et al.*, 2021). Selain itu dilakukan juga evaluasi metrik *stability* untuk mengukur konsistensi penjelasan terhadap perubahan input yang sama (Alvarez-Melis and Jaakkola, 2018).

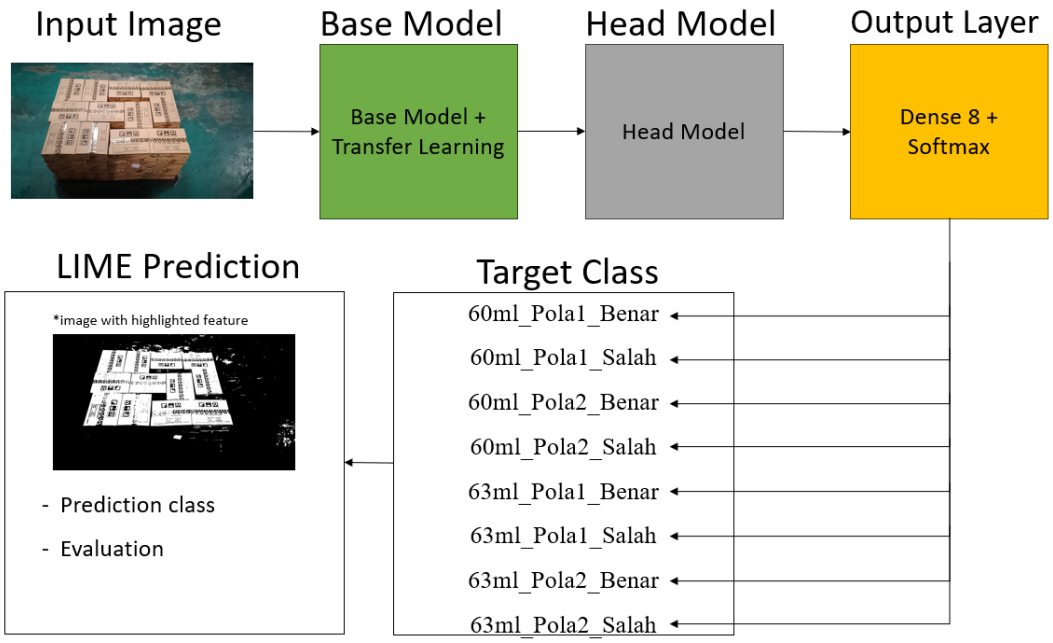
- *Fidelity (R^2 Score)*

  Fidelity mengukur seberapa baik model yang dapat diinterpretasikan (model lokal LIME) mendekati prediksi model *blak box* asli. Secara matematis metrik ini mengkuantifikasi keselarasan antara prediksi yang dibuat oleh model lokal dan prediksi yang dibuat oleh model asli yang lebih kompleks untuk titik data yang terganggu yang digunakan untuk menghasilkan penjelasan.

  Pengukuran *fidelity* tidak hanya mempertimbangkan atribusi, namun juga mempertimbangkan noise dan skor penghitungannya. Pengukuran *fidelity* dilakukan dengan memberikan gangguan yang signifikan terhadap input atau instansi asli. Gangguan signifikan ini dapat diberikan dengan dua metode, yaitu dengan menggunakan *noisy baseline* dan *square removal*.

- *Stability*

  Metrik *stability* digunakan untuk mengevaluasi konsistensi penjelasan yang dihasilkan oleh model *machine learning*. Secara khusus, metrik ini mengukur seberapa besar perubahan penjelasan ketika data *input* diberi sedikit gangguan. Jika perubahan kecil pada data *input* menyebabkan perubahan besar pada penjelasan, maka penjelasan tersebut dianggap kurang stabil (atau kurang dapat dipercaya).



Gambar 3.4 Penambahan LIME pada Tahapan Klasifikasi

Pada penelitian ini, implementasi LIME dilakukan untuk melengkapi tahapan uji coba model. Pada gambar 3.4, ditunjukkan peran LIME pada tahapan metodologi penelitian ini.

3.2.8 Penulisan Buku Tesis

Pada tahap ini dilakukan tahap akhir dari seluruh penelitian ini, yaitu penulisan buku sebagai dokumentasi dari seluruh proses yang dilakukan dalam penelitian ini mulai dari studi literatur hingga penyusunan buku tesis. Penyusunan buku tesis ini diperlukan untuk membantu penulis mengambil kesimpulan dari penelitian serta agar seluruh pembelajaran dan hasil dari penelitian ini dapat menjadi referensi untuk penelitian selanjutnya.

3.3 Rencana Jadwal Pelaksanaan Penelitian

Rencana jadwal untuk penelitian tesis yang akan dilakukan ditunjukkan pada tabel berikut.

Tabel 3.4 Tabel Jadwal Pengerjaan Penelitian Tesis

No	Kegiatan	Minggu ke-															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Studi Literatur																
2	Pengumpulan Data																

3	Pra-pemrosesan Data																
4	Pembuatan Model																
5	Evaluasi Model																
6	Implementasi XAI																
7	Penulisan Buku Tesis																

BAB 4 HASIL DAN PEMBAHASAN

Pada bab ini, akan dijelaskan alur dan setiap proses untuk mendapatkan luaran yang diinginkan pada penelitian. Hal-hal yang dipaparkan pada bab ini adalah proses pengumpulan data, pra-pemrosesan data, eksperimen pada model, eksperimen pada XAI sampai dengan evaluasi hasil.

4.1 Pengumpulan Data

Selain menggunakan data susunan box palette sebagai data primer, dilakukan juga pengumpulan dataset sekunder sebagai dataset untuk uji metode. Dataset sekunder yang digunakan adalah dataset *flower images* dan *intel images*. Dataset *flower images* dipilih karena dataset tersebut merupakan dataset multikelas yang dapat dikatakan mudah diklasifikasi karena karakteristik tiap kelas bunga dapat dilihat jelas dari bentuk dan warnanya. Dataset ini juga memiliki ukuran yang cukup mirip dengan dataset primer dengan ukuran dataset berisi 210 gambar. Kemudian, dataset *intel images* dipilih karena dataset ini juga merupakan dataset multikelas. Namun, dataset ini termasuk sulit diklasifikasi karena karakteristik beberapa kelas yang mirip. Selain itu, dataset ini juga berfungsi sebagai contoh dataset dengan ukuran yang besar, karena berisi 25000 gambar.

4.1.1 Dataset *Box Palette*

Dataset primer yang digunakan adalah dataset *box palette* yang dikumpulkan sendiri. Dataset yang dikumpulkan adalah foto susunan box untuk dua varian produk, yaitu box untuk varian 60 ml dan box untuk varian 63 ml. Untuk tiap varian, terdapat dua pola yang digunakan dalam satu *box palette*. Kemudian untuk tiap pola terdapat dua kelas, yaitu data untuk susunan benar dan data untuk susunan salah. Secara keseluruhan, terdapat delapan kelas data. Untuk tiap kelas pola susunan benar, terdapat 50 foto. Sedangkan untuk tiap kelas pola susunan salah terdapat 100 foto. Sehingga, total terdapat 600 foto yang dikumpulkan untuk menjadi dataset. Tiap foto pada dataset memiliki dimensi 1600x900. Data diambil dengan menggunakan *smartphone* dari sisi atas susunan *box palette.*

Tabel 4.1 Label dan Nama Kelas Dataset Palette

Label	Kelas
0	60ml_Pola1_Benar
1	60ml_Pola1_Salah
2	60ml_Pola2_Benar
3	60ml_Pola2_Salah
4	63ml_Pola1_Benar
5	63ml_Pola1_Salah
6	63ml_Pola2_Benar
7	63ml_Pola2_Salah

4.1.2 Dataset *Flower Images*

Dataset *Flower Color Images* adalah dataset yang dikumpukan oleh Olga Belitskaya dan dapat diakses pada situs kaggle.com. Dataset terdiri dari 210 gambar dengan ukuran dimensi 128x128. Kemudian, dataset terdiri dari 10 kelas bunga, yang dapat dilihat pada tabel 4.1 berikut. Contoh gambar pada dataset dapat dilihat pada gambar 3.1.

Tabel 4.2 Label dan Nama Kelas Dataset Bunga

Label	Nama
0	phlox
1	rose
2	calendula;
3	iris;
4	leucanthemum maximum (Shasta daisy)
5	campanula (bellflower)
6	viola

7	rudbeckia laciniata (Goldquelle)
8	peony
9	aquilegia

Gambar 4.1 Dataset Flower Color Images

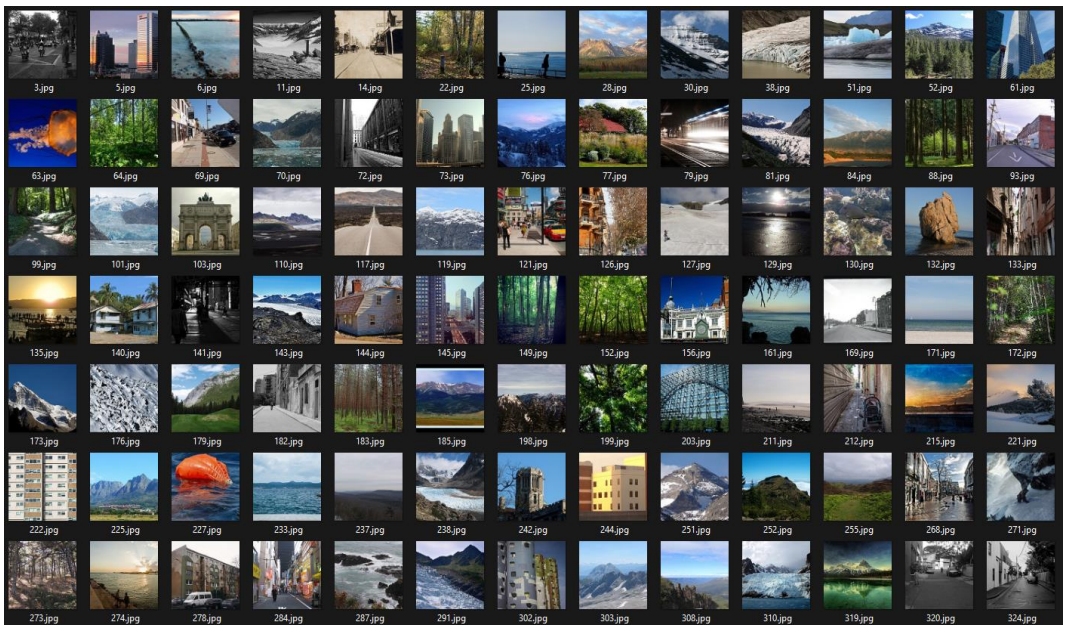
4.1.3 Dataset *Intel Images*

Berikutnya akan digunakan juga dataset *Intel Image Classification* yang dibuat oleh Intel. Dataset terdiri dari 25000 gambar dengan ukuran dimensi 150x150. Dataset ini berisi gambar dari berbagai macam pemandangan yang dibagi ke dalam enam kelas. Enam kelas tersebut dapat dilihat pada tabel 4.2 dan contoh gambar dataset dapat dilihat pada gambar 4.2 berikut.

Tabel 4.3 Pembagian Kelas Dataset Intel Images

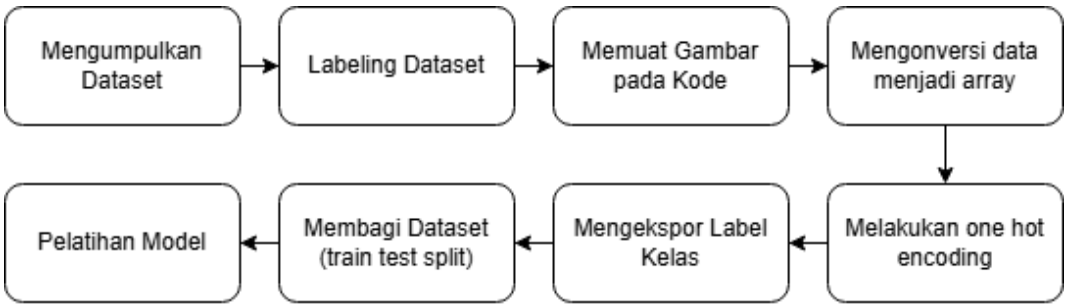
Label	Nama
0	'buildings'
1	'forest'
2	'glacier'
3	'mountain'

4	'sea'
5	'street'



Gambar 4.2 Gambar pada Dataset Intel Images

4.2 Pra-Pemrosesan Data



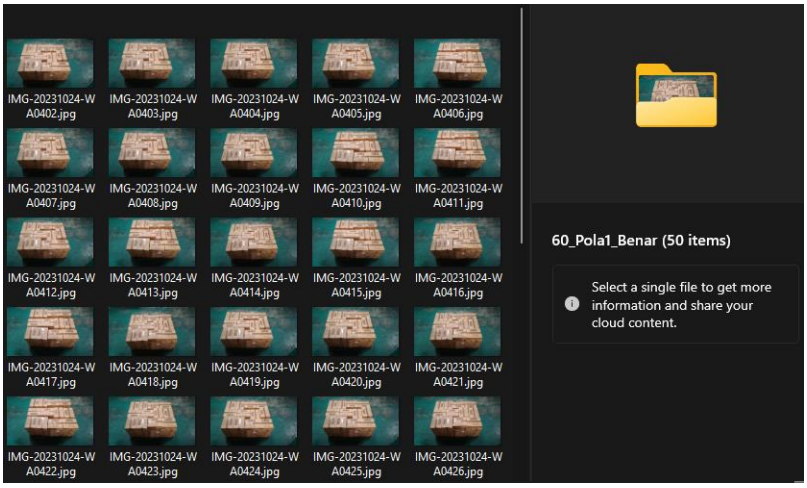
Gambar 4.3 Alur Pra-Pemrosesan Data

Pada tahap ini, dilakukan pra-pemrosesan pada data primer foto susunan box yang telah dikumpulkan. Alur pra-pemrosesan data dapat dilihat pada gambar 4.3

4.2.1 Labeling Dataset

Data foto yang telah dikumpulkan dikelompokkan ke dalam delapan kelas, yaitu 60ml_Pola1_Benar, 60ml_Pola1_Salah, 60ml_Pola2_Benar, 60ml_Pola2_Salah, 63ml_Pola1_Benar, 63ml_Pola1_Salah, 63ml_Pola2_Benar, 63ml_Pola2_Salah. Pengelompokkan dilakukan dengan cara meletakkan foto ke masing-masing direktori sesuai dengan kelasnya.

Gambar 4.4 Pembagian Gambar ke dalam Direktori sesuai Kelas



Gambar 4.5 Gambar pada Kelas 60_Pola1_Benar

Pada gambar 4.4 dan 4.5, ditunjukkan kondisi dataset setelah dilakukan pembagian pada direktori sesuai kelas.

4.2.2 Kode Pra-Pemrosesan Data

Kemudian, sebelum data *box palette* bisa digunakan untuk melatih model *deep dearning*, dilakukan beberapa tahapan pra-pemrosesan dengan menggunakan *Python*. Langkah yang dilakukan dimulai dari data mentah sampai dengan data siap latih. Langkah-langkah yang dilakukan akan dijelaskan pada sub bab berikut.

4.2.2.1 Memuat Gambar

```
dataset = r"D:\dataset tesis\combined";

# Ambil daftar gambar di direktori kumpulan data yang sudah dibuat
lalu Inisialisasi
# daftar data (mis., Gambar) dan gambar kelas
print("[INFO] loading images...")
```

```
imagePaths = list(paths.list_images(dataset))
data = []
labels = []


# Masukkan data diatas dalam Path Gambar
for imagePath in imagePaths:
    # ekstrak label kelas dari nama file
    label = imagePath.split(os.path.sep)[-2]

    # Masukkan / Impor gambar dengan piksel (320x180) dan
preprocess sesuai model
    image = load_img(imagePath, target_size=(320, 180, 3))
    image = img_to_array(image)
    image = preprocess_input(image)

    # Perbarui daftar data dan label masing-masing
    data.append(image)
    labels.append(label)
```

Kode 4.1 Kode untuk Memuat Gambar

Pertama, ditentukan direktori dataset *box palette* melalui variabel "dataset". Direktori ini adalah direktori yang berisi gambar yang sudah dibagi menjadi delapan direktori sesuai kelasnya. Kemudian "imagePaths" akan membuka direktori "dataset"yang telah ditentukan sebelumnya. Lalu dibuat *list* kosong bernama "data" dan "labels" untuk menyimpan data gambar dan label yang akan dimuat.

Kemudian, dibuat loop untuk memuat seluruh data dari direktori. Dilakukan ekstrak label dari nama-nama direktori yang ada dan disimpan dalam variabel "label". Kemudian gambar diubah ukurannya menjadi 320x180, lalu diubah ke dalam bentuk array dan disimpan dalam variabel "image". Kemudian dilakukan tahap preprocess_input pada "image". Langkah ini dilakukan karena merupakan prasyarat yang dibutuhkan oleh model deep learning yang akan digunakan.

Kemudian, "image" dan "label" masing-masing disimpan ke dalam list kosong "data" dan "label" yang sudah dibuat sebelumnya.

4.2.2.2 Mengonversi list gambar dan label menjadi array

```
# Konversikan data dan label ke array NumPy
data = np.array(data, dtype="float32")
labels = np.array(labels)
```

Kode 4.2 Kode Konversi Gambar ke Array

Pada kode ini, dilakukan konversi dari “data” dan “labels” pada data *box palette* dari bentuk list menjadi bentuk NumPy array. Konversi ini dilakukan karena array memiliki efisiensi yang lebih baik dibanding list, sehingga dapat membantu proses pelatihan data terhadap model.

4.2.2.3 Label dan One-Hot encoding

```
# lakukan Pengkodean satu-hot pada label
lb = LabelEncoder()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
print(lb.classes_)

# Assuming 'lb.classes_' contains the class labels
class_labels = lb.classes_
```

Kode 4.3 Kode Label Encoding

Pada tahap ini, dijalankan fungsi LabelEncoder dari modul sklearn.preprocessing pada dataset *box palette* yang sudah dimuat. LabelEncoder digunakan untuk mengubah label kategorikal (yang biasanya berupa string) menjadi label numerik. Kemudian “class labels” akan menyimpan label dalam bentuk nama sebelum diubah menjadi numerik.

4.2.2.4 Mengekspor Label Kelas

```
# Convert the class labels to a list
class_labels_list = list(class_labels)

# Define the path where you want to save the text file
txt_path = 'class_labels_combined.txt'

# Export the class labels to a text file
with open(txt_path, 'w') as txt_file:
    for label in class_labels_list:
        txt_file.write(f"{label}\n")

print(f"Class labels exported to {txt_path}")
```

Kode 4.4 Mengekspor Label ke File Teks

Pada langkah ini, “class_labels” dari dataset *box* palette diubah menjadi list. Kemudian list akan disimpan dalam file teks agar label yang sudah ada dapat terdokumentasi dan dapat diakses dengan lebih mudah.

4.2.2.5 Membagi Dataset

```
(trainX, testX, trainY, testY) = train_test_split(
data, labels, test_size=0.20, stratify=labels, random_state=42)

print(len(trainX))
print(len(testX))
```

Kode 4.5 Membagi data dengan *train test split*

Pada langkah ini, dilakukan pembagian dataset *box palette* untuk training dan test dengan menggunakan metode *train test split*. Pembagian dilakukan dengan rasio data latih 80% dan data tes 20%. Parameter "stratify=labels' digunakan agar pembagian data yang dilakukan lebih merata untuk tiap kelas data. Parameter "random_state=42" digunakan agar pembagian data dilakukan secara acak, namun tetap mengikuti *random state* yang ditentukan agar pembagian tetap konsisten ketika diulang.

4.2.2.6 Augmentasi Data

```
# image generator (augmentation)
aug = ImageDataGenerator(
    rotation_range=30,
    zoom_range=0.25,
    width_shift_range=0.25,
    height_shift_range=0.25,
    shear_range=0.15,
    horizontal_flip=True,
    # brightness_range=(0.5,2.0),
    fill_mode="nearest")
```

Kode 4.6 Melakukan augmentasi pada Data

Pada tahap ini dilakukan augmentasi data untuk meningkatkan keragaman dataset *box palette* yang dilatih dengan menerapkan transformasi acak pada gambar. Augmentasi adalah transformasi terhadap dataset, seperti melakukan rotasi, zoom, dan flip pada dataset. Augmentasi dilakukan karena dapat membantu meningkatkan performa dan generalisasi model pembelajaran mesin.

4.3 Pembuatan dan Evaluasi Model Deep Learning

Kemudian, pada tahap ini dilakukan pembuatan model Deep Learning. Model deep learning dibuat dengan menggunakan *transfer learning*. Base model yang digunakan pada penelitian ini adalah ResNet50V2, VGG16, InceptionV3, dan MobileNetV2

4.3.1 Set Parameter

```
# learning rate
INIT_LR = 1e-4
EPOCHS = 50
BS = 32
NETWORK = "ResNet50V2_Dense512_KFold5"
```

Kode 4.7 Kode parameter

Pertama, dilakukan penentuan parameter yang akan digunakan untuk melakukan pelatihan model. Parameter yang ditentukan adalah *learning rate* (INIT_LR), *epochs*, *batch size*, dan nama *network* atau model.

4.3.2 Loading Pre-Trained Model (Base Model)

```
baseModel = ResNet50V2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(320, 180, 3)))
baseModel.trainable = False #freezing base model layers
```

Kode 4.8 Pembuatan Base Model

Pembuatan model dilakukan dengan menggunakan metode transer learning. Oleh karena itu, pertama dilakukan *loading pre-trained* model atau disebut juga dengan *base* model. Base model yang digunakan pada penelitian ini adalah ResNet50V2, VGG16, InceptionV3, dan MobileNetV2. Kemudian ditentukan parameter terkait *base* model yang digunakan. Pada parameter "weights" digunakan imagenet untuk menggunakan pembobotan berdasarkan pelatihan base model dengan dataset imagenet. Kemudian pada base model ini, lapisan paling atas (*fully connected* layer) pada base model tidak digunakan sehinga parameter pada include_top = False. Lalu ditentukan bentuk input yang ingin digunakan. Ukuran input ditentukan berdasarkan dataset. Pada dataset *flower images*, gambar dikonversi ke ukuran 224x224. Pada dataset *intel images* gambar dikonversi ke ukuran 150x150. Pada dataset *box palette* gambar akan dikonversi ke ukuran 320x180. Gambar yang digunakan di semua dataset adalah gambar berwarna, sehingga parameter shape yang digunakan adalah (320, 180, 3) yang berarti 320x180 piksel dengan 3 kanal warna (RGB). Terakhir, lapisan pada base model dibekukan agar semua layer pada base dibuat menjadi *non-trainable* sehingga model yang sudah ada tidak terbarui ketika awal proses pelatihan.

4.3.3 Menyusun Head Model

```
# Head Model
```

```
headModel = baseModel.output
print(headModel.shape)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(512, activation="relu")(headModel)
headModel = Dense(8, activation="softmax")(headModel)

# tempatkan kepala model di atas model dasar ini akan menjadi
model sebenarnya yang akan dilatih
model = Model(inputs=baseModel.input, outputs=headModel)
# model.summary()
# Model Compile ( Mengukur accuracy )
print("[INFO] compiling model...")
opt = Adam(learning_rate=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt,
              metrics=["accuracy"])
```

Kode 4.9 Pembuatan Head Model

Setelah base model, kemudian dilakukan pembuatan head model. Pada penelitian ini digunakan tiga lapisan, yaitu satu lapis Flatten dan dua lapis Dense. Lapisan Flatten digunakan untuk mengonversi output dari base model yang berbentuk multi dimensi menjadi bentuk satu dimensi. Kemudian, lapisan Dense merupakan lapisan *fully connected* yang berarti tiap neuron pada lapisan ini terhubung dengan lapisan sebelumnya. Untuk aktivasi digunakan "relu", yang berarti lapisan dense ini menggunakan metode aktivasi ReLU (Rectified Linear Unit). ReLU membantu model agar model dapat mempelajari pola yang kompleks.

Kemudian, setelah seluruh lapisan model disusun, digunakan optimizer bertipe Adam (*Adaptive Moment Estimation*) dengan parameter *learning rate* yang telah ditentukan sebelumnya. Setelah itu, dilakukan *compiling* model dengan fungsi *loss* untuk mengukur perbedaan antra label sebenarnya dengan probabilitas prediksi. Parameter loss yang digunakan adalah categorical_crossentropy karena cocok digunakan untuk klasifikasi *multi-class*. Kemudian ditentukan juga untuk menggunakan metrik *accuracy* untuk evaluasi performa model.

4.3.4 Pelatihan Model terhadap Dataset

```
# Model Fit
lala = aug.flow(trainX, trainY)
print(len(lala))
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
```

```
    # steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    # validation_steps=len(testX) // BS,
    epochs=EPOCHS)
```

Kode 4.10 Pelatihan Model pada Data

Setelah model berhasil *dicompile,* kemudian dilakukan pelatihan model terhadap dataset. Pertama dibuat iterator (lala) yang menghasilkan kumpulan data augmentasi dari set pelatihan. Setiap iterasi lala dilakukan, maka akan menghasilkan kumpulan gambar baru yang diaugmentasi dengan labelnya yang sesuai. Kemudian pelatihan dilakukan dengan parameter aug.flow(trainX, trainY, batch_size=BS) untuk membuat iterator yang menghasilkan kumpulan data pelatihan augmentasi dengan *batch size* yang ditentukan. Kemudian dataset bagian testX dan testY masing-masing digunakan sebagai data validasi dan labelnya.

4.3.5 Melakukan Prediksi Model

```
# Buat prediksi pada set pengujian
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# Untuk setiap gambar dalam set pengujian kita perlu menemukan
Indeks File label
# dengan probabilitas prediksi terbesar yang sesuai
predIdxs = np.argmax(predIdxs, axis=1)
```

Kode 4.11 Melakukan Prediksi Model

Setelah semua epoch pelatihan model selesai, model dievaluasi dengan melakukan prediksi model terhadap dataset bagian testX dengan menggunakan parameter batch_size. Hasil prediksi disimpan dalam variabel predIdxs. Kemudian hasil prediksi dikonversi ke indeks kelas dengan memilih nilai indeks maksimal pada axis=1, yang berarti pada bagian kolom. Fungsi ini akan menghasilkan array satu dimensi yang tiap elemennya merupakan indeks kelas hasil prediksi untuk tiap sampel tes.

4.3.6 Evaluasi dan Penyimpanan Model

```
# Tampilkan Laporan Klasifikasi yang diformat dengan baik
print(classification_report(testY.argmax(axis=1), predIdxs,
                            target_names=lb.classes_))

# Lakukan Penyimpanan model ke disk
print("[INFO] saving number of boxes detector model...")
```

```
# Save the model in the native Keras format
model.save('ResNet50V2_Kfold5_20.keras')
```

Kode 4.12 Evaluasi dan Penyimpanan Model

Kemudian, hasil evaluasi akan ditampilkan dengan menggunakan fungsi classification_report. Fungsi ini akan menampilkan precision, recall, F1-score, dan support untuk tiap kelas pada dataset.

4.3.7 Hasil Evaluasi Model

Kemudian, dilakukan uji coba model terhadap tiap dataset. Pengujian yang dilakukan adalah menguji jenis *base model* ResNet50V2, VGG16, InceptionV3, dan MobileNetV2 pada tiap dataset. Digunakan jumlah epoch sebesar 50 untuk tiap jenis *base model* yang diujikan.

4.3.7.1 Evaluasi model dataset *flower*

Pada bagian ini dilakukan pelatihan model pada dataset *flower images* sesuai percobaan jenis *base model* dan jumlah *epoch* 50.

Tabel 4.4 Hasil Evaluasi Model (Dataset Flower)

Base Model	**Performa Model (%)**				***Running Time* (detik)**
	Accuracy	***Precision***	***Recall***	***F1-Score***	
ResNet50V2	79	83	79	77	90
VGG16	76	83	76	77	86
InceptionV3	74	80	74	74	88
MobileNetV2	**88**	**90**	**88**	**88**	**84**

Dari evaluasi yang dilakukan, maka model terbaik untuk dataset flower adalah model MobileNetV2 dengan 50 epoch. Model ini memiliki F1-score terbaik dengan nilai skor 88%.

Gambar 4.6 Grafik Akurasi dan Loss pada Training/Validation (flower)

Kemudian, dari grafik akurasi pada gambar 4.6, dapat dilihat grafik akurasi dan loss ketika melakukan pelatihan model.

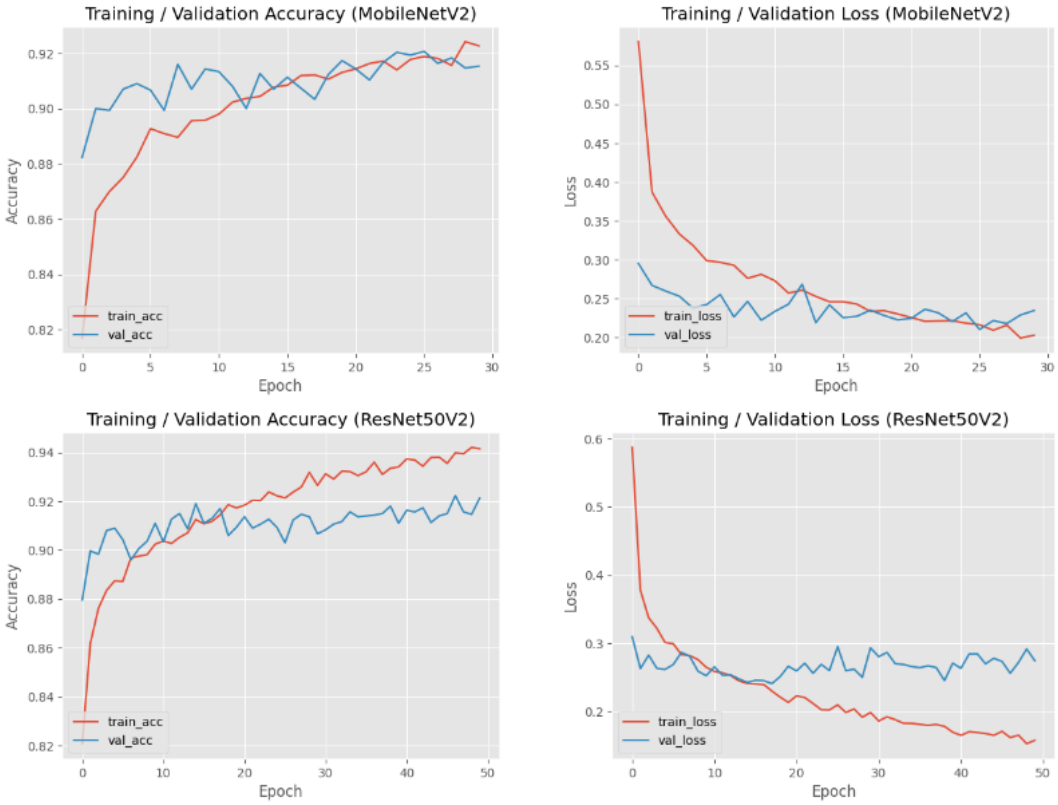
4.3.7.2 Evaluasi model dataset *intel images*

Pada bagian ini dilakukan pelatihan model pada dataset *intel images* sesuai percobaan jenis *base model* dan jumlah *epoch* yang ditentukan.

Tabel 4.5 Hasil Evaluasi Model (Dataset Intel)

Base Model	**Performa Model (%)**				***Running Time* (detik)**
	Accuracy	***Precision***	***Recall***	***F1-Score***	
ResNet50V2	92	92	92	92	3157
VGG16	90	90	90	90	2994
InceptionV3	90	90	90	90	2925
MobileNetV2	**92**	**92**	**92**	**92**	**1792**

Dari evaluasi yang dilakukan, maka model terbaik untuk dataset intel images adalah model MobileNetV2 dengan 50 epoch. Model ini memiliki F1-score terbaik dengan nilai skor 92%. Model ini memiliki F1-score yang sama dengan model ResNet50V2 dengan 50 epoch. Namun model MobileNetV2 lebih dipilih karena running time yang lebih singkat serta grafik train/validation yang lebih baik.



Gambar 4.7 Grafik Perbandingan Akurasi dan Loss pada Training/Validation (intel)

Kemudian, dari grafik akurasi pada gambar 4.7, dapat dilihat grafik akurasi dan loss ketika melakukan pelatihan model untuk dataset flower. Grafik pada model MobileNetV2 lebih diutamakan karena menunjukkan tren validation loss yang menurun jika dibandingkan dengan validation loss pada ResNet50V2 yang lebih stagnan.

4.3.7.3 Evaluasi model dataset box pallete

Pada bagian ini dilakukan pelatihan model pada dataset *box palette* sesuai percobaan jenis *base model* dan jumlah *epoch* 50.

Tabel 4.6 Hasil Evaluasi Model (Dataset Palet)

Base Model	**Performa Model (%)**					***Running Time* (detik)**
	Epochs	***Accuracy***	***Precision***	***Recall***	***F1-Score***	
ResNet50V2	50	98	98	98	98	279
VGG16	50	100	100	100	100	315
InceptionV3	50	96	96	96	96	308
MobileNetV2	**50**	**100**	**100**	**100**	**100**	**268**

Dari evaluasi yang dilakukan, maka model terbaik untuk dataset box palette adalah model MobileNetV2 dengan 50 epoch. Model ini memiliki F1-score terbaik dengan nilai skor 100%. Model ini lebih dipilih daripada VGG16 karena *running time* yang lebih cepat.

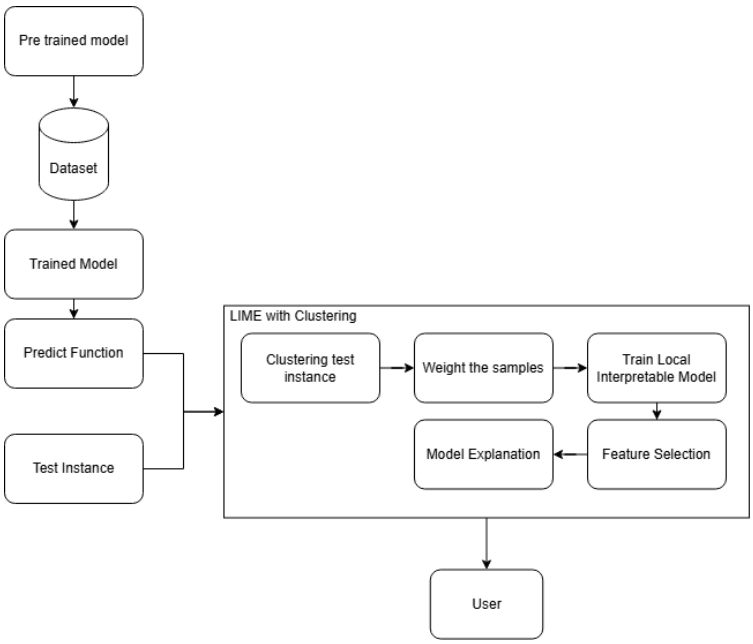
Gambar 4.8 Grafik Akurasi dan Loss pada Training/Validation (palette)

Kemudian, dari grafik akurasi pada gambar 4.8, dapat dilihat grafik akurasi dan loss ketika melakukan pelatihan model ResNet50V2. Dari grafik ini, train_loss dan val_loss menunjukkan tren menurun dan tidak memilik *gap* yang besar. Oleh karena itu dapat dikatakan bahwa model ini tidak menunjukkan indikasi *overfitting.*

4.4 Uji Coba dan Evaluasi LIME

Pada bagian ini, dilakukan uji coba serta evaluasi pada LIME explainable AI. Uji coba yang dilakukan adalah modifikasi terhadap langkah kerja LIME. Kemudian dilakukan perbandingan antara LIME dengan LIME yang dimodifikasi terhadap dataset.

4.4.1 Penambahan Clustering pada LIME



Gambar 4.9 Penambahan Clustering pada Tahapan LIME

Pada tahap ini, dilakukan modifikasi pada cara kerja LIME. Modifikasi dapat dilihat pada gambar 4.9 Modifikasi yang dilakukan adalah dengan menambahkan clustering untuk membuat sampel dataset yang akan digunakan dalam LIME. Digunakan teknik K-Means clustering untuk menggantikan metode segmentasi yang digunakan oleh LIME untuk membuat sampel dataset.

4.4.1.1 Clustering test instance

```
def explain_instance(self, image, classifier_fn, labels=(1,),
                         hide_color=None,
                         top_labels=5, num_features=100000,
                         num_samples=1000,
                         batch_size=10,
                         segmentation_fn=None,
                         distance_metric='cosine',
                         model_regressor=None,
                         random_seed=None,
                         progress_bar=True):
        if len(image.shape) == 2:
            image = gray2rgb(image)
        if random_seed is None:
            random_seed = self.random_state.randint(0, high=1000)

        if segmentation_fn is None:
```

```
            segmentation_fn = SegmentationAlgorithm('quickshift',
                                                    kernel_size=4,
                                                    max_dist=200,
                                                    ratio=0.2,
                                                    random_seed=rand
                                                    om_seed)

        segments = segmentation_fn(image)

(Continuation of the function...)
```

Kode 4.13 Fungsi Segmentasi Bawaan LIME

Pada kode 4.13, ditunjukkan potongan kode yang menjalankan fungsi bawaan segmentasi gambar pada LIME. Segmentasi berjalan di dalam fungsi "explain_instance" pada LIME. Metode segmentasi yang digunakan adalah "quickshift" dengan parameter yang sesuai. Parameter menentukan bagaimana algoritma segmentasi akan berfungsi, mempengaruhi faktor-faktor seperti granularitas dan jangkauan segmen. Setelah "segmentation_fn" diinisialisasi dengan benar, segmentasi dilakukan dengan memanggil fungsi segmentasi dan diterapkan pada gambar *input* dengan variabel "image". Hasil operasi ini disimpan dalam variabel "*segments*". Variabel ini berisi hasil segmentasi yang kemudian digunakan oleh LIME untuk menghasilkan sampel dataset untuk melatih model lokal LIME.

```
def explain_instance_clust(self, image, classifier_fn,
                           labels=(1,),
                           hide_color=None,
                           top_labels=5, num_features=100000,
                           num_samples=1000,
                           batch_size=10,
                           distance_metric='cosine',
                           model_regressor=None,
                           random_seed=None,
                           progress_bar=True,
                           num_clusters=None):  # Add
num_clusters parameter for k-means

        if len(image.shape) == 2:
            image = gray2rgb(image)
        if random_seed is None:
            random_seed = self.random_state.randint(0, high=1000)
```

```
        # Reshape image for k-means clustering
        reshaped_image = image.reshape((-1, 3))

        # Apply PCA to reduce the number of dimensions
        pca = PCA(n_components=2)  # Reduce to 2 components
        pca_image = pca.fit_transform(reshaped_image)
        if num_clusters is None:
            # Automatically determine the optimal number of
clusters
            best_num_clusters = 2
            best_silhouette_score = -1

            for k in range(5, 11):  # Test k values from 5 to 10
                kmeans = KMeans(n_clusters=k, init='k-means++',
                   random_state=random_seed).fit(pca_image)
                cluster_labels = kmeans.labels_
                silhouette_avg = silhouette_score(reshaped_image,
                   cluster_labels)
                if silhouette_avg > best_silhouette_score:
                    best_num_clusters = k
                    best_silhouette_score = silhouette_avg

            num_clusters = best_num_clusters

        # Use k-means clustering for segmentation with k-means++
            initialization
        reshaped_image = image.reshape((-1, 3))
        kmeans = KMeans(n_clusters=num_clusters, init='k-means++',
            random_state=42).fit(reshaped_image)
        segments = kmeans.labels_.reshape(image.shape[:2])
(Continuation of the function...)
```

Kode 4.14 Modifikasi Segmentasi LIME

Kode 4.14 merupakan fungsi “explain_instance_clust”. Fungsi ini merupakan fungsi baru yang dibuat berdasarkan fungsi “explain_instance” pada kode 4.13,. Perbedaannya adalah pada fungsi ini digunakan metode *clustering k-means* untuk segmentasi gambar, sebagai pengganti metode segmentasi *quickshift*. Tahap segmentasi menghasilkan superpiksel yang dapat diinterpretasikan yang kemudian digunakan untuk menghasilkan sampel dataset, seperti fungsi “explain_instance” pada kode 4.13

Kemudian, jika parameter num_clusters tidak ditentukan, dilakukan kalkulasi untuk mencari num_clusters yang optimal dengan menggunakan

silhouette score, yang mengukur seberapa mirip sebuah piksel dengan clusternya sendiri dibandingkan dengan cluster lainnya. Fungsi ini menguji nilai k dari 5 hingga 10, memilih yang memiliki skor tertinggi sebagai nilai num_clusters optimal. *Silhouette score* digunakan karena jumlah *cluster* optimal pada tiap input gambar tidak sama, sehingga penggunaan *silhouette score* dapat menyesuaikan jumlah *cluster* sesuai input gambar.

Setelah jumlah klaster ditentukan, clustering dengan inisialisasi k-means++ diterapkan pada data gambar yang telah diubah bentuknya. Label cluster yang dihasilkan kemudian diubah bentuknya agar sesuai dengan dimensi gambar asli, secara efektif membagi gambar menjadi superpiksel berdasarkan cluster yang diidentifikasi. Segmen-segmen ini kemudian dapat digunakan untuk menghasilkan dataset versi gambar yang terganggu untuk digunakan sebagai data pelatihan model LIME.

4.4.1.2 Weight the samples and feature selection

```
def feature_selection(self, data, labels, weights, num_features,
method):
        """Selects features for the model. see
explain_instance_with_data to
           understand the parameters."""
        if method == 'none':
            return np.array(range(data.shape[1]))
        elif method == 'forward_selection':
            return self.forward_selection(data, labels, weights,
num_features)
        elif method == 'highest_weights':
            clf = Ridge(alpha=0.01, fit_intercept=True,
                        random_state=self.random_state)
            clf.fit(data, labels, sample_weight=weights)

            coef = clf.coef_
            if sp.sparse.issparse(data):
                coef = sp.sparse.csr_matrix(clf.coef_)
                weighted_data = coef.multiply(data[0])
                # Note: most efficient to slice the data before
reversing
                sdata = len(weighted_data.data)
                argsort_data =
np.abs(weighted_data.data).argsort()
```

```
                # Edge case where data is more sparse than
requested number of feature importances
                # In that case, we just pad with zero-valued
features
                if sdata < num_features:
                    nnz_indexes = argsort_data[::-1]
                    indices = weighted_data.indices[nnz_indexes]
                    num_to_pad = num_features - sdata
                    indices = np.concatenate((indices,
np.zeros(num_to_pad, dtype=indices.dtype)))
                    indices_set = set(indices)
                    pad_counter = 0
                    for i in range(data.shape[1]):
                        if i not in indices_set:
                            indices[pad_counter + sdata] = i
                            pad_counter += 1
                            if pad_counter >= num_to_pad:
                                break
                else:
                    nnz_indexes = argsort_data[sdata -
num_features:sdata][::-1]
                    indices = weighted_data.indices[nnz_indexes]
                return indices
            else:
                weighted_data = coef * data[0]
                feature_weights = sorted(
                    zip(range(data.shape[1]), weighted_data),
                    key=lambda x: np.abs(x[1]),
                    reverse=True)
                return np.array([x[0] for x in
feature_weights[:num_features]])
        elif method == 'lasso_path':
            weighted_data = ((data - np.average(data, axis=0,
weights=weights))
                             * np.sqrt(weights[:, np.newaxis]))
            weighted_labels = ((labels - np.average(labels,
weights=weights))
                               * np.sqrt(weights))
            nonzero = range(weighted_data.shape[1])
            _, coefs = self.generate_lars_path(weighted_data,
                                               weighted_labels)
            for i in range(len(coefs.T) - 1, 0, -1):
                nonzero = coefs.T[i].nonzero()[0]
                if len(nonzero) <= num_features:
                    break
            used_features = nonzero
```

```
            return used_features
        elif method == 'auto':
            if num_features <= 6:
                n_method = 'forward_selection'
            else:
                n_method = 'highest_weights'
            return self.feature_selection(data, labels, weights,
                                          num_features, n_method)
```

Kode 4.15 Fungsi Feature Selection dan pembobotan

Fungsi feature_selection dirancang untuk memilih subset fitur dari data masukan yang akan digunakan untuk membangun model lokal yang dapat diinterpretasikan. Fungsi ini mengambil data, label, bobot, jumlah fitur yang diinginkan (num_features), dan metode pemilihan fitur.

4.4.1.3 Training local model and model explanation

```
def explain_instance_with_data(self,
                               neighborhood_data,
                               neighborhood_labels,
                               distances,
                               label,
                               num_features,
                               feature_selection='auto',
                               model_regressor=None):
        # Add debugging statements
        print("Neighborhood data shape:", neighborhood_data.shape)
        print("Neighborhood labels shape:",
neighborhood_labels.shape)
        weights = self.kernel_fn(distances)
        labels_column = neighborhood_labels[:, label]
        used_features = self.feature_selection(neighborhood_data,
                                               labels_column,
                                               weights,
                                               num_features,
                                               feature_selection)
        if model_regressor is None:
            model_regressor = Ridge(alpha=1, fit_intercept=True,
                                    random_state=self.random_state
)
        easy_model = model_regressor
        easy_model.fit(neighborhood_data[:, used_features],
                       labels_column, sample_weight=weights)
        prediction_score = easy_model.score(
            neighborhood_data[:, used_features],
            labels_column, sample_weight=weights)
```

```
        local_pred = easy_model.predict(neighborhood_data[0,
used_features].reshape(1, -1))

        if self.verbose:
            print('Intercept', easy_model.intercept_)
            print('Prediction_local', local_pred,)
            print('Right:', neighborhood_labels[0, label])
        return (easy_model.intercept_,
                sorted(zip(used_features, easy_model.coef_),
                       key=lambda x: np.abs(x[1]), reverse=True),
                prediction_score, local_pred)
```

Kode 4.16 Kode fungsi *explain instance with data*

Fungsi "explain_instance_with_data" memberikan penjelasan terkait prediksi model *machine learning* untuk *input* tertentu dengan melatih model lokal yang dapat diinterpretasi. Parameter yang dibutuhkan adalah data *input* yang terganggu, label, jarak dari instance asli, label yang akan dijelaskan, jumlah fitur, metode pemilihan fitur, dan model regressor opsional. Fungsi ini menghitung bobot untuk *input* yang terganggu menggunakan fungsi kernel dan memilih fitur yang relevan berdasarkan metode yang ditentukan. Kemudian melatih model lokal yang dapat diinterpretasikan menggunakan data berbobot dan fitur yang dipilih. Fungsi ini menghitung skor R^2 (fidelity) model lokal dan membuat prediksi untuk *input* aslinya. Terakhir, fungsi ini mengembalikan intersep model, daftar bobot fitur yang diurutkan, skor R^2, dan prediksi lokal.

4.4.2 Kode Inisiasi LIME

```
import tensorflow as tf
# Load the saved model
model = tf.keras.models.load_model('D:/OneDrive - 2/OneDrive -
Institut Teknologi Sepuluh Nopember/Model_XAI/saved
model/palet_ResNet50V2_50.keras')
```

Kode 4.17 Kode load model

Pertama dijalankan kode 4.17 untuk memuat model yang telah dilatih sebelumnya dengan menggunakan library tensorflow. Model inilah yang nantinya akan digunakan pada LIME sebagai model *black box*.

```
from PIL import Image

# Preprocess the input image
def preprocess_image(image_path):
```

```
    img = Image.open(image_path)
    img = img.resize((180, 320))  # Resize the image to match the
input size of the model
    img = np.array(img) / 255.0  # Normalize the pixel values to
the range of [0, 1]
    return img

# Define the path to the class labels file
class_labels_file = 'class_labels_combined.txt'

# Read the class labels from the file
with open(class_labels_file, 'r') as file:
    class_labels = [line.strip() for line in file]  # Replace with
your actual class labels

# Predict the class for a new image
image_path = 'D:/dataset tesis/combined/63_Pola1_Salah/IMG-
20231024-WA0026.jpg'

# preprocessed_image = preprocess_image(image_path)
preprocessed_image = preprocess_image(image_path)
preprocessed_image_expanded = np.expand_dims(preprocessed_image,
axis=0)  # Add a batch dimension for prediction

# Measure the prediction time
start_time = time.time()
predictions = model.predict(preprocessed_image_expanded)
end_time = time.time()
prediction_time = end_time - start_time

predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]
confidence = predictions[0][predicted_class_index]

print("Predicted class:", predicted_class_label)
print("Confidence:", confidence)
print("Prediction time:", prediction_time, "seconds")


# Display the image
plt.imshow(preprocessed_image)
plt.axis('off')  # Turn off the axis labels
plt.show()
```

Kode 4.18 Kode melakukan prediksi model deep learning

Kemudian dengan kode 4.16 dilakukan prediksi dengan model deep learning yang dimuat.

```
from explainer_image import LimeImageExplainer
explainer = LimeImageExplainer(
kernel_width=0.25, kernel=None, verbose=False,
feature_selection='auto', random_state=42
)
```

Kode 4.19 Kode memuat LIME explainer

Dengan menggunakan kode 4.19, dilakukan pemuatan LIME *explainer* agar dapat memanggil fungsi-fungsi yang disediakan oleh *library* LIME.

```
from skimage.segmentation import mark_boundaries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import jaccard_score
from scipy.spatial.distance import cosine

# Store explanations, fidelity scores, and masks
lime_stored = []
lime_feature_weights = []
fidelity_scores = []
lime_masks = []

# Define a prediction function for LIME
def predict_function(images):
    return model.predict(images, verbose=0)

# Number of times to repeat the explanation process
num_repeats = 5
num_samples = 1000

for i in range(num_repeats):
    # Explain the prediction using LIME
    exp_lime = explainer.explain_instance(
        preprocessed_image,
        predict_function,
        top_labels=1,
        hide_color=0,
        num_samples=num_samples,
        batch_size=10,
        progress_bar=True
    )

    # Get the image and mask
```

```
    temp, mask = exp_lime.get_image_and_mask(
        label=exp_lime.top_labels[0],
        positive_only=True,
        num_features=5,
        hide_rest=False
    )

    # Display and save the explanation using explicit figure and
axes
    fig, ax = plt.subplots()
    ax.imshow(mark_boundaries(temp / 2 + 0.5, mask))
    ax.set_title(f'LIME Explanation')
    plt.show()
    save_path = os.path.join(save_dir,
f'lime_pallet_explanations_{i}.png')
    fig.savefig(save_path)
    plt.close(fig)  # Ensure the plot is properly closed

    # Evaluate Local Fidelity
    local_fidelity = exp_lime.score
    fidelity_scores.append(local_fidelity)
    print(f"Local Fidelity (R² of the local surrogate model):
{local_fidelity}")

    # Store the explanation
    feature_weights = exp_lime.local_exp[exp_lime.top_labels[0]]
    lime_stored.append(feature_weights)
    lime_feature_weights.append(feature_weights)
    lime_masks.append(mask)
```

Kode 4.20 Kode untuk menghasilkan penjelasan LIME

Kemudian, digunakan kode 4.20 untuk menghasilkan penjelasan LIME menggunakan input data dan model yang telah dimuat sebelumnya. Kode ini akan menghasilkan luaran gambar prediksi yang diberi sorotan fitur dan skor *fidelity*. Kemudian pada kode ini, bobot fitur dan *mask* akan disimpan untuk penghitungan skor *jaccard* dan *cosine similarity* di tahap berikutnya.

```
import numpy as np
from sklearn.metrics import jaccard_score
from scipy.spatial.distance import cosine
import matplotlib.pyplot as plt
from skimage.segmentation import mark_boundaries
```

```
# Store explanations and masks
clime_stored = []
clime_feature_weights = []
clime_masks = []

# Define a prediction function for LIME
def predict_function(images):
    return model.predict(images, verbose=0)

num_repeats = 5
num_clusters = None
num_samples = 1000

for i in range(num_repeats):
    # Explain the prediction using LIME
    exp_lime_clust = explainer.explain_instance_clust(
        preprocessed_image,
        predict_function,
        top_labels=1,
        hide_color=0,
        num_samples=num_samples,
        batch_size=10,
        num_clusters=num_clusters,
        progress_bar=True
    )

    # Get the image and mask from the ImageExplanation object
    temp, mask = exp_lime_clust.get_image_and_mask(
        label=exp_lime_clust.top_labels[0],
        positive_only=True,
        num_features=5,
        hide_rest=False
    )

    # Display the explanation
    fig, ax = plt.subplots()
    ax.imshow(mark_boundaries(temp / 2 + 0.5, mask))
    ax.set_title(f'LIME Explanation with Clustering
(clusters={num_clusters})')
    plt.show()
    save_path = os.path.join(save_dir,
f'clime_{save_dir}_{i}.png')
    plt.savefig(save_path)
    fig.savefig(save_path)
    plt.close(fig)  # Ensure the plot is properly closed
```

```
    # Evaluate Local Fidelity
    local_fidelity = exp_lime_clust.score
    print(f"Local Fidelity (R² of the local surrogate model) with
num_clusters: {local_fidelity}")

    # Store the explanation
    feature_weights =
exp_lime_clust.local_exp[exp_lime_clust.top_labels[0]]
    clime_stored.append(feature_weights)
    clime_masks.append(mask)

    # Collect all feature weights
    clime_feature_weights.append(feature_weights)
```

Kode 4.21 Kode untuk menghasilkan penjelasan LIME clustering

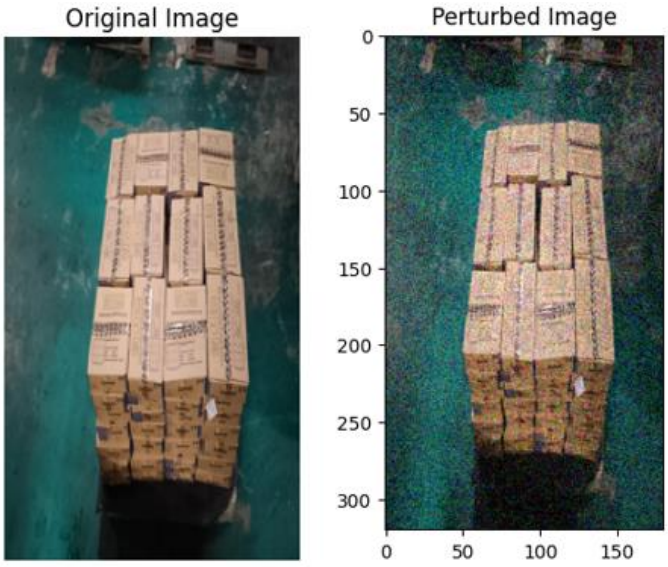
Kode 4.21 merupakan kode untuk menghasilkan penjelasan dengan menggunakan LIME clustering. Secara garis besar, kode 4.19 sama dengan kode 4.20. Perbedaannya adalah pada kode 4.18 menggunakan fungsi "explainer.explain_instance" (LIME biasa), sedangkan pada kode 4.19 menggunakan fungsi "explainer.explain_instance_clust" (LIME clustering). Selain itu, terdapat parameter tambahan yaitu "num_cluster" karena menggunakan LIME clustering. Parameter ini dapat diisi dengan cluster yang diinginkan atau diisi dengna "None" agar nilai parameter optimal dapat ditentukan oleh fungsi dengan menggunakan *silhouette score*.

```
# Perturb the input image
    perturbed_images = perturb_image(preprocessed_image,
num_perturbations, noise_level)

    # Generate explanations for perturbed images and compute
stability score
    stability_score = 0
    for perturbed_image in perturbed_images:
        exp_perturbed = explainer.explain_instance_clust(
            perturbed_image,
            predict_function,
            top_labels=1,
            hide_color=0,
            num_samples=num_samples,
            batch_size=10,
            num_clusters=num_clusters,
```

```
            progress_bar=False  # Turn off progress bar for
perturbations
        )
        feature_weights_perturbed =
dict(exp_perturbed.local_exp[exp_perturbed.top_labels[0]])

        # Align the feature weights by creating a union of all
features
        all_features =
set(feature_weights_original.keys()).union(set(feature_weights_per
turbed.keys()))
        aligned_orig =
np.array([feature_weights_original.get(feat, 0) for feat in
all_features])
        aligned_perturbed =
np.array([feature_weights_perturbed.get(feat, 0) for feat in
all_features])

        # Calculate the squared L2 norm difference between
original and perturbed explanations
        weight_diff = aligned_orig - aligned_perturbed
        stability_score += np.linalg.norm(weight_diff, 2)**2

    # Average stability score over all perturbations
    stability_score /= num_perturbations
    stability_scores.append(stability_score)
    print(f"Stability Score: {stability_score}")
```

Kode 4.22 Kode untuk mengukur *jaccard* dan *cosine similarity*

Kode 4.22 digunakan untuk mengukur metrik *stability* dengan menggunakan *L2 distance*. Skor ini digunakan sebagai metrik pengukuran stabilitas penjelasan LIME. Pengukuran metrik ini dilakukan dengan mengulang penjelasan LIME sebanyak 10 kali, namun di tiap pengulangan input asli diberi sedikit gangguan (noise) agar dapat melihat pengaruh perubahan input terhadap penjelasan LIME. Noise yang digunakan pada proses ini adalah *gaussian noise*. *Gaussian noise* adalah jenis *noise* statistik yang memiliki fungsi kepadatan probabilitas sama dengan distribusi normal, yang juga dikenal sebagai distribusi *Gaussian*. *Gaussian noise* digunakan karena menyerupai *noise* di dunia nyata, mudah dikendalikan, dan membantu dalam menilai stabilitas dan ketahanan model secara efektif.



Gambar 4.10 Contoh Gambar Asli dan Gambar diberi Gangguan/Noise

Kemudian, skor stability yang didapat menandakan kemiripan antara hasil penjelasan input tanpa gangguan dengan hasil penjelasan input dengan gangguan. Sehingga skor stability yang semakin kecil menandakan stabilitas yang baik karena penjelasan yang dihasilkan masih mirip, meskipun input diberi gangguan.

4.4.3 Uji Coba dan Evaluasi LIME

Pada tahap ini dilakukan uji coba serta evaluasi LIME. Metode LIME tanpa modifikasi dan LIME dengan clustering diujikan ke tiga dataset, yaitu dataset flower, Intel Images, dan dataset box pallete. Kemudian hasilnya akan dievaluasi.

4.4.3.1 Uji coba pada dataset *Flower Images*

Pertama dilakukan percobaan LIME dan LIME K-Means Clustering terhadap dataset flower. Pengujian dilakukan untuk mengukur skor *fidelity* dan kestabilan hasil penjelasan LIME. Pengujian dilakukan terhadap 50 data pada dataset.

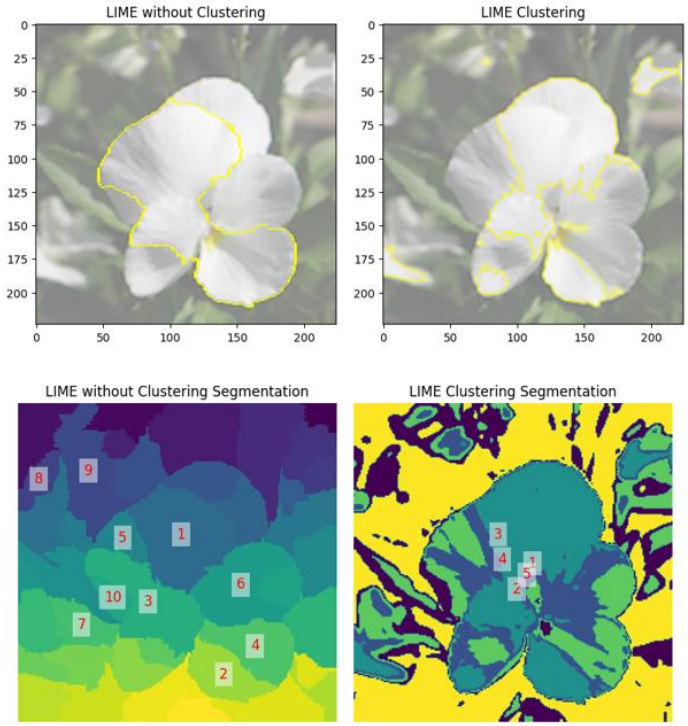
Tabel 4.7 Hasil Skor Fidelity dan Stability pada Dataset Flower

Num Samples	LIME tanpa Clustering		LIME Clustering	
	Avg Fidelity (50 Images)	*Avg Stability (3 images, 1000 samples)*	*Avg Fidelity* (50 Images)	*Avg Stability (3 images, 1000 samples)*
500	0.333	**0.282**	**0.4644**	0.530

1000	0.294		**0.4639**	
2500	0.273		**0.4639**	

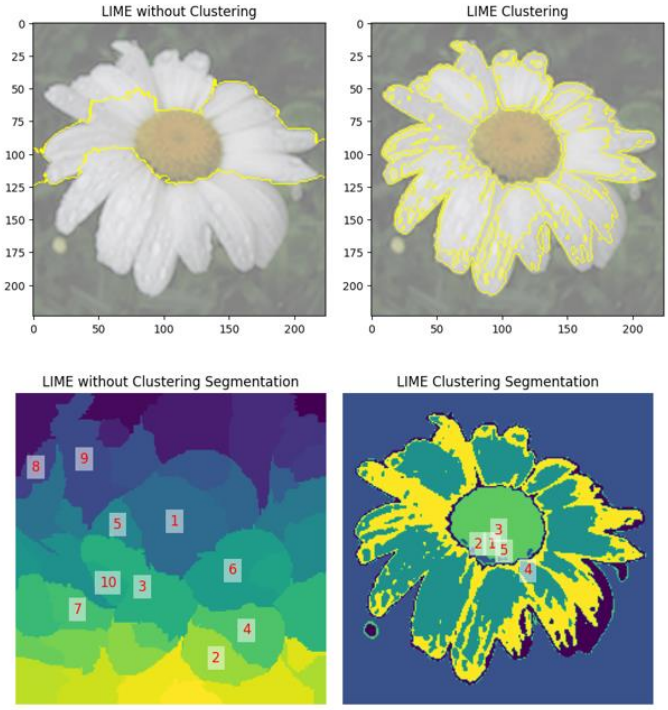
Pada tabel 4.7 dapat dilihat rata-rata skor *fidelity* dari masing-masing metode LIME untuk jumlah sampel 500, 1000, dan 2500. Pada metode LIME, hasil menunjukkan bahwa semakin kecil jumlah sampelnya, maka rata-rata skor *fidelity* akan lebih baik. Sedangkan pada LIME *Clustering*, rata-rata skor *fidelity* relatif sama untuk tiap jumlah sampel. Kemudian, LIME clustering menghasilkan skor *fidelity* yang lebih tinggi di semua pengujian jumlah sampel.

Kemudian, dapat dilihat juga hasil evaluasi LIME untuk mengukur kestabilan hasil penjelasan. Pengujian dilakukan dengan mengulang penjelasan LIME sebanyak lima kali untuk satu input yang sama. Pengujian dilakukan pada tiga kelas gambar, yaitu *viola*, *leucantheum,* dan *calendula.* Kemudian hasil penjelasan dari tiap iterasi akan diukur kestabilannya dengan menggunakan metrik *stability*. Dari hasil yang didapat pada dataset *flower images*, dapat dilihat bahwa hasil LIME biasa memiliki skor *stability* yang lebih baik secara rata-rata.



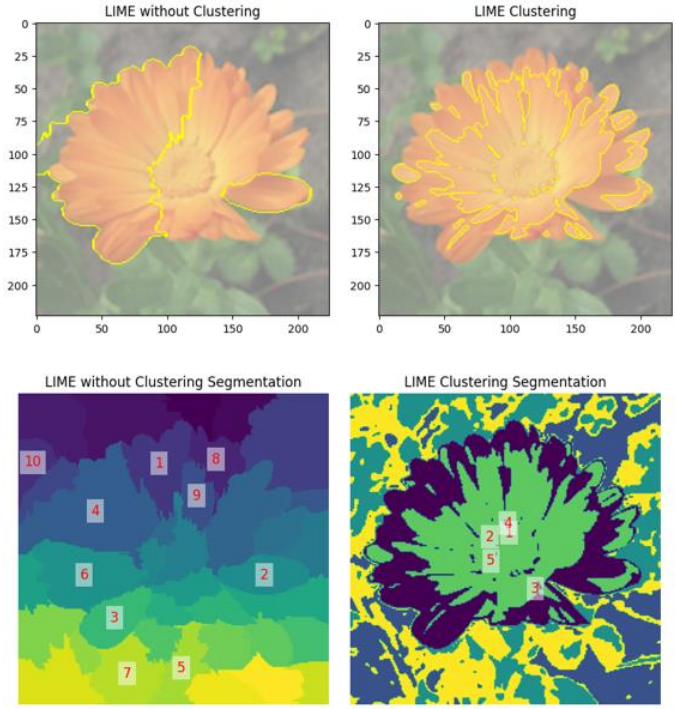
(a) (b)

Gambar 4.11 Perbandingan Penjelasan LIME tanpa clustering dan LIME clustering (*viola*)



(a) (b)

Gambar 4.12 Perbandingan Penjelasan LIME tanpa clustering dan LIME clustering (*leucanthemum*)



(a) (b)

Gambar 4.13 Perbandingan Penjelasan LIME tanpa clustering dan LIME clustering (*calendula*)

Kemudian, pada gambar 4.11, 4.12, dan 4.13 dapat dilihat perbandingan antara penjelasan LIME tanpa clustering dengan LIME yang menggunakan K Means. Pada metode LIME tanpa clustering, hasil penjelasan menyoroti fitur kelopak bunga sebagai ciri bunga. Sedangkan hasil dari LIME clustering juga menyoroti kelopak bunga, namun lebih menyeluruh.

Kemudian angka pada hasil segmentasi menunjukkan peringkat bobot segmentasi berdasarkan hasil prediksi model. Dapat dilihat bahwa pada hasil segmentasi kedua metode, model berhasil memprediksi segmen yang penting pada bagian tengah bunga.

4.4.3.2 Uji coba pada dataset Intel Images

Berikutnya, pengujian LIME dilakukan dengan menggunakan dataset Intel Images.

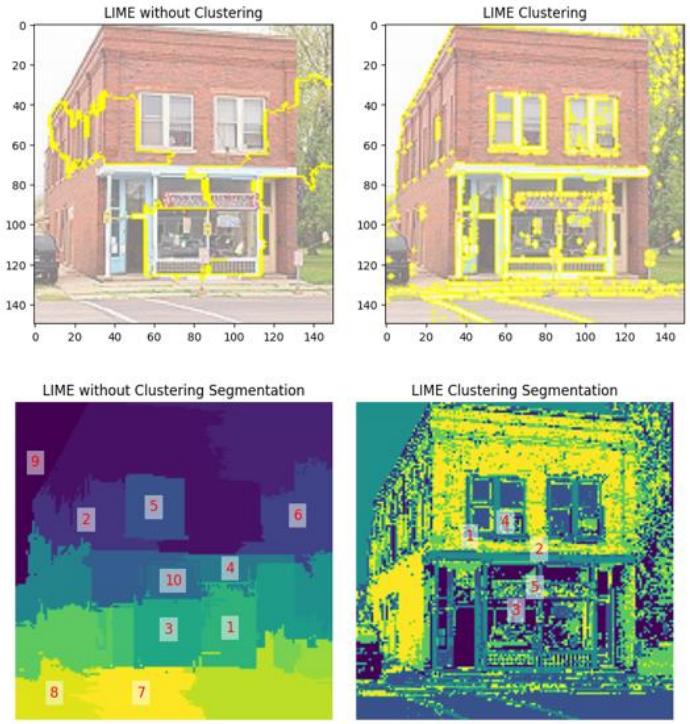
Tabel 4.8 Hasil Skor Fidelity dan Stability pada Dataset Intel

Num Samples	LIME tanpa Clustering		LIME Clustering	
	Avg Fidelity (50 Images)	*Avg Stability (3 images, 1000 samples)*	*Avg Fidelity* (50 Images)	*Avg Stability (3 images, 1000 samples)*
500	**0.4794**	0.118	0.4558	**0.079**
1000	0.457		**0.4631**	
2500	0.461		**0.5291**	

Pada tabel 4.8 dapat dilihat hasil evaluasi skor *fidelity* dari kedua metode LIME. Pada LIME standar, skor rata-rata *fidelity* tertinggi terjadi pada evaluasi dengan jumlah sampel 500, yaitu dengan skor 0,4794. Sedangkan pada metode LIME dengan *clustering*, skor *fidelity* yang dihasilkan unggul pada pengujian dengan 1000 dan 2500 sampel.. Dari hasil ini, LIME clustering dapat meningkatkan skor rata-rata *fidelity*.

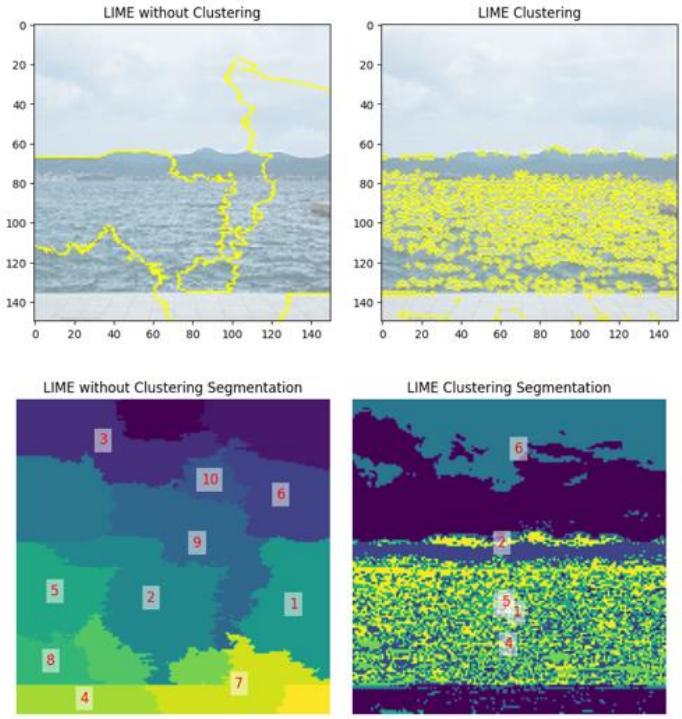
Selain itu, ditunjukkan juga hasil evaluasi LIME untuk mengukur kestabilan hasil penjelasan pada dataset intel images. Pengujian dilakukan dengan mengulang penjelasan LIME sebanyak sepuluh kali dengan *input* yang diberi sedikit gangguan di tiap pengulangannya. Pengujian dilakukan pada 3 kelas gambar, yaitu *buildings*,

sea, dan *mountain*. Dari hasil yang ada dapat dilihat bahwa metode LIME dengan clustering menghasilkan skor *stability* yang lebih baik dibanding LIME biasa pada semua input pengujian.



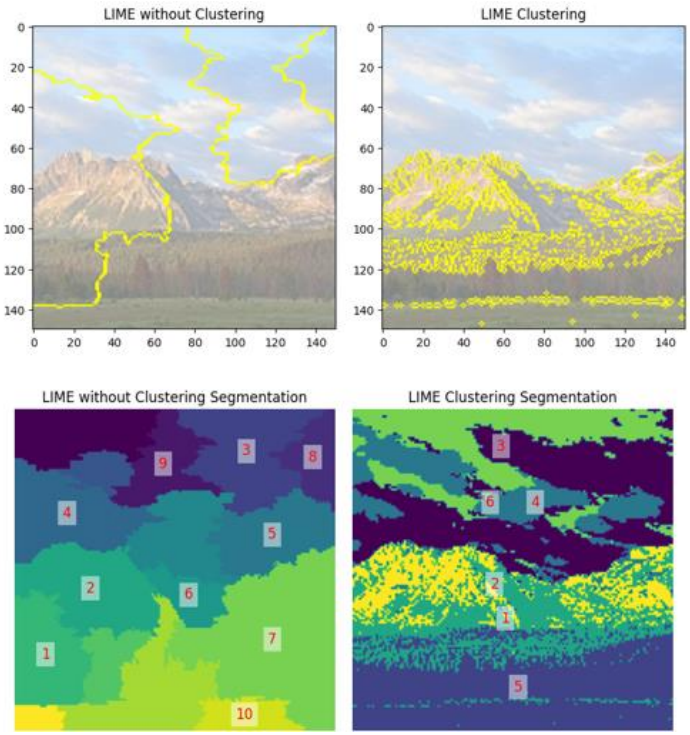
(a) (b)

Gambar 4.14 Perbandingan LIME tanpa clustering dan LIME Clustering (*buildings*)



(a) (b)

Gambar 4.15 Perbandingan LIME tanpa clustering dan LIME Clustering (*sea*)



(a) (b)

Gambar 4.16 Perbandingan LIME tanpa clustering dan LIME Clustering (*mountain*)

Kemudian, pada gambar 4.14, 4.15, 4.16 dapat dilihat perbandingan antara penjelasan LIME tanpa clustering dengan LIME clustering. Hasil penjelasan LIME biasa menyoroti bentuk gedung pada gambar gedung, bentuk laut pada gambar laut, dan bentuk gunung serta langit pada gambar gunung. Lalu pada hasil LIME clustering, disoroti fitur yang kurang lebih sama dengan hasil penjelasan LIME, terutama pada gambar laut dan gunung yang menyoroti fitur gambar yang lebih sesuai dengan kelasnya (menyoroti gunung pada gambar gunung dan laut pada gambar laut). Kemudian angka pada hasil segmentasi menunjukkan peringkat bobot segmentasi berdasarkan hasil prediksi model.

Kemudian angka pada hasil segmentasi menunjukkan peringkat bobot segmentasi berdasarkan hasil prediksi model. Dapat dilihat bahwa pada hasil segmentasi kedua metode, segmen yang penting terletak pada bagian yang menjadi ciri tiap kelas.

4.4.3.3 Uji coba pada dataset box pallete

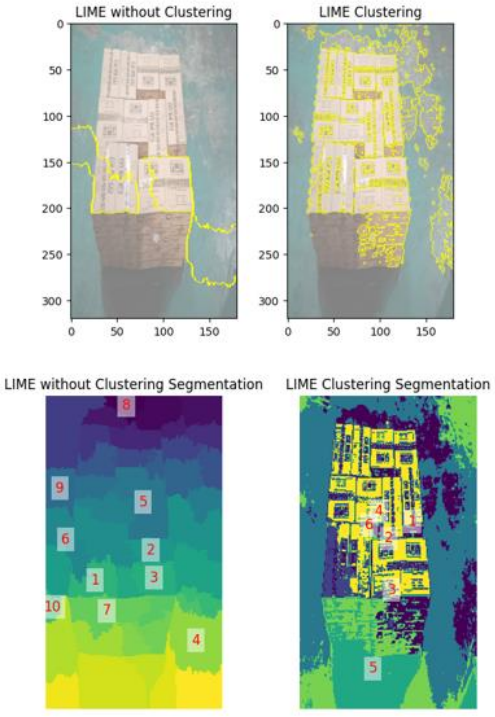
Pada bagian ini, LIME kembali diujikan ke dataset lain, yaitu dataset susunan box palette yang sudah dikumpulkan.

Tabel 4.9 Hasil Skor Fidelity dan Stability pada Dataset Pallet

Num Samples	LIME tanpa Clustering		LIME Clustering	
	Avg Fidelity (50 Images)	*Avg Stability (3 images, 1000 samples)*	*Avg Fidelity* (50 Images)	*Avg Stability (3 images, 1000 samples)*
500	0.2666	**0.183**	**0.5947**	0.370
1000	0.215		**0.5884**	
2500	0.1752		**0.635**	

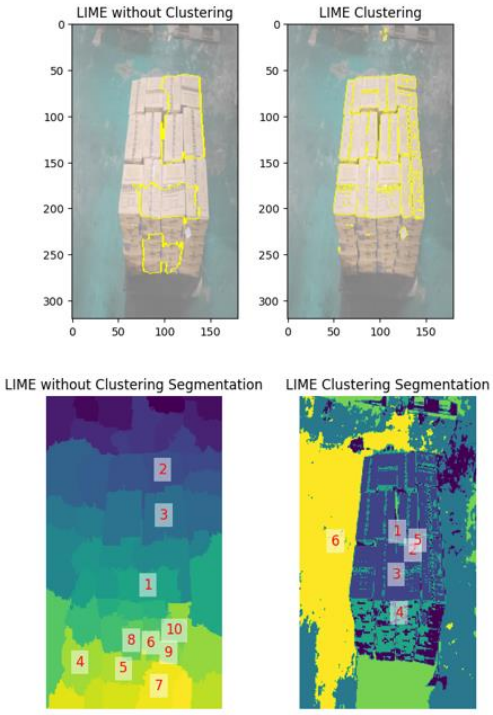
Kemudian, pada tabel 4.8 dapat dilihat hasil evaluasi skor fidelity untuk tiap metode LIME. Ketika menggunakan LIME biasa, rata-rata skor fidelity tertinggi didapatkan ketika menggunakan jumlah sampel 500. Kemudian ketika menggunakan LIME clustering, skor tertinggi didapat ketika menggunakan jumlah sampel 2500, yaitu sebesar 0,635. Secara keseluruhan, rata-rata skor fidelity pada LIME clustering meningkat dibanding ketika menggunakan LIME standar.

Kemudian, dapat dilihat juga hasil evaluasi LIME untuk mengukur kestabilan hasil penjelasan pada dataset box palette. Pengujian dilakukan dengan mengulang penjelasan LIME sebanyak sepuluh kali dengan input yang sama namun diberi sedikit gangguan. Dari hasil yang ada dapat dilihat bahwa metode LIME biasa menghasilkan skor *stability* yang lebih baik dibanding LIME *clustering.*



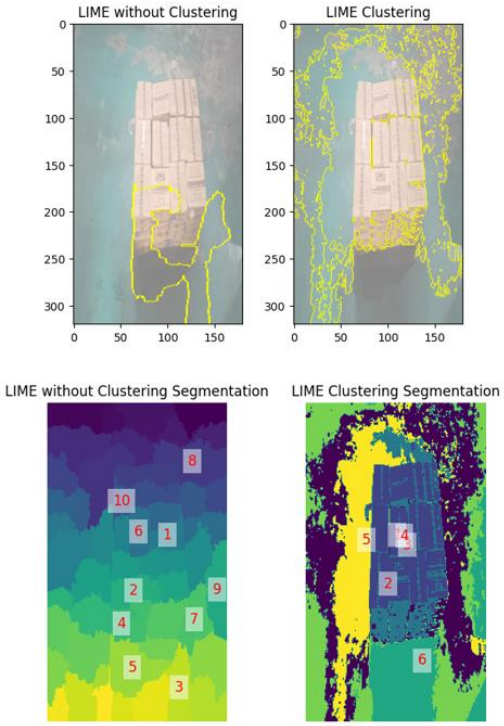
(a) (b)

Gambar 4.17 Perbandingan LIME tanpa clustering dan LIME Clustering (60 Pola 1 Benar)



(a) (b)

Gambar 4.18 Perbandingan LIME tanpa clustering dan LIME Clustering (63 Pola 2 Benar)



(a) (b)

Gambar 4.19 Perbandingan LIME tanpa clustering dan LIME Clustering (63 Pola 1 Salah)

Kemudian, pada gambar 4.17, 4.18, dan 4.19 dapat dilihat perbandingan antara penjelasan LIME tanpa clustering dengan LIME clustering pada dataset box palette. Hasil LIME biasa berhasil menyoroti fitur sebagian pola box dan sedikit menyoroti bagian lantai, khususnya pada kelas 63 pola 1 salah. Sedangkan hasil LIME clustering juga berhasil menyoroti box sebagai fitur penting, dengan menyoroti sedikit lantai pada kelas 60 pola 1 benar. Namun, pada gambar 63 pola 1 salah, hasil penjelasan lebih banyak menyoroti lantai dengan sedikit sorotan pada box. Hal ini sesuai dengan hasil penjelasan LIME tanpa clustering yang juga sebagian besar menyoroti lantai dibanding box, jika dibandingkan dengan dua kelas lainnya.

Kemudian angka pada hasil segmentasi menunjukkan peringkat bobot segmentasi berdasarkan hasil prediksi model. Dapat dilihat bahwa pada hasil segmentasi kedua metode, model berhasil memprediksi segmen yang menjadi ciri tiap kelas, yaitu pada bagian box.

4.5 Evaluasi Tambahan

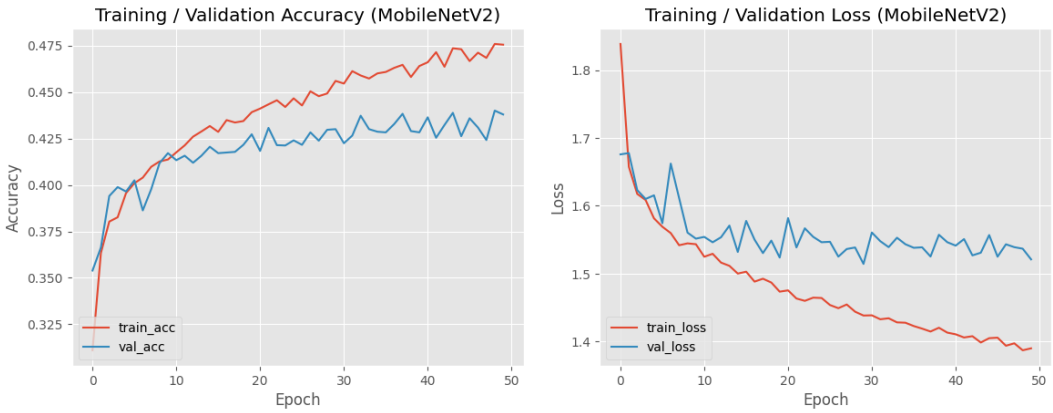
Pada tahap ini, dilakukan evaluasi tambahan dengan menggunakan dataset wajah untuk klasifikasi emosi. Dataset yang digunakan adalah *Facial Expressions Training Data* yang dibuat oleh Noam Segal berdasarkan Affectnet dataset. Dataset ini berisi 29.042 gambar yang dibagi dalam delapan kelas, yaitu *anger, contempt, disgust, fear, happy, neutral, sad,* dan *surprise.* Seluruh gambar pada dataset memiliki dimensi 96 x 6 dan berwarna.

Gambar 4.20 Dataset *Facial Expression*

Kemudian, dilakukan training model deep learning dengan menggunakan MobileNetV2 sebagai base model. Model ini dipilih karena berdasarkan evaluasi pada tiga dataset sebelumnya, model ini memiliki performa terbaik. Kemudian, digunakan head model yang sama dengan evaluasi sebelumnya, yaitu satu lapisan *Flatten*, satu lapisan *Dense* 512, dan satu lapisan *Dense* 8 sebagai *output*. Pelatihan model dilakukan dengan 50 *epoch*.

Tabel 4.10 Evaluasi Model pada Dataset Facial

Base Model	Performa Model (%)				*Running Time (s)*
	Accuracy	*Precision*	*Recall*	*F1-Score*	
MobileNetV2	43	42	43	42	2747



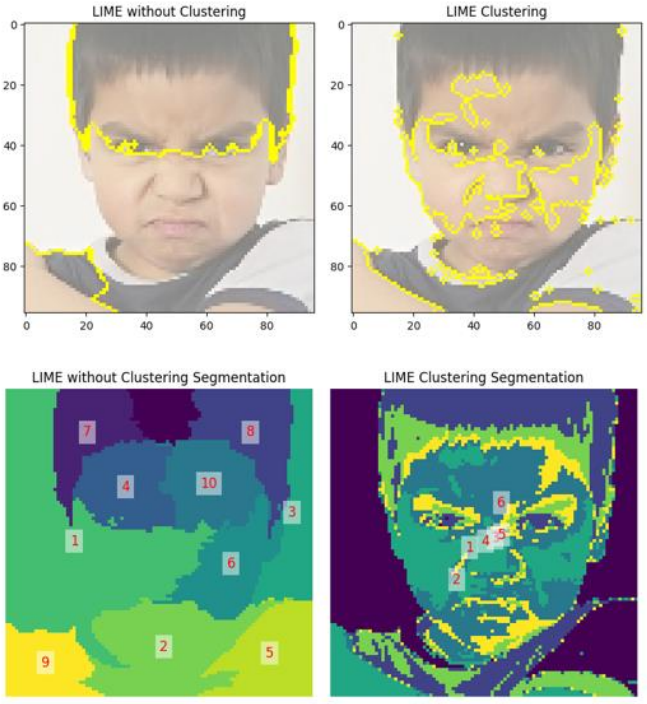
Gambar 4.21 Grafik Akurasi dan Loss pada Training/Validation (facial)

Berdasarkan tabel 4.10, dapat dilihat model memiliki F1 score 42% pada dataset facial. Meskipun performa model tidak maksimal, grafik akurasi dan *loss* pada *training*/*validation* menunjukkan bahwa model dapat mempelajari dataset dengan cukup baik namun pada epoch setelah 30 mulai menunjukkan *gap* antara akurasi dan *loss* pada *training* dan *validation*.

Tabel 4.11 Hasil Skor Fidelity dan Stability pada Dataset Facial

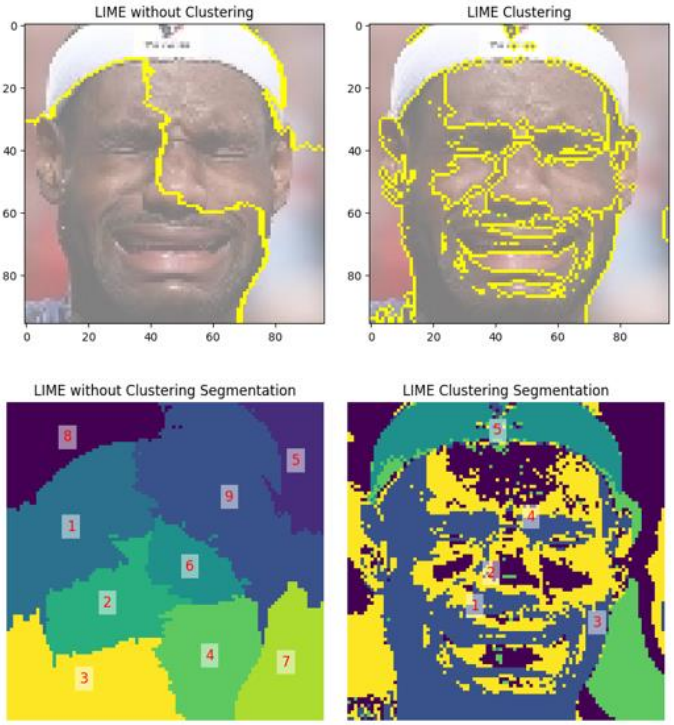
Class	LIME tanpa Clustering		LIME Clustering	
	Avg Fidelity (11 iteration)	*Avg Stability (3 images, 1000 samples)*	*Avg Fidelity* (50 Images)	*Avg Stability (3 images, 1000 samples)*
anger	0.616	**0.085**	**0.807**	0.19
sad	0.360		**0.529**	
happy	0.402		**0.582**	

Berdasarkan hasil pada tabel 4.11, dapat dilihat bahwa skor rata-rata fidelity lebih baik pada hasil LIME clustering, dan skor rata-rata stability lebih baik pada hasil LIME tanpa clustering. Hasil ini sesuai dengan percobaan sebelumnya yang telah dilakukan pada dataset *flower, intel images,* dan *box palette.*



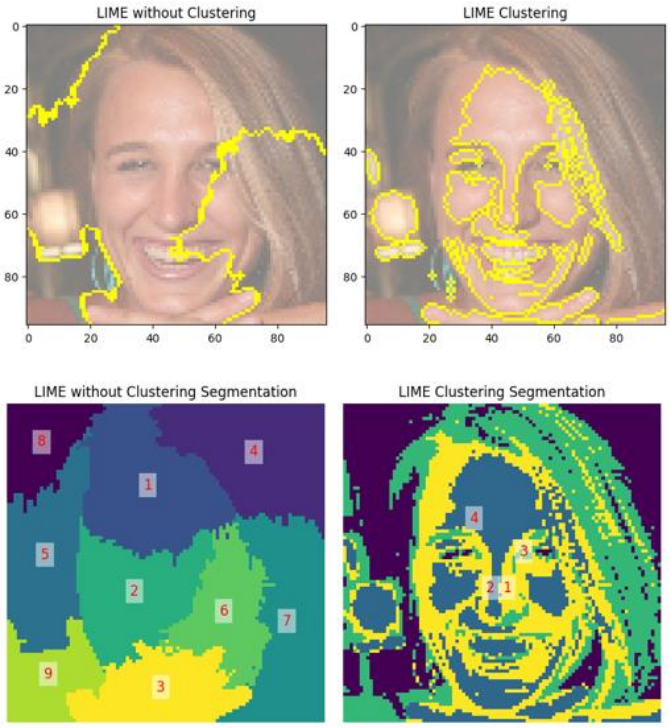
(a) (b)

Gambar 4.22 Perbandingan LIME tanpa clustering dan LIME Clustering (anger)



(a) (b)

Gambar 4.23 Perbandingan LIME tanpa clustering dan LIME Clustering (sad)



(a) s(b)

Gambar 4.24 Perbandingan LIME tanpa clustering dan LIME Clustering (happy)

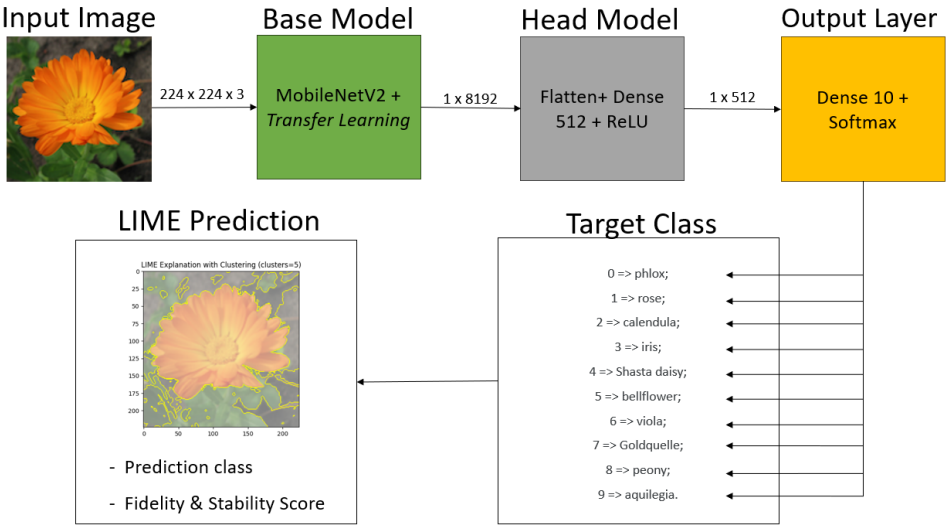
Pada gambar 4.22, 4.23, 4.24 dapat dilihat perbandingan hasil penjelasan LIME tanpa clustering dengan LIME clustering. Hasil penjelasan LIME tanpa clustering pada ketiga kelas berhasil menyoroti fitur penting pada gambar. Hal ini dapat dilihat dari peringkat segmentasi pada gambar yang disoroti adalah peringkat yang tinggi yaitu 1-5 dan dijadikan satu segmen yang disorot. Kemudian hasil penjelasan LIME clustering juga berhasil menyoroti fitur penting pada gambar. Namun, pada hasil LIME clustering, yang disoroti hanyalah segmen peringkat 1 yang paling menunjukkan fitur penting dari gambar.

4.6 Pembahasan Hasil Eksperimen

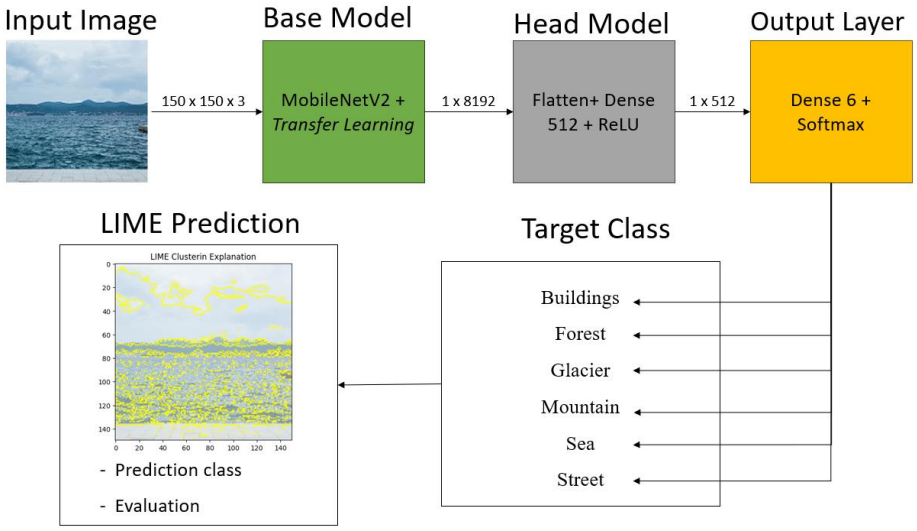
Pada penelitian tesis ini, telah dilakukan beberapa eksperimen. Eksperimen yang dilakukan bertujuan untuk mengukur performa klasifikasi gambar dengan menggunakan model Deep Learning dan untuk mengukur performa LIME dalam menjelaskan hasil prediksi model deep learning.

Pada eksperimen model, dilakukan evaluasi untuk mengukur performa model berdasarkan skor F1. Dari hasil eksperimen performa model deep learning, arsitektur model yang diusulkan memiliki performa terbaik dengan menggunakan *base model* MobileNetV2 pada dataset *flower images*, *intel images* dan *box palette*. Skor F1 yang didapatkan adalah 88% untuk dataset *flower images*, 92% untuk dataset *intel images* dan 100% untuk dataset *box palette*.

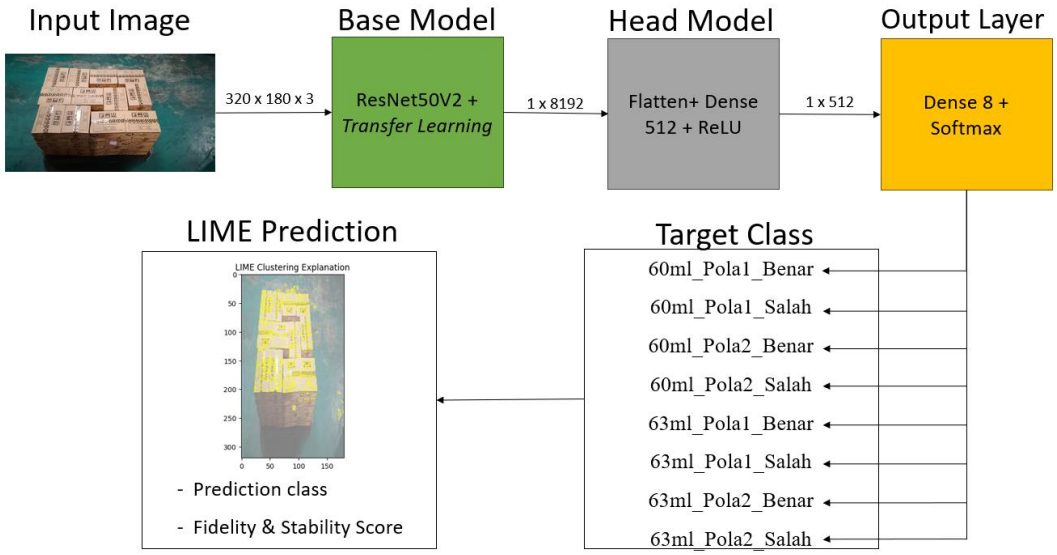
Pada dataset *flower images*, performa model terbaik adalah ketika pelatihan model dilakukan sebanyak 50 *epochs*. Lalu pada dataset *intel images* performa terbaik didapatkan dengan 50 *epochs*. Terakhir, pada dataset *box palette*, performa terbaik didapatkan dengan 50 *epochs*.



Gambar 4.25 Arsitektur Model pada Dataset *Flower*



Gambar 4.26 Arsitektur Model pada Dataset *Intel Images*



Gambar 4.27 Arsitektur Model pada Dataset *Box Palette*

Dari hasil evaluasi yang dilakukan, maka arsitektur model untuk tiap dataset dapat dilihat pada gambar 4.13, 4.14, dan 4.15. Pada dataset flower dan intel images, digunakan *base model* MobileNetV2. MobileNetV2 adalah model ringan dan efisien yang telah dilatih sebelumnya pada kumpulan data ImageNet. Dengan mengatur include_top=False, lapisan atas (klasifikasi) MobileNetV2 tidak ikut dalam pelatihan model, dengan fokus pada penggunaan kemampuan ekstraksi fiturnya. Parameter input_tensor membentuk ulang input ke format 224 x 224 x 3 (dataset flower) atau 150 x 150 x 3 (dataset intel), karena gambar berwarna. Lapisan model dasar kemudian dibekukan (baseModel.trainable = False), agar lapisan tersebut tidak diperbarui selama pelatihan sehingga dapat mempertahankan fitur yang dipelajari. Setelah *base model*, kemudian ditambahakan *head model*. Pertama, *output* MobileNetV2 diratakan menggunakan lapisan Flatten untuk mengubah matriks fitur 2D menjadi vektor 1D. Selanjutnya, lapisan padat dengan 512 unit dan fungsi aktivasi ReLU ditambahkan, yang memperkenalkan non-linearitas dan membantu mempelajari pola yang kompleks. Terakhir, lapisan padat lainnya ditambahkan dengan jumlah unit yang sama dengan jumlah kelas yang sesuai, yaitu 10 untuk dataset *flower* dan 6 untuk dataset *intel images*. Kemudian digunakan fungsi aktivasi softmax, menghasilkan distribusi probabilitas pada kelas-kelas untuk klasifikasi *multiclass*. Kemudian model terakhir menggabungkan *input base*

model MobileNetV2 dengan *head model* yang baru dibuat, sehingga menghasilkan model yang siap untuk dilatih pada tugas klasifikasi.

Untuk dataset *box palette*, perbedaan hanya terletak di *base model* dan *output layer*. Pada *base model*, digunakan ResNet50V2 dengan dimensi input 320 x 180 x 3. Kemudian pada *output layer*, digunakan Dense 8 + softmax, karena menyesuaikan dataset *box palette* yang memiliki delapan kelas.

Kemudian, pada eksperimen LIME dilakukan dua eksperimen untuk mengukur skor *fidelity* (R^2 *Score)* dan *stability (L2 Distance)*. Eksperimen dilakukan terhadap dua metode LIME, yaitu metode LIME standar, dan metode usulan LIME menggunakan *K-Means Clustering*. Usulan metode LIME dengan K-Means clustering didasarkan pada penelitian terdahulu yang menggunakan metode clustering untuk meningkatkan stabilitas penjelasan LIME pada data tabular (Zafar and Khan, 2019).

Pengujian pertama untuk mengukur skor *fidelity* dilakukan pada 50 data target untuk tiap dataset. Pada hasil evaluasi LIME biasa dan LIME clustering untuk dataset *flower images*, LIME clustering memiliki skor fidelity yang lebih baik daripada LIME biasa dengan peningkatan skor rata-rata sekitar 0,16 dibandingkan hasil LIME biasa. Selain itu, skor fidelity yang dihasilkan tidak berubah secara signifikan ketika diujikan menggunakan jumlah sampel yang berbeda pada metode LIME clustering. Kemudian untuk dataset *intel images*, LIME clustering juga menghasilkan skor *fidelity* yang lebih baik, namun dengan peningkatan skor rata-rata 0,02. Kemudian pada dataset ini, jumlah sampel yang digunakan pada LIME clustering tidak berpengaruh banyak pada skor *fidelity* yang dihasilkan. Terakhir, untuk dataset *box palette*, LIME clustering memiliki hasil skor *fidelity* yang lebih baik dengan peningkatan skor sekitar 0,37 secara rata-rata. Jumlah sampel juga tidak berpengaruh signifikan terhadap skor *fidelity* pada metode LIME clustering. Meskipun pada ketiga dataset jumlah sampel tidak mempengaruhi hasil LIME clustering, jumlah sampel berpengaruh terhadap skor *fidelity* ketika menggunakan metode LIME biasa. Untuk ketiga dataset, semakin kecil jumlah sampel yang digunakan maka skor fidelity akan lebih tinggi ketika menggunakan LIME biasa. Dengan skor fidelity yang lebih baik, maka LIME clustering dapat dikatakan

mampu menjelaskan prediksi model asli mendekati perilaku model asli untuk input yang digunakan.

Kemudian dilakukan pengukuran metrik *stability* untuk mengukur kestabilan penjelasan LIME. Pengujian dilakukan pada tiga data target untuk tiap dataset. Untuk tiap input, dilakukan sepuluh pengulangan dan pada tiap pengulangan data *input* diberi sedikit gangguan. Pada dataset *flower images*, secara keseluruhan LIME biasa menghasilkan skor *stability* yang lebih baik dibandingkan LIME clustering. Kemudian pada dataset *intel images*, didapatkan hasil yang berbeda. Pada dataset ini LIME clustering memiliki skor *stability* yang lebih baik dibandingkan dengan LIME biasa Terakhir, pada dataset *box palette*, LIME biasa menghasilkan *stability* yang lebih baik dibandingkan LIME clustering . Dari hasil yang ada didapatkan bahwa LIME clustering hanya meningkatkan skor *stability* pada dataset *intel images*. Sedangkan pada dataset *flower* dan *palette*, stabilitas LIME clustering stabilitasnya lebih buruk dibandingkan LIME biasa, meskipun skornya masih terhitung rendah atau cukup stabil.

Dari seluruh pengujian yang dilakukan, didapatkan bahwa penggunaan *K-Means* clustering dapat meningkatkan skor *fidelity* pada LIME. Pengukuran skor *fidelity* yang dilakukan melengkapi penelitian terdahulu yang tidak membahas dampak teknik clustering pada LIME terhadap metrik *fidelity*.

Kemudian, pada pengujian skor *stability*, peningkatan skor terjadi pada dataset *intel images.* Hal ini sesuai dengan penelitian terdahulu yang menunjukkan peningkatan stabilitas LIME dengan clustering pada data tabular (Zafar and Khan, 2019). Namun pada dataset *flower images* dan *box palette*, skor *stability* LIME biasa lebih baik dibanding LIME clustering. Pada kedua dataset ini secara keseluruhan skor *stability* dari LIME clustering tidak menurun jauh dari skor *stability* LIME biasa, sehingga dapat dikatakan hasilnya masih cukup stabil. Oleh karena itu hasil eksperimen terhadap stabilitas LIME pada dataset *flower* dan *palette* tidak mendukung penelitian sebelumnya yang menguji penggunaan LIME dengan clustering pada data tabular. Hal ini mungkin disebabkan karena perbedaan bentuk data antara data tabular dan data gambar serta perbedaan pada cara model dan LIME memproses tipe data yang berbeda. Selain itu hal ini juga dapat dipengaruhi oleh ukuran dataset, karena peningkatan stabilitas hanya terjadi pada

dataset *intel images* yang memiliki 25000 gambar. Sedangkan pada dataset *flower* dan *palette* yang tidak mengalami peningkatan, masing-masing memiliki 210 dan 600 gambar. Dengan mengaplikasikan LIME clustering pada data gambar, penelitian ini juga melengkapi penelitian terdahulu yang hanya mengaplikasikan teknik clustering pada data tabular.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian tesis yang telah dilakukan, didapatkan kesimpulan sebagai berikut:

a. Penggunaan model CNN, dengan *transfer learning* dapat memberikan performa yang baik untuk melakukan klasifikasi gambar.
b. Pada dataset *flower images*, *intel images*, dan box palette performa terbaik didapatkan dengan menggunakan model berbasis arsitektur MobileNetV2 dengan skor F1 masing-masing sebesar 88%, 92%. dan 100%.
c. Penggunaan K-Means Clustering pada tahapan LIME dapat meningkatkan performa penjelasan LIME. Hal ini dapat dilihat dari skor fidelity yang secara rata-rata meningkat pada LIME K-Means jika dibandingkan LIME biasa pada semua dataset.
d. Penggunaan K-Means Clustering tidak selalu meningkatkan stabilitas penjelasan LIME. Hal ini dapat dilihat dari hasil evaluasi stabilitas pada dataset *flower images* dan *box palette* yang menunjukkan skor *stability* LIME biasa yang lebih baik dibandingkan skor pada LIME clustering. Namun, LIME clustering menghasilkan skor *stability* yang lebih baik pada dataset *intel images*. Hal ini mungkin disebabkan karena perbedaan ukuran dataset *intel images* dengan dataset *flower* dan *box palette*.
e. Hasil visualisasi LIME K-Means Clustering juga berhasil menyoroti fitur penting ketika diujikan pada seluruh dataset. Namun, pada data box 63 pola salah, hasil penjelasan menyoroti fitur yang kurang penting. Hal ini terjadi pada penjelasan LIME tanpa clustering dan LIME clustering. Terdapat beberapa faktor yang dapat menjadi penyebab. Pertama, faktor dataset. Dataset box palette merupakan dataset yang dikumpulkan secara mandiri dalam waktu yang terbatas, sehingga secara visual, perbedaan antar kelasnya kurang terlihat. Hal ini dapat menyebabkan model CNN yang dilatih kurang bisa mengenali perbedaan dan hanya memiliki performa baik hanya pada dataset latih dan validasi. Oleh karena itu meskipun secara evaluasi model memiliki performa yang baik, bisa jadi model akan memiliki

performa yang kurang baik jika diberikan data di luar data latih dan validasi. Faktor berikutnya adalah lapisan arsitektur yang digunakan pada model mungkin kurang optimal untuk dataset ini.

5.2 Saran

Berikut adalah saran yang dapat diberikan untuk berdasarkan penelitian tesis yang telah dilakukan:

a. Karena keterbatasan penelitian, dataset box palette yang dikumpulkan kualitasnya tidak maksimal sehingga mempengaruhi performa klasifikasi model. Untuk pengembangan selanjutnya, dapat dilakukan pengumpulan dataset yang lebih maksimal dan hati-hati agar kualitas dataset dapat meningkat.
b. Pada penelitian ini, modifikasi LIME hanya menggunakan metode K-Means untuk clustering. Untuk pengembangan berikutnya dapat diujikan metode clustering lain seperti hierarchical clustering atau DBSCAN untuk diintegrasikan dengan LIME.
c. Pada penelitian ini digunakan model ResNet50V2 sebagai model dasar klasifikasi, lalu kemudian diberi layer tambahan dalam proses pelatihannya. Untuk pengembangan berikutnya, susunan layer dapat lebih dioptimalkan agar performa klasifikasi dapat meningkat.

DAFTAR PUSTAKA

Agarap, A.F. (2018) 'Deep Learning using Rectified Linear Units (ReLU)', (1), pp. 2–8. Available at: http://arxiv.org/abs/1803.08375.

Alvarez-Melis, D. and Jaakkola, T.S. (2018) 'Towards robust interpretability with self-explaining neural networks', *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS), pp. 7775–7784.

Bhandare, A. *et al.* (2016) 'Applications of convolutional neural networks', *International Journal of Computer Science and Information Technologies*, 7(5), pp. 2206–2215.

Bhatia, K. *et al.* (2023) 'Integrating explainability into deep learning-based models for white blood cells classification', *Computers and Electrical Engineering*, 110(June), p. 108913. Available at: https://doi.org/10.1016/j.compeleceng.2023.108913.

Ganganath, N., Cheng, C.T. and Tse, C.K. (2014) 'Data clustering with cluster size constraints using a modified k-means algorithm', *Proceedings - 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2014*, (1), pp. 158–161. Available at: https://doi.org/10.1109/CyberC.2014.36.

Gozzi, N. *et al.* (2022) 'XAI for myo-controlled prosthesis: Explaining EMG data for hand gesture classification', *Knowledge-Based Systems*, 240, p. 108053. Available at: https://doi.org/10.1016/j.knosys.2021.108053.

Indraswari, R., Herulambang, W. and Rokhana, R. (2022) 'Deteksi Penyakit Mata Pada Citra Fundus Menggunakan Convolutional Neural Network (CNN)', *Techno.Com*, 21(2), pp. 378–389. Available at: https://doi.org/10.33633/tc.v21i2.6162.

Ishikawa, S. nosuke *et al.* (2023) 'Example-based explainable AI and its application for remote sensing image classification', *International Journal of Applied Earth Observation and Geoinformation*, 118(November 2022), p. 103215. Available at: https://doi.org/10.1016/j.jag.2023.103215.

Islam, M.A. *et al.* (2023) 'An integrated convolutional neural network and sorting algorithm for image classification for efficient flood disaster management',

Decision Analytics Journal, 7(November 2022), p. 100225. Available at: https://doi.org/10.1016/j.dajour.2023.100225.

Khater, T. *et al.* (2023) 'Skin cancer classification using explainable artificial intelligence on pre-extracted image features', *Intelligent Systems with Applications*, 20(September), p. 200275. Available at: https://doi.org/10.1016/j.iswa.2023.200275.

Liu, Y. *et al.* (2023) 'A review of deep leaning in image classification for mineral exploration', *Minerals Engineering*, 204(November 2022), p. 108433. Available at: https://doi.org/10.1016/j.mineng.2023.108433.

Lundberg, S.M. and Lee, S.I. (2017) 'A unified approach to interpreting model predictions', *Advances in Neural Information Processing Systems*, 2017-Decem(Section 2), pp. 4766–4775.

O'Shea, K. and Nash, R. (2015) 'An Introduction to Convolutional Neural Networks', *International Journal for Research in Applied Science and Engineering Technology*, 10(12), pp. 943–947. Available at: https://doi.org/10.22214/ijraset.2022.47789.

Ribeiro, M., Singh, S. and Guestrin, C. (2016) '"Why Should I Trust You?": Explaining the Predictions of Any Classifier', in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 97–101. Available at: https://doi.org/10.18653/v1/N16-3020.

Sevillano-Garcia, I., Luengo, J. and Herrera, F. (2023) 'Optimizing LIME Explanations Using REVEL Metrics', in, pp. 304–313. Available at: https://doi.org/10.1007/978-3-031-40725-3_26.

Shi, S., Du, Y. and Fan, W. (2020) 'An Extension of LIME with Improvement of Interpretability and Fidelity'. Available at: http://arxiv.org/abs/2004.12277.

Velmurugan, M. *et al.* (2021) 'Evaluating Fidelity of Explainable Methods for Predictive Process Analytics', in, pp. 64–72. Available at: https://doi.org/10.1007/978-3-030-79108-7_8.

Visani, G., Bagli, E. and Chesani, F. (2020) 'OptiLIME: Optimized LIME Explanations for Diagnostic Computer Algorithms', *CEUR Workshop*

Proceedings, 2699. Available at: http://arxiv.org/abs/2006.05714.

Yeh, C.-K. *et al.* (2019) 'On the (In)fidelity and Sensitivity for Explanations', *Advances in Neural Information Processing Systems*, 32(NeurIPS). Available at: http://arxiv.org/abs/1901.09392.

Yoon, J., Han, J. and Nguyen, T.P. (2023) 'Logistics box recognition in robotic industrial de-palletising procedure with systematic RGB-D image processing supported by multiple deep learning methods', *Engineering Applications of Artificial Intelligence*, 123(April), p. 106311. Available at: https://doi.org/10.1016/j.engappai.2023.106311.

Zafar, M.R. and Khan, N.M. (2019) 'DLIME: A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems'. Available at: http://arxiv.org/abs/1906.10263.

Zhang, C. (Abigail), Cho, S. and Vasarhelyi, M. (2022) 'Explainable Artificial Intelligence (XAI) in auditing', *International Journal of Accounting Information Systems*, 46(August), p. 100572. Available at: https://doi.org/10.1016/j.accinf.2022.100572.