



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

DILLEMA: Metamorphic Testing for Deep Learning using Diffusion and Large Language Models

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA IN-
FORMATICA

Author: **Muhammad Irfan Mas'udi**

Student ID: 952541

Advisor: Prof. Luciano Baresi

Co-advisors: Davide Yi Xian Hu

Academic Year: 2022-2023

Abstract

In the last decade, deep learning has propelled the development of intelligent systems, notably autonomous vehicles and tumor detection tools. Despite their proven efficacy in real-world scenarios, deep learning applications exhibit unpredictable behaviors in corner cases, raising concerns about potential fatal errors. To address this challenge, effective testing methodologies are important.

This thesis explores the application of Metamorphic Testing in deep learning, emphasizing its efficacy compared to alternative approaches. Unlike traditional software, deep learning systems learn from complex data, making the interpretation of erroneous corner-case behaviors challenging. Metamorphic Testing is proposed as a solution that creates test cases by transforming existing ones with the objective of highlighting failure cases in software and systems. This method has shown great capability to reveal failures in deep learning systems and enhance their robustness.

This thesis introduces DILLEMA (Diffusion Model and Large Language Model for Augmentation) framework. DILLEMA combines Diffusion Models and Large Language Models (LLMs) to perform data augmentation for Metamorphic Testing by providing a more comprehensive approach to generating test cases. The proposed framework aims to address the shortcomings of current methodologies and enhance the reliability of deep learning applications.

Keywords: Metamorphic Testing, Alternative Examples, Data Augmentation, Diffusion Models, Large Language Models, Autonomous Driving

Abstract in lingua italiana

Nell'ultimo decennio, il deep learning ha spinto lo sviluppo di sistemi intelligenti, in particolare veicoli autonomi e strumenti per la rilevazione dei tumori. Nonostante la loro comprovata efficacia in scenari del mondo reale, le applicazioni di deep learning, specialmente nell'ambito della guida autonoma, mostrano comportamenti imprevedibili in casi limite, sollevando preoccupazioni riguardo a possibili errori fatali. Per affrontare questa sfida, sono importanti metodologie di test efficaci.

Questa tesi esplora l'applicazione del Metamorphic Testing nel deep learning, enfatizzando la sua efficacia rispetto ad approcci alternativi. A differenza del software tradizionale, i sistemi di deep learning imparano da dati complessi, rendendo difficile l'interpretazione di comportamenti errati in casi limite. Il Metamorphic Testing è proposto come soluzione per creare casi di test trasformando quelli già esistenti con l'obiettivo di evidenziare casi di errore in software e sistemi. Questo metodo ha dimostrato notevoli capacità nel rivelare errori all'interno di sistemi deep learning e migliorare la loro robustezza.

Questa tesi introduce il framework DILLEMA (Diffusion Model and Large Language Model for Augmentation). DILLEMA combina i Modelli di Diffusione e i Large Language Models (LLMs) per eseguire l'aumento dei dati per il Metamorphic Testing, fornendo un approccio più completo alla generazione di casi di test. Il framework proposto mira a risolvere le limitazioni delle metodologie attuali e migliorare la affidabilità delle applicazioni di apprendimento profondo.

Parole chiave: Metamorphic Testing, Esempi Alternativi, Aumento dei Dati, Modelli di Diffusione, Large Language Models, Auto Autonoma

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Structure	3
2 Related work	5
2.1 Metamorphic Testing	5
2.1.1 DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars	5
2.1.2 DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing	6
2.1.3 DeepXplore: Automated Whitebox Testing of Deep Learning Systems	7
2.2 Alternative Generation and Data Augmentation	8
2.2.1 Dataset Interfaces: Diagnosing Model Failures Using Controllable Alternative Generation	8
2.2.2 Diversify Your Vision Datasets with Automatic Diffusion-Based Augmentation	9
3 Background	11
3.1 Metamorphic Testing	11
3.2 Deep Learning	12
3.2.1 Feed Forward Neural Network	13
3.2.2 Convolutional Neural Network	16
3.2.3 Classification and Semantic Segmentation	18
3.2.4 Transformer and Large Language Model	22

3.2.5	Diffusion Model	24
3.3	Metamorphic Testing in Deep Learning	27
4	Solution	29
4.1	Describing the Image with Captioning Model	30
4.2	Large Language Model (LLM) Application	31
4.2.1	LLM: Generate Keywords	32
4.2.2	LLM: Generate Alternatives	32
4.2.3	LLM: Generate New Caption	33
4.3	Controlling Image Generation in Diffusion Model	33
5	Implementation	37
5.1	BLIP-2 Captioning Model	37
5.2	LLaMA-2 as Large Language Model	37
5.3	ControlNet for Image Generation	39
6	Evaluation	43
6.1	Experimental Description	43
6.2	Experimental Setup	44
6.3	Result	45
6.3.1	ImageNet1K Result	53
6.3.2	ResNet18 Performance	54
6.3.3	ResNet50 Performance	66
6.3.4	ResNet152 Performance	78
6.3.5	Fine-tune Pre-trained Model	89
6.3.6	SHIFT Result	90
7	Conclusion and Future Works	93
7.1	Conclusion	93
7.2	Future Works	93
	Bibliography	95
	List of Figures	105
	List of Tables	107

1 | Introduction

Over the past decade, remarkable advancements in deep learning have facilitated the creation of safety-critical Intelligence systems like autonomous vehicles [2, 17, 30] and medical tool for tumor detection. Prominent automobile companies, such as Tesla have been engaged in constructing and rigorously assessing these self-driving cars [45]. Recently, autonomous vehicles have proven to be highly effective in real-world scenarios, having covered millions of miles without the need for human intervention [21]. In the case of tumor detection, *Adel et. al.* highlight the importance of early tumor detection in increasing the probability of survival rate [13]

Nevertheless, despite the significant advancements, deep learning, including those applied in safety-critical systems, can be similar to conventional software, it has erroneous or unexpected behaviors in corner cases, which could potentially result in dangerous outcomes such as a fatal collision. Therefore, the testing process becomes a crucial aspect. From a conceptual perspective, the faulty corner-case behaviors in deep learning can be semantically equivalent to software erroneous. But the difference with traditional software, deep learning learn from data that are trained to approximate complex functions that can be difficult to interpret or explain [65]. Various techniques are proposed for testing in deep learning [37, 64], including Metamorphic Testing [71], Combinational Testing [26, 40], Mutation Testing [22, 43], and Fuzzing Testing [15, 16]. Liu *et al.* Metamorphic Testing stands out as a notably effective method when compared to alternative approaches [34].

The conventional method for assessing deep learning involves the collection and manual labeling of a substantial amount of real-world test data. In computer vision applications of self-driving cars, some deep learning systems also employ simulation to create synthetic training data. Nevertheless, this simulated data lacks guidance and does not take into account the intricacies of the target system. Consequently, when dealing with the vast input spaces of real-world scenarios, such as encompassing all potential road conditions for self-driving cars, most of these methods can reasonably expect to have corner cases [45].

Metamorphic Testing can be applied to deep learning, by creating a set of test cases

by applying transformations to the original input data [12]. Then evaluate the system by comparing it with the expected outputs [9, 69]. Furthermore, adding error-inducing inputs to the training datasets can help to improve the reliability and accuracy of existing deep learning models, especially in autonomous driving car [72].

Generation test cases or data augmentation for Metamorphic Testing could use traditional transformation [62], including adjustments to brightness and contrast, translation, scaling, horizontal shearing, rotation, and blurring. However, this approach failed to capture several driving scenarios that were artificially generated to simulate various conditions. In particular, real-world driving environments are seldom amenable to affine transformations and the limitations of autonomous driving system perception. *Generative Adversarial Networks* (GANs)-based test case generation could have more varied scenarios and conditions, e.g generate the snowy environment from the sunny environment. GANs needs training for each precise transformation within a targeted domain, which makes GANs rely on paired data to learn the mapping from one domain to another [4, 27]. In the driving scenario, there could be a lot of possibilities target domain. Hence, it is not easy to prepare even from the initial implementation [70]. Generate test cases could also use pre-trained models to generate new test cases with additional perturbation. However, this approach still falls short in capturing the possibilities in corner cases, as it may not be comprehensive enough to encompass various conditions and environments.

Generating data augmentation could deploy controllable alternative to generate distribution shift of datasets. Alternative generation acquires augmented data that stays in class or label data training distribution with specific changes from text description of images. Image description is a natural language that describes the image. Recent work, *Large Language Model* (LLMs) used to tailor the image descriptions in order to make a new description for data augmentation based on the original data. The original data image is described as text and it is used to generate new images with different domains and conditions that are specified with text. The existing solution for image generation with the original image as its reference and text description managed by LLMs combined with GANs-based cannot capture the spatial context in the image. Therefore, this thesis propose *DILLEMA* (Diffusion Model and Large Language Model for Augmentation) to perform alternative generation and data augmentation as the generated test cases for Metamorphic Testing.

1.1. Structure

The thesis structure is organized as follows:

- Chapter 1 Introduction, provide the general overview of the thesis.
- Chapter 2 Related work, this chapter will provide the recent research related to thesis work. Exploring different techniques that have been implemented dealing with Metamorphic Testing on autonomous driving car, and related to alternative generation and data augmentation.
- Chapter 3 Background, introduces some concepts, technologies and theoretical models that are used in the context of this thesis.
- Chapter 4 Solution, this chapter will provide a description of *DILLEMA* (Diffusion Model and Large Language Model for Augmentation) to perform alternative generation and data augmentation as the generated test cases for Metamorphic Testing.
- Chapter 5 Implementation, this chapter will provide the technicalities of the implementation of *DILLEMA*. Proposed implementation of Image Captioning Model, Large Language Model, and Diffusion model.
- Chapter 6 Evaluation, In this chapter, a crucial element is presented as the assessment and validation of *DILLEMA* are detailed. To contextualize this evaluation, thorough explanations are provided regarding the methodologies and approaches used, covering tools, techniques, and datasets.
- Chapter 7, Conclusion and Future Works, This chapter is dedicated to present the conclusive remarks based on the outcomes of our study and delving into promising directions for future research.

2 | Related work

This chapter provides the recent research related to thesis work. Exploring different techniques that have been implemented dealing with Metamorphic Testing on deep learning application, and related to alternative generation and data augmentation.

2.1. Metamorphic Testing

2.1.1. DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars

The main goal of this research is specifically designed for the automated testing of autonomous vehicles. These vehicles rely on deep neural networks (DNNs) for decision-making and control. Autonomous vehicles must undergo rigorous testing to ensure their safety and reliability. This includes assessing how well they respond to various scenarios and conditions, as well as verifying that the DNNs make appropriate decisions. Automated testing involves using software tools and scripts to systematically evaluate the performance of autonomous vehicles. Their method implemented Systematic Testing with Neuron Coverage, which tries to generate inputs that maximize neuron coverage of the test DNN. As each neuron's output affects the final output of a DNN, maximizing neuron coverage also increases output diversity. They tried increasing Neuron Coverage with Synthetic Images by applying a set of image transformations to original images, they aim to replicate various real-world phenomena such as distortions caused by camera lenses, object motion, diverse weather conditions, and more. In pursuit of this objective, they explore nine distinct and authentic image transformations, including adjustments to brightness and contrast, translation, scaling, horizontal shearing, rotation, and blurring, as well as the addition of fog and rain effects. These transformations can be categorized into three groups: linear, affine, and convolutional. This kind of transformation shows it can uncover the possible erroneous in the steering decision, see Figure 2.1.



Figure 2.1: *Blue* arrow is the expected steering in original images and red arrow is the steering result with six-degree rotation [55].

Nevertheless, the techniques employed in DeepTest for generating test cases may not faithfully replicate actual real-world driving scenarios. In particular, real-world driving environments are seldom amenable to affine transformations and the limitations of autonomous driving system cameras. Additionally, the simulated blurring, fog, and rain effects can often appear unrealistic, thus potentially undermining the effectiveness and dependability of DeepTest.

2.1.2. DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing

DeepRoad has created a Metamorphic Testing component designed for DNN-driven autonomous systems. In this approach, Metamorphic Relations are established to ensure that, regardless of how driving scenarios are artificially generated to simulate various weather conditions, the driving behavior should remain consistent with that observed in the corresponding genuine driving scenarios. This development assesses the precision and dependability of current DNN-based autonomous driving systems (DeepTest), for example when faced with diverse extreme weather scenarios, such as heavy snow and heavy rain. The generation can hardly be distinguished from the original image and cannot be generated using simple transformations. DeepRoad is used to simulate various weather conditions, based on the *Generative Adversarial Networks* (GANs) technique. Generating various real-world road scenes fully automatically, they implemented UNIT, the GAN-based method to perform unsupervised image-to-image transformation. The different input domains can be projected into a shared latent space and have the same latent representation. In this way, given a new image from one domain image, UNIT can automatically generate its corresponding version in the other domain. See Figure 2.3

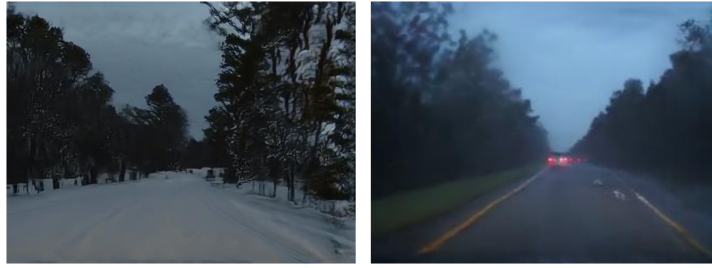


Figure 2.2: GAN based generated images, snowy and rainy image [67].

The main issue with this solution is that they require separate GAN training for each specific transformation within a given domain. This process can be costly and time-consuming. Furthermore, these methods assume the availability of a set of images from the target domain. This assumption may not always hold true, as collecting such images can be challenging, especially in scenarios where neural networks must operate in several domains.

2.1.3. DeepXplore: Automated Whitebox Testing of Deep Learning Systems

DeepXplore was developed to automatically test the robustness of deep learning models by generating test inputs that can show previously unobserved behavior in the model. The idea is to generate test cases that are similar to the data train based on the neuron converge parameters. First, by taking the unlabeled test inputs as seeds and generating new tests that cover a large number of neurons. Note that, the pre-trained model is used to do the generating process. Instead of having gradient descent to optimize the weights of NN, they use gradient ascent to compute the input value that maximizes the outputs.



Figure 2.3: Left image is the original input image that correctly makes a decision result. The right image is the generated image which makes an error with respect to the expected result [45].

DeepXplore focuses on examining a limited set of transformations to evaluate their associated characteristics. While these transformations may be considered more realistic compared to adversarial perturbations, they still fall short of encompassing the complete spectrum of real-world input distortions.

2.2. Alternative Generation and Data Augmentation

2.2.1. Dataset Interfaces: Diagnosing Model Failures Using Controllable Alternative Generation

Dataset Interface deployed alternative examples to generate distribution shift of the image datasets. To perform reliability of real-world input of the model represented in the training dataset. The machine learning model could fail when faced with a distribution shift, for instance, changing the background of the image input. In order to improve the performance, they applied alternative generation, by acquiring new images that stay in class or label data training distribution with specific changes. Dataset Interfaces use synthetically generated data with textual inversion. Given a set of image inputs, the text inversion will find a word or token in the Diffusion Model's text space. Textual inversion will learn embedded vectors in the text embedding space of the Diffusion Model. Therefore, the token will not be diverse with the expected label. Then it can be used to make distribution shift of images. For example a $\langle \text{dog} - \text{class} \rangle$ in the "A photo of $\langle \text{dog} - \text{class} \rangle$ in the beach" and in "A photo of $\langle \text{dog} - \text{class} \rangle$ in the grass" is the same, for instance, the dog breed and its color. The only change is the background of the image. See Figure 2.4



Figure 2.4: Left image is "A photo of $\langle \text{dog} - \text{class} \rangle$ in the beach". The right image is "A photo of $\langle \text{dog} - \text{class} \rangle$ in the grass" [60].

2.2.2. Diversify Your Vision Datasets with Automatic Diffusion-Based Augmentation

Classification task in some cases have limited training data. Because of that trained model classifiers on these datasets come to fail when faced with the variation of domain changes. Automated Language-guided Image Augmentation (ALIA) is a method to automatically generate new data related to the original input data. This method uses image captioning and a language model to generate a natural language description of the image. Then the description will be used for image augmentation. In order to verify the augmented data, ALIA employed filtering which will remove the augmented image if it does not change the domain or corrupt task-relevant information. Generating caption from the image using BLIP Captioning Model, then tailoring the expected result with GPT 4 Large Language Model (LLM). Editing images with language guidance deployed two techniques, *Image to Image with Text Guidance (Img2Img)* and *Instruct Pix2Pix*. The generated images based on tailored descriptions might fail. Hence, the implementation of *Semantic Filtering* as a classification task from given input text and images to get the correct target class of generated images by using CLIP. *Confidence-based Filtering*, comparing the original datasets' confidence interval with augmented data. For the comparison between the original image and the augmented result, see Figure 2.5.



Figure 2.5: ALIA augmented image result and the original image [11]

3 | Background

This chapter introduces some concepts, technologies and theoretical models that are used in the context of this thesis.

3.1. Metamorphic Testing

In software engineering, software testing is to examine the correctness of a piece of software. It is a dynamic technique with an experimental approach for verification and validation processes [35, 52]. *Verification* is an assessment of whether the software aligns with the specified requirement. Instead, *Validation* is to check the system with the expected software needs [8]. The testing process needs a test case, it is error-revealing if it detects a software error otherwise it is called successful. In detail, test case generation can be selected *Randomly*, it is blind testing that generates and executes hundreds of thousands of test cases. And *Systematically*, selecting test cases using characteristics of the software artifacts (e.g. code), and using the information on the behavior of the system (e.g., specification) [51]. The Figure 3.1 depict the testing process in the software engineering

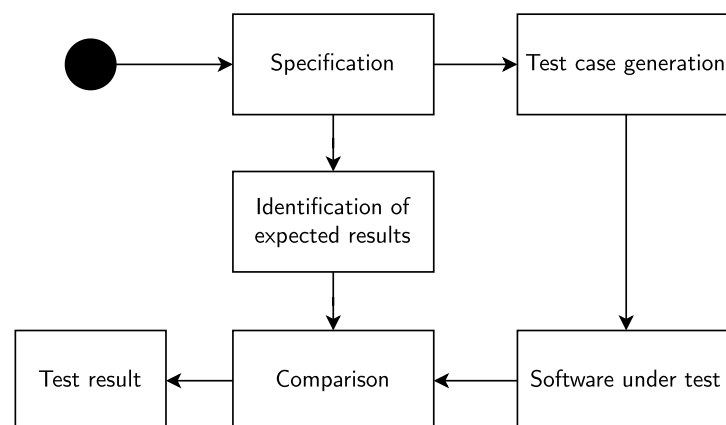


Figure 3.1: Testing process.

The result of a testing process is comparing the output of the software under test, for a given generated test case input, to the output that determines that software should have,

then concluded whether a test has passed or failed, it is known as *test oracle*. Test case generation is constructed based on some predefined criterion and generally assumes the availability of a test oracle. However, as indicated by Weyuker [8] this assumption may not hold in practice. *Metamorphic Testing* rise because of the difficulties in constructing a perfect test case that does not assume the availability of such a test oracle. This technique refers to the generation of new test cases based on the input-output pairs of previous test cases and the types of errors usually associated with that particular type of application. The approach is considered fault-based as new test cases aim at uncovering specific errors which left undetected in previous successful test cases [7].

3.2. Deep Learning

In recent years, Artificial Intelligence has experienced extraordinary developments, which linearly make it hard to give one exact definition because it can be defined in many ways. Artificial Intelligence is a big umbrella that covers Machine Learning, and conceptually, Deep Learning lies inside.

Starting from Machine Learning definition, according to Tom Mitchell, Machine Learning is a computer program that learns from experience with respect to some class of task and has a measurable performance, if the performance of the task improves, it is because of the experience [38]. Based on the learning paradigm, Machine Learning can be divided into some categories. This thesis will mainly focus on *Supervised Learning*, it is a learning paradigm when there are available data $input = \{x_1, x_2, x_3, \dots, x_N\}$ and data $target = \{y_1, y_2, y_3, \dots, y_N\}$. The task is to predict the new input t_k based on new data input x_k . *Classification* is a Supervised Learning example, the task that asked to specify which k categories when given data inputs. The learning process will learn the function $f : \mathbb{R}^n \rightarrow \{1 \dots k\}$. When $y = f(x)$, let x as the inputs then the model will provide the categorical result of class k [18]. Another task that is related to supervised learning is *Regression*, in this task, is asked to predict a numerical value given some input. To solve this task, the algorithm should learn the function of $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

The difference between Machine Learning and Deep Learning is the way to train the model from the data based on features. *Feature* is the characteristic or measurable property to describe the data. In Machine Learning, extracting the features is handcrafted by humans as a feature engineering process. Instead, in Deep Learning, feature extraction will be done in the training process, asking the machine to make the best representation of the data. Hence, the term Deep in Deep Learning comes from the fact that the learning process is pushed down in the hierarchy between data and output through the layers.

Deep learning is about learning data representation from the data, and it needs a huge amount of data.

3.2.1. Feed Forward Neural Network

Artificial Neural Network is inspired by the biological system of the human brain which contains layers of neurons connected. Each neuron receives inputs, processes them, and produces an output [61]. The perceptron is a mathematical model of the biological neuron shown in Figure 3.2 that contains input x_i , weight w_i , bias b , and h_j is the output.

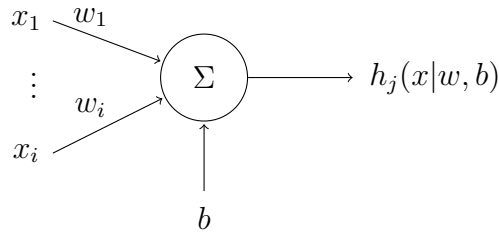


Figure 3.2: The perceptron.

The perceptron has weight w_i and bias b as the learnable parameters to get better performance on the specific task from the input x_i . If $b = w_0$ then the output h_j is the result of the activation function of $w^T x$. The formula shown in Equation (3.1)

$$h_j(x|w, b) = h_j \left(\sum_{i=1}^I w_i \cdot x_i + b \right) = h_j \left(\sum_{i=0}^I w_i \cdot x_i \right) = h_j(w^T x) \quad (3.1)$$

Feed forward neural network is the basic concept of Deep Neural Networks (DNNs) a non-linear model constructed from a multilayer perceptron with several layers (input layer, hidden layer, and output layer) shown in Figure 3.3. By stacking more hidden layers the network will tackle the non-linear problem. The training process is to find the optimal value of learnable parameters. Since the learning process works in the non-linear problem, it cannot have convex functions, which means the closed-form solutions are practically never available. Therefore, the learning process is in the scope of non-linear optimization. By iterative solution using *Gradient-descent* also known as *Backpropagation*, the learning process will update the trainable parameters based on the error of the output with respect to the target. The parameters should minimize the error distance between the output and the target. In another way, the output of the neural network is $g(x_n|w) \sim t_n$.

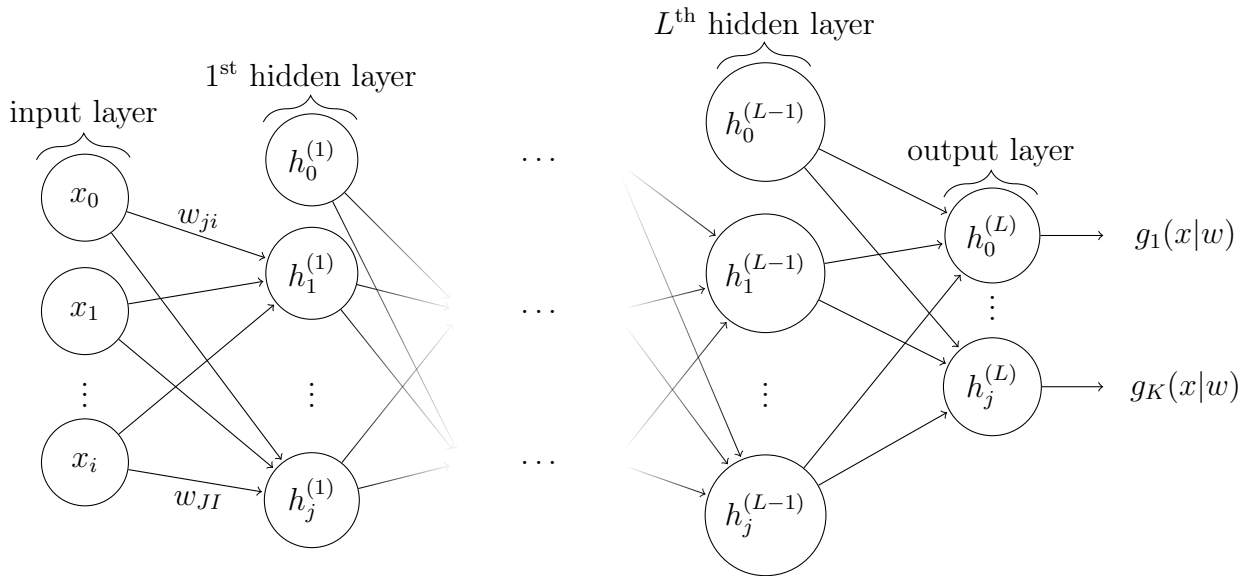
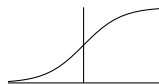
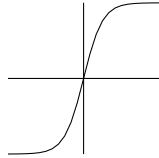
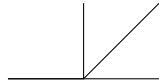


Figure 3.3: Feed Forward Neural Network for a (L) -hidden layer with i input units and K output units. w_{ji} is the weight going to neuron j -th from input or neuron i -th.

Gradient descent only works on differentiable activation functions. There are several non-linear activation functions [6, 10, 39, 41], in Table 3.1 are the activation functions that commonly used. The choice of activation functions depends on the task. For instance, in the classification problem for multiclass, the output of the network should use Softmax activation function in order to normalize the output in the range 0 to 1. In this case, the number of output layer of the network should be the same of the number of K class.

Table 3.1: Non-linear activation functions.

Name	Function	Derivative	Figure
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))^2$	
tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	
ReLU	$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0. \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0. \end{cases}$	
Softmax	$f(x) = \frac{e^x}{\sum_i e^x}$	$f'(x) = \frac{e^x}{\sum_i e^x} - \frac{(e^x)^2}{(\sum_i e^x)^2}$	

Finding the weight of a neural network are non-linear optimization problem. The optimal weight will provide the minimum error. For each task have specific *error function* or *cost function*. The regression problem uses sum squared error as a cost function see the Equation 3.2. The classification task uses categorical cross-entropy, see the Equation 3.3

$$E(w) = \sum_{n=1}^N (t_n - g(x_n|w))^2 \quad (3.2)$$

$$E(w) = - \sum_{n=1}^N t^T \log(g(x_n|w)) \quad (3.3)$$

Starting the iteration from the initial random configuration, Gradient descent will update the weight iteratively through the Stochastic Gradient Descent (SGD) formula [3], see the Formula 3.4.

$$w^{k+1} = w^k - \eta \frac{\partial E(w)}{\partial w} \Big|_{w^k} \quad (3.4)$$

Where $\frac{\partial E(w)}{\partial w} = \frac{\partial E(x_n, w)}{\partial w}$, each iteration updated for each data points. w^{k+1} is the weight at $k + 1$ -th iteration as updated weight, w^k is the weight at k -th iteration, η is the learning rate, E is the mentioned cost function, and $\frac{\partial E}{\partial w}$ is the gradient of the cost function with respect to the weight. Several variants of optimization also exist such as SGD with momentum, Rprop, AdaGrad, RMSprop, AdaDelta, and Adam [23, 47, 58] which try to improve the basic gradient descent method both in the performance of convergence and with respect to the time to convergence. Partial derivative cost function with respect to the weight could be done in two steps; *Forward pass* and then *Backward pass*. Forward pass will produce the result of output the neural network given the input, $g(x_n|w)$. The backward pass is the process that allows the information of the cost function results to propagate backward in the network to compute its gradient and then update the weight. The gradient of the cost function with respect to weight can be computed by using the chain rule for partial derivatives in an efficient way with parallel computation. Instead of having the updated weight for each iteration with each data. *Batch Gradient Descent*, uses all the available data training that is used at once and the gradient is averaged. For one epoch (when all training data is already used) there will be only one iteration, see the Equation 3.5, where N is the number of data train.

$$\frac{\partial E(w)}{\partial w} = \frac{1}{N} \sum_{n=1}^N \frac{\partial E(x_n, w)}{\partial w} \quad (3.5)$$

In practice batch gradient descent might not be possible due to the limitation of computer resources, because it is going to load N all available data train once [24]. Hence, *Mini-batch gradient descent* helps to slice N into M sections. Therefore, in one epoch there will be M iterations, in this case loading the N/M per iteration is easier to handle by computer, see the Equation 3.6

$$\frac{\partial E(w)}{\partial w} = \frac{1}{M} \sum_{n \in \text{Minibatch}}^{M < N} \frac{\partial E(x_n, w)}{\partial w} \quad (3.6)$$

In the training process, the gradient may lead to *vanishing gradient*. The condition that the result of the gradient is nearly zero or too small [25, 44]. ReLU (Rectifier Linear Unit) is used as an activation function because Sigmoid and Tanh are prone to vanishing gradient when the layer is going deeper [20]. A vanishing gradient problem could also happen if the initial weight is started with a very small value [53]. Therefore, a proper weight initialization is needed. Such as *Xavier Initialization* propose weight initialization $w \sim \mathcal{N}(0, \frac{1}{n_{in}})$ where n_{in} number of neuron inputs and *Gloroth Initialization* also propose $w \sim \mathcal{N}(0, \frac{2}{n_{in} + n_{out}})$ where n_{out} is the number of output layers [14].

3.2.2. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a widely used deep learning architecture, particularly in computer vision. CNN is constructed by several types of layers, convolutional layer, sub-sampling or pooling, and fully connected layer [68]. See Figure 3.4, which is the architecture for the image classification task. A convolutional layer is a layer where convolution operations are executed. When working on the data image, the input should have height, width, and channel (e.g. RGB) [29].

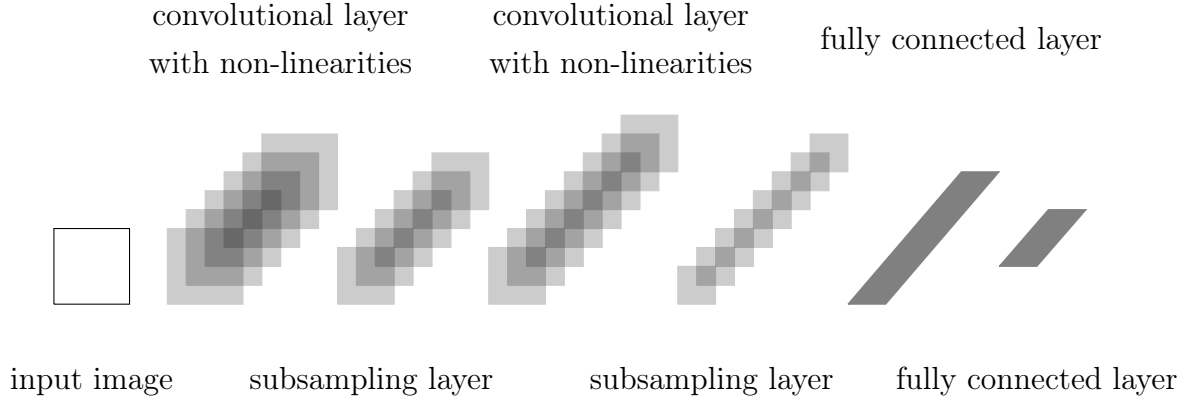


Figure 3.4: The architecture of the original convolutional neural network, as introduced by *LeCun et al.* [28].

A convolution operation is defined as follows: Let $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ is an input and $\mathbf{V} = \{\mathbf{v}_i \in \mathbb{R}^{h \times w \times C} \mid i \in I\}$ is a filter or kernel. The filter could have an arbitrary number of I and the important thing is the depth of the filter should same as the input depth. The convolution operation of \mathbf{v}_i over \mathbf{X} can be written the Equation 3.7.

$$\mathbf{u}_i = \mathbf{v}_i * \mathbf{X} = \sum_{s=1}^I \mathbf{v}_i^s * \mathbf{x}^s \quad (3.7)$$

where \mathbf{u}_i is the result of convolution, \mathbf{v}_i^s is the i -th filter of \mathbf{V} operating on the s -th channel of \mathbf{X} , and \mathbf{x}^s is the s -th channel of \mathbf{X} and F is the number of the filter. The Equation 3.8 is the detail formula.

$$\mathbf{v}_i^s * \mathbf{x}^s(r, c) = \sum_{\substack{u, v \in \mathbf{v}_i^s \\ r, c \in \mathbf{x}^s}} \mathbf{v}_i^s(u, v) * \mathbf{x}^s(r - u, c - v) \quad (3.8)$$

where (r, c) is the position pixel in \mathbf{x}^s , (u, v) is the pixel's position in filter \mathbf{v}_i^s . The output will be $\mathbf{U} \in \mathbb{R}^{H' \times W' \times I}$ that can be represented as sliced layer $\mathbf{u}_i \in I$, which $[\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_I]$ whose I is the depth that will be equal to the number of the filter \mathbf{V} applied to the input. The spatial size $H \times W$ changes to $H' \times W'$ due to the different settings values of stride (how many steps the filter window will move), kernel size (filter height and width), and the padding type (how the filter and input pixel combination to get the intended result). The filter is a learnable parameter, it is similar to neuron, $\mathbf{u}_i = \sum_j^J x_j \cdot w_{kj}^{(i)} + b^{(i)}$. The result of the convolutional layer should be added with activation functions in order to give non-linearities.

The receptive field is a portion of the input image that a particular neuron in the network is sensitive to. It describes the spatial extent of the input data that influences a particular feature or unit in the network. The receptive field can vary in size and shape depending on the depth of the layer and the architecture of the CNN. The receptive field of a neuron in a CNN is determined by the size of the convolutional filter and the stride used in each layer. When performing a convolution operation, a filter slides over the input image, and the receptive field of a neuron in a given layer is the area of the input image that the filter covers. In the initial layers of the CNN, the receptive field of neurons is relatively small. Each neuron responds to a small region of the input image. When moving deeper into the network, the receptive field of neurons increases. This is because multiple convolutional layers process information from previous layers, causing the receptive field to grow. Neurons in deeper layers are sensitive to larger and more abstract features. The pooling layers, such as max-pooling or average-pooling, can further increase the receptive field. Pooling layers downsample the feature maps, making the spatial dimensions smaller while retaining the most important information.

Subsampling layer or pooling layer is used for reducing the spatial size of the convolutional layer output. Because it is going to help in terms of computation by reducing the parameters but not missing the important things. Hence, there is an existing pooling technique; *Max-pooling*, which will take the maximum value as the result in the neighborhood. Instead, *Average-pooling* will average the neighborhood as the result. The result of the convolutional layer and subsampling is a *Latent representation* and the process is usually called *High-level feature extraction*. Then the last layer of CNN architecture is a fully connected layer, conceptually it is a feed-forward neural network. The latent representation will feed as the input of the network by *Flattening* process, change the dimension $\mathbb{R}^{H' \times W' \times I} \rightarrow \mathbb{R}^n$ and the output will be used to solve the image classification task, the input is $\mathbf{X} \in \mathbb{R}^{H \times W \times C} \rightarrow k$, where k is predicted class. In some cases, instead of having flattening, it can be replaced with *Global-Average Pooling* by averaging each $\mathbf{u}_i \in \mathbf{U}$. This technique can reduce the trainable parameters [33].

3.2.3. Classification and Semantic Segmentation

In the computer vision context, there are many tasks that can be applied in deep learning. First task is image classification, given an image as an input to a trained network, then the network has to determine the class of the input image. Classification task uses categorical cross-entropy as the loss function. In order to measure how good the performance, there are many metrics to calculate the performance of the model. *Accuracy* provides an overall view of correct predictions see Equation 3.9, the accuracy formula for multiclass. But it

may fall short in scenarios where class imbalances or specific objectives demand a more nuanced evaluation. Precision, recall, and F1 score are three metrics that offer a more comprehensive understanding of a model's performance.

$$\text{Accuracy} = \frac{\sum_{k=1}^K \text{True Positives}_k}{\text{Total Observations}} \quad (3.9)$$

Precision, also known as positive predictive value, calculates the ratio of correctly predicted positive observations to the total predicted positives. It is particularly valuable in scenarios where minimizing false positives is crucial, as it provides insight into the model's ability to avoid misclassifying negative instances as positive. *Macro* refers to a type of averaging method used to compute aggregate metrics across multiple classes. In the case of precise, calculate precision for each class individually and then average them see Equation 3.10. Macro averaging method is used also in the recall and f1 score metrics.

$$\text{Precision}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{\text{True Positives}_k}{\text{True Positives}_k + \text{False Positives}_k} \quad (3.10)$$

Recall, or sensitivity, measures the proportion of actual positive instances that are correctly identified by the model. This metric is essential when the cost of missing positive cases is high, as it assesses the model's capacity to capture all relevant instances of the positive class see Equation 3.11

$$\text{Recall}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{\text{True Positives}_k}{\text{True Positives}_k + \text{False Negatives}_k} \quad (3.11)$$

F1 score is the harmonic mean of precision and recall, offering a balanced assessment that considers both false positives and false negatives. It is especially useful when there is an uneven distribution between classes, providing a single metric that considers both precision and recall see Equation 3.12.

$$\text{F1 Score}_{\text{macro}} = \frac{2}{K} \sum_{k=1}^K \frac{\text{Precision}_{\text{macro}} \times \text{Recall}_{\text{macro}}}{\text{Precision}_{\text{macro}} + \text{Recall}_{\text{macro}}} \quad (3.12)$$

There is other task in the computer vision, *Classification + Localization*, the goal of this task is not only to classify objects but also to provide a bounding box that indicates the object's location within the image. In addition to determining whether an image contains a cat or a dog, the model will also draw a box around the detected object. Another task

is *Object Detection*, which takes the classification + localization a step further by allowing the model to detect multiple objects within an image. It can identify and locate several objects, drawing bounding boxes around each one. For example, in an image with both a cat and a dog, an object detection model would locate and label both. While classification + localization only draws the box, *Semantic segmentation* tries to draw the boundary (or mask) between each class (e.g. cat, dog, and carpet), see Figure 3.5

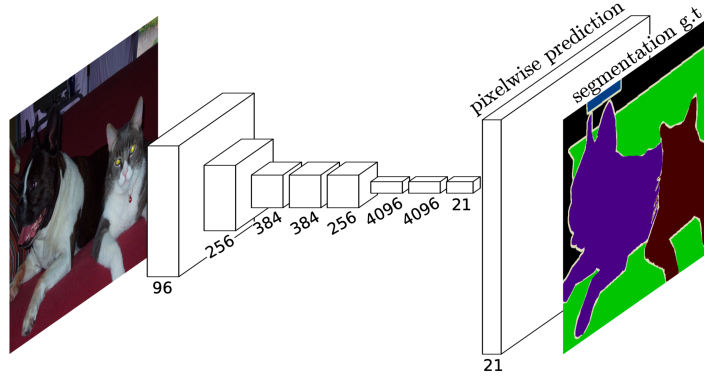


Figure 3.5: Semantic segmentation with Fully-Convolutional Network [36].

Image segmentation problem assigns to each pixel of image $I \in \mathbb{R}^{R \times C \times 3}$ with a label l_i from fixed categories of $L = \{cat, \dots, dog\}$. The result is $S(x, y) \in L$ denotes the class that is associated with the pixel (x, y) , $I \rightarrow S \in L^{R \times C}$. Conceptually semantic segmentation task uses the convolutional operation and down-sampling (Max-pooling, Average-pooling) to extract the feature to get latent representation, which is usually called *Encoder*. The next part is *Decoder*, doing the opposite way of the encoder, instead of making deep of the layer, it will reduce the depth of the result for each convolution and at the same time do up-sampling (e.g. Max-unpooling, Transpose convolution).

Long et al. (2015) also propose the *skip connection*, by adding the result of down-sampling layer into the result of up-sampling layer which has the same spatial space. In the sense that the latent representation contains semantic information about what the class belongs to, while the spatial information knows the position that belongs to that class in the image. See Figure 3.6

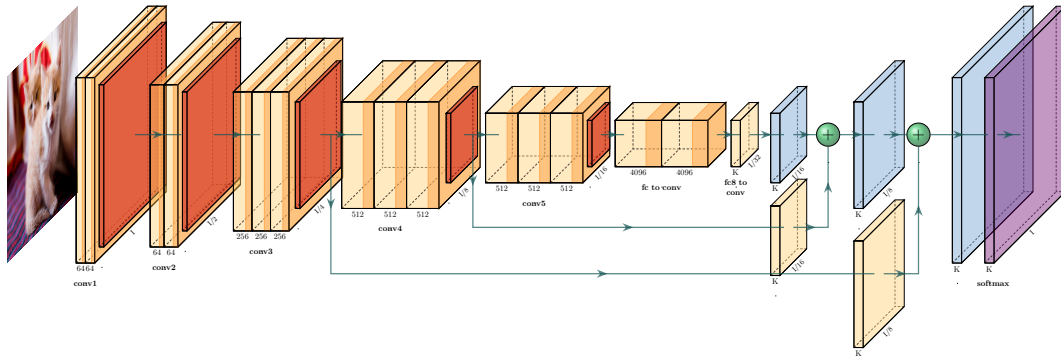


Figure 3.6: Fully Convolutional Network with 8 times upsampled.

Another architecture that has improved the segmentation result is *U-Net*, instead of only having the skipping connection. U-Net implements a stacked result of convolution in the encoder part with convolution in the decoder part. Therefore, it is not only going deeper to get the latent representation but also keeping the information in the encoder part to be used in the decoder. See Figure 3.7

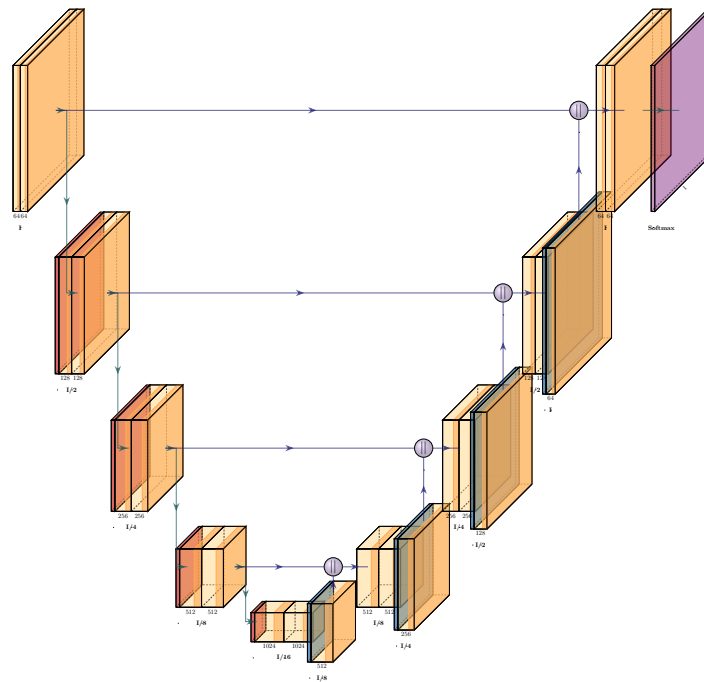


Figure 3.7: U-Net Architecture, the orange layer is convolution, the red color is max-pooling, and the blue color is convolutional transpose.

In image segmentation, a metric that is usually used is Intersection over Union (IoU), also known as the Jaccard index. It measures the similarity between the predicted segmen-

tation mask and the ground truth (or reference) segmentation mask. IoU is particularly useful when dealing with binary or multi-class image segmentation tasks. The IoU is calculated as the ratio of the intersection of the predicted and ground truth regions to the union of these regions. The formula for calculating IoU is as Equation 3.13

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (3.13)$$

where A is predicted and B is ground truth, *Intersection* The area where the predicted segmentation mask and the ground truth mask overlap or intersect. *Union* The total area covered by both the predicted and ground truth masks. IoU values range from 0 to 1, with 0 indicating no overlap between the predicted and ground truth masks (completely dissimilar), and 1 indicating perfect overlap (perfectly similar). The higher the IoU, the better the segmentation accuracy.

3.2.4. Transformer and Large Language Model

The Transformer is a deep learning architecture model that widely used for Seq2Seq problem. *Seq2Seq* or Sequence to Sequence, is the concept for learning the data not an independent data point. Instead, they related to each other. For instance the input $\mathbf{x} = \{x_i, x_{i-1}, \dots, x_{i-z}\}$, and also for the output $\mathbf{y} = \{y_i, y_{i-1}, \dots, y_{i-z}\}$. The example of Seq2Seq problem is a machine translation, which makes a translation from an input language into another language. The input is a sequence of words that have semantic, then the output is a sequence of words that have similar semantic to the sequence input language. In this case it is a sequence of inputs and the sequence of output. Another example of Seq2Seq is *Image Captioning*, the input is single image, but the result is a sequence of words that describe the semantic of the image. Seq2Seq rely on the Encoder-Decoder architecture, in the sense of finding feature extraction from the sequence of input as encoder process. Then Encoder part will generate the feature of latent representation into another domain. Vaswani et al. (2017) proposed Transformer to improve performance in term of quality and training process parallelization, compared to other approaches, Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), and Gated Recurrent Unit (GRU). The Transformer architecture combines some functionalities that are stacked into Decoder and Encoder.

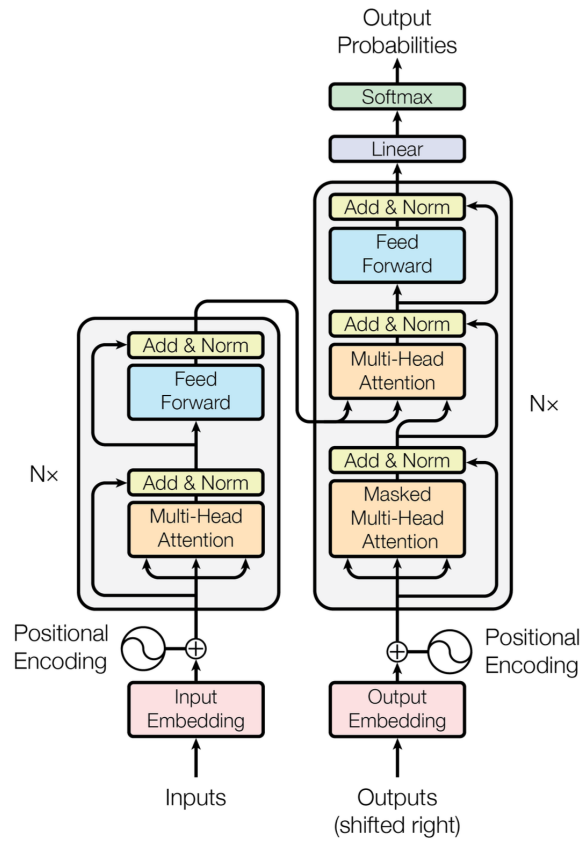


Figure 3.8: The Transformer Encoder-Decoder architecture [59].

The Figure 3.8 is the transformer architecture that inside of the Encoder-Decoder part there exist some components:

- *Input Embedding*, transformer originally used for the input of a sequence of words, that is usually called as token. It is use neural network to get trainable weight for predicting the next token as the input embedding. This is used for get the semantic of each word and give the similarity value to other words in the list of the input tokens.
- *Positional Encoding*, the idea is to give the input to keep track of the sequence of tokens order. Because word or phrase order is important in the semantic or meaning of the sentences.
- *Multi-Head Attention*, in order to know the similarity of each token including itself. The transformer using the Attention. The idea is each token have value of Query, Key, and Value. Query related to the value of similarity of tokens itself. And the Key is related to the similarity to other tokens and Value is to make sure that the token is exactly the input token. Attention related to the token itself is a Self-

Attention. Then the result is a Scaled Dot Product Attention, see the Equation 3.14.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (3.14)$$

where $\sqrt{d_k}$ is Keys dimensions. The Multi-Head Attention is stacking Self-Attention in order to handle a more complex sequence of tokens. The attention mechanism also used in the connection between Encoder and Decoder because the Decoder needs to keep track with the inputs.

- Feed Forward Neural Network is added to have more nonlinearity with weight and bias to fit with more complicated data.

In the training process, each token is independent, which means they are not dependent on each other when the training process happens. Therefore, the training process can be done in parallel, and because of that the transformer is fast in the training process.

Large Language Model (LLM), is a specific application of the Transformer architecture. It is trained on a large and structured collection of text data, which allows it to learn grammar, syntax, facts, and even some degree of common-sense reasoning. To perform the generation, LLM could use different approaches depending on how many demonstrations are provided at inference time [5]:

- *Fine-Tuning* is the approach that involves updating the weights of a pre-trained model.
- *Few-Shot* refers to a scenario in which the model is provided with a limited number of task (few) demonstrations during inference, acting as conditioning while preventing any weight updates.
- *One-Shot* is the same as few-shot inference, except that only one demonstration or example is allowed.
- *Zero-Shot* is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task.

3.2.5. Diffusion Model

Image synthesis represents one of the computer vision domains that has witnessed remarkable advancements in recent times. However, it also stands out as one of the most computationally high demands, particularly when it comes to generating high-resolution synthesis of complex images. Diffusion Model [19] contains the *forward process* or *diffu-*

sion process, the sampling process that adding the noise, From the original images, then add noise step by step and adds more noise each step, from \mathbf{x}_0 to \mathbf{x}_T . see Equation 3.15.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad (3.15)$$

where $\sqrt{1 - \beta_t}\mathbf{x}_{t-1}$ is the mean and $\beta_t\mathbf{I}$ is the variance. The other process is *reverse process*, which is the process that learns how to remove the noise from the input. The process does not remove the noise once, but step by step. To do that, needed U-Net to train, given the input \mathbf{x}_t and predict the output \mathbf{x}_{t-1} , see the Equation 3.16

$$p_\theta(\mathbf{x}_{0:T}) := p(x_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \quad (3.16)$$

where $\mu_\theta(\mathbf{x}_t, t)$ is the mean, and the variance is $\Sigma_\theta(\mathbf{x}_t, t)$. Predicting the image of \mathbf{x}_{t-1} is equal to predicting the noise of \mathbf{x}_t that want to remove. In the training process, let \mathbf{x}_t be the input, the output of U-Net is ϵ_θ , and the noise of the output is ϵ_t , and the cost function is a mean square error. See Equation 3.17

$$\mathcal{L}oss = MSE(\epsilon_t, \epsilon_\theta(\mathbf{x}_t, t)) \quad (3.17)$$

The idea is to find the minimum error of the predicted noise and sampled noise. *Latent Diffusion Model* (LDM) [49], instead of applying the diffusion process on the pixel-space image, the diffusion process is applied in the latent representation of U-Net. The process is depicted in Figure 3.9

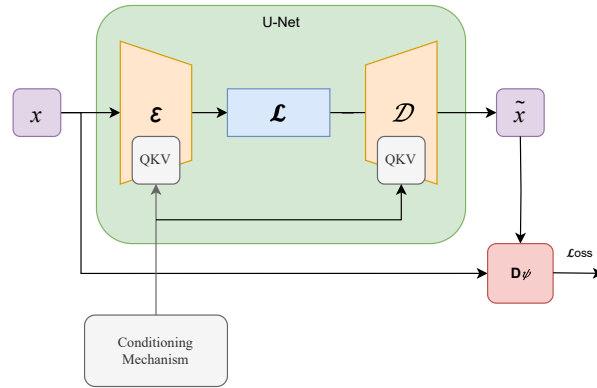


Figure 3.9: Latent Diffusion Model.

The training process will follow the formula in the Equation 3.18

$$\mathcal{L}oss = \min_{\mathcal{E}, \mathcal{D}} \max_{\psi} (\mathcal{L}oss_{rec}(x, \mathcal{D}(\mathcal{E}(x))) - \mathcal{L}oss_{adv}(\mathcal{D}(\mathcal{E}(x))) + \log D_{\psi}(x) + \mathcal{L}oss_{adv}(x; \mathcal{E}, \mathcal{D})) \quad (3.18)$$

where $\mathcal{L}oss_{rec} = MSE(x, \mathcal{D}(\mathcal{E}(x)))$ lost function for autoencoder. $\mathcal{L}oss_{reg} = KL(\mathcal{E}(x) = \mathcal{L}, \mathcal{N}(0, 1))$ will make the latent representation forced to a normal distribution. The adversarial loss function going to measures the output image of the decoder as much as the input, by implementing *Discriminative* function $D_{\psi}(\mathcal{D}(\mathcal{E}(x)) \approx x)$. Minimizing the encoder decoder means, it will have an output image that is similar to the input. The discriminant part comes to predict the encoder input in order to improve the quality of output. After getting the best latent representation, then training the Diffusion Model the process is similar to Diffusion Model training in pixel-space, and freezes the autoencoder part. To generate the image, the procedure is implemented *cross-attention* that combines the conditional input with the input image that is trained with the Diffusion Model.

Controlling Image Diffusion Model [66], aim to control the output of generated image of the Diffusion Model. The idea of architecture is to copy every layer of the Diffusion model, see Figure 3.10.

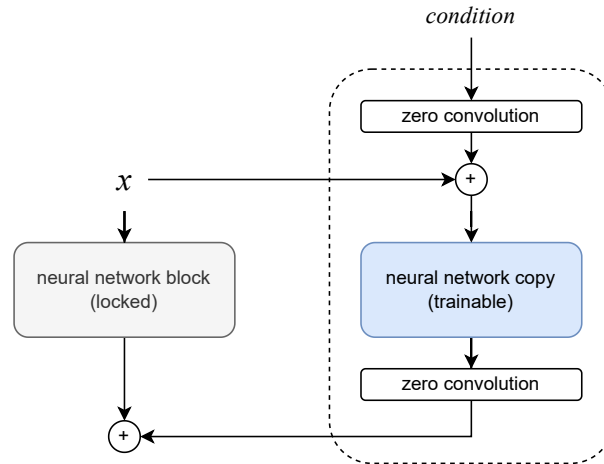


Figure 3.10: Controlling Diffusion Model.

The copied neural network layer is locked, which means when the training process it will not retrain. The copy neural network will train with zero convolution. *Zero convolution* is the convolution process with weight and bias initialized with zero. The inference process combines the original image as a spatial context reference and text as semantic context

conditioning. The original image will be inverted into spatial context conversion. Capturing the spatial context from the original input image can be done with several techniques. Canny edges detection, dept-map, segmentation map, Scribble, Holistically-Nested Edge Detection (HED). Canny edge detection will provide the spatial context as a canny edge detection algorithm result. Depth-map provides the depth image result. Semantic map uses semantic segmentation as context spatial conversion. HED will provide an edge detection algorithm that automatically learns rich hierarchical representations (guided by deep supervision on side responses). See Figure 3.11.

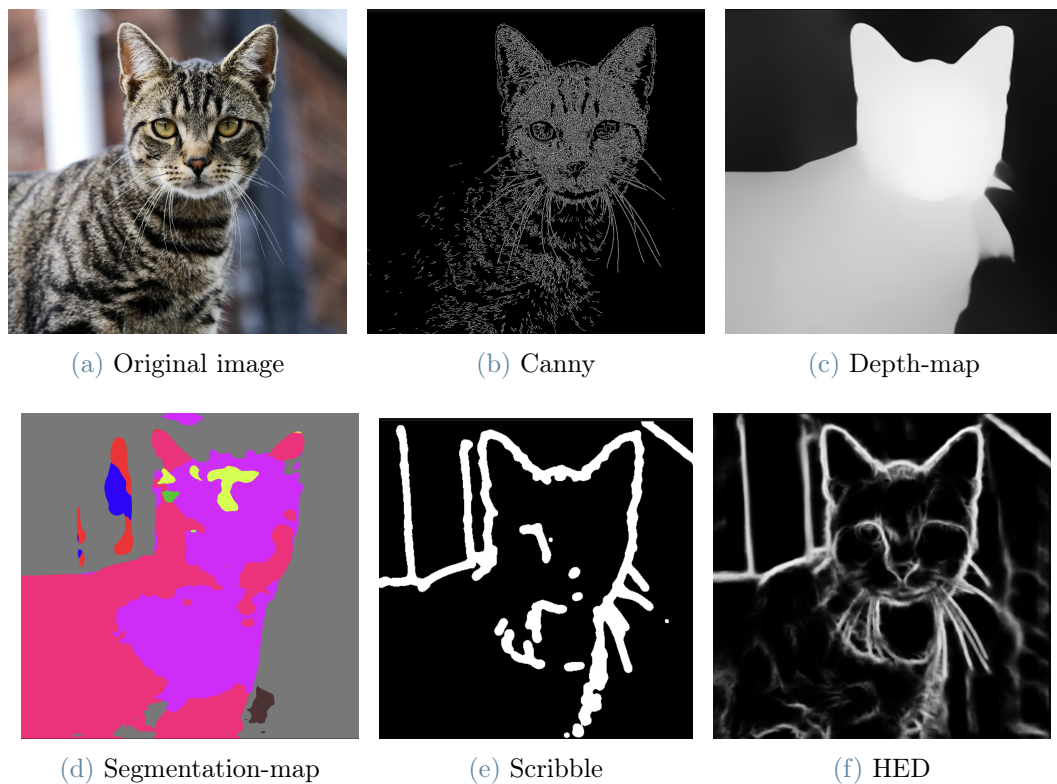


Figure 3.11: Controlling Image with spatial context.

3.3. Metamorphic Testing in Deep Learning

Metamorphic Testing can be used in the context of deep learning to assess the quality and reliability of models. It is based on the principle of metamorphism, which involves transforming or altering the input data and examining the corresponding changes in the model's output. Metamorphic Testing can help in scenarios where traditional testing methods might be inadequate, such as complex and non-deterministic, which is a deep learning scope. Metamorphic Testing can be applied by Generate Test Cases, creating a set of test cases by applying transformations to the original input data. Then Evaluate

the Model, by comparing the model's predictions with the expected outputs. If there are significant discrepancies, it indicates potential issues with the model's robustness or correctness. This may prompt the discovery of the corner case, further investigation or model refinement.

The similarity between traditional software and DNNs. In traditional software, each statement performs a certain operation to transform the output of previous statement to the input to the following statement, whereas in DNN, each neuron transforms the output of previous layer to the input of the following layer. The output layer is related to the receptive field of the input. But unlike traditional software, DNNs do not have explicit branches and flow, but a neuron influences other neurons. A lower output value indicates less influence and vice versa. When the output value of a neuron becomes zero, the neuron does not have any influence on following neurons. Corner case behavior will never be seen unless the test input is not cover all possibility. Similarly, low coverage inputs will also leave different behaviors of DNNs unexplored. Therefore, many incorrect DNN behaviors will remain unexplored even after performing fine-tuning the model with the same input dataset.

Metamorphic Testing offers several advantages in the context of deep learning:

- *Non-Bias Testing*, It can uncover issues that traditional testing methods might miss because it uses transformations of the input data, which can reveal subtle model flaws.
- *Automatic Test Case Generation*, once the Metamorphic Relation is defined, test cases can be automatically generated, making it easier to test a wide range of scenarios.
- *Robustness Assessment*, Metamorphic Testing helps assess a model's robustness to different inputs and transformations, which is important for real-world applications.
- *Continual Testing*, it can be used for continual monitoring of model performance, as new data can be subjected to the same Metamorphic Testing.

4 | Solution

This chapter provides a description of *DILLEMA* (**D**iffusion model and **L**arge **L**anguag**E** Model for **A**ugmentation) to perform alternative generation and data augmentation as the generated test cases for Metamorphic Testing. *DILLEMA* has five-step processes from the input image to generate the image result. See Figure 4.1.

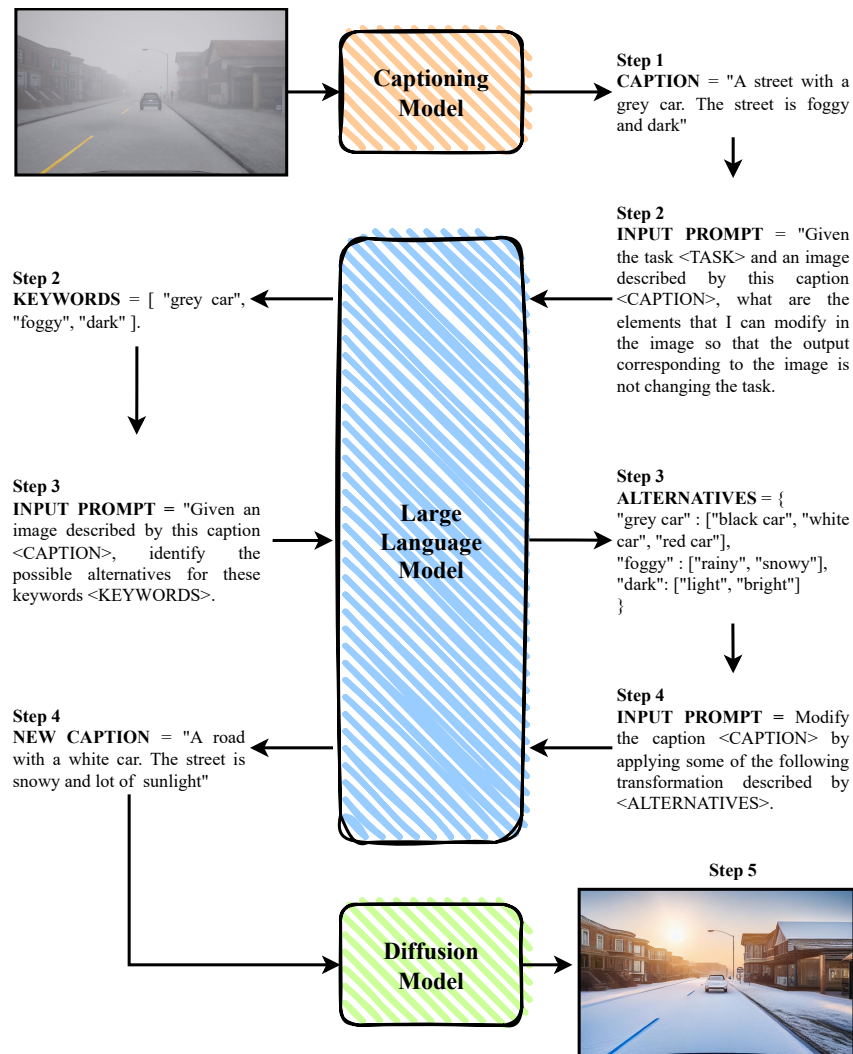


Figure 4.1: DILLEMA schema.

Step 1) the input image is described as text with an image Captioning Model, the result provides some sentences related to the input image that is described as a caption. Step 2) LLM takes place in order to find the words that can be modified corresponding to the description of the caption without changing the task objective, and the expected result (*Keywords*) is a list of the most relevant words in the caption. Step 3) given the specific task, and the caption (the original text description of image input), LLM is asked to identify *Alternatives*, a set of the possible words to replace the elements in *Keywords*. Step 4) creates *New Caption* by changing *Keywords* with their alternatives and generating complete sentences that describe the output of the image. Step 5) control a Diffusion Model to generate a new image based on the original image and *New Caption*.

4.1. Describing the Image with Captioning Model

In step 1, DILLEMA uses an image Captioning Model to capture the description of the image. Captioning Model performs the generation of sentences which semantically equivalent to the image description [31]. In this process, the Captioning Model can be used to generate more than one sentence to depict a comprehensive description of the image [32]. The large number of sentence generations in the caption can cause a problem due to word repetitions. Because in step 4) should avoid word repetition. Increasing the number of sentence generations also increases the time cost of this process. More generated captions, more time needed, and vice-versa. Therefore, finding the optimal number of captions becomes an important hyperparameter in this step. See Figure 4.2 which is an image input.



Figure 4.2: Input image for caption generation.

The result for generated captions that describe the input image with ten sentences,


```
[“a vehicle driving on a foggy road in front of a tree”,  
“a view from a windshield of a car driving down the road”,  
“a car driving down the road and through a forest”,  
“a photo of a street that has trees on it”,  
“a car is driving down the side of a road”,  
“a car on the highway next to a car that’s on the grass”,  
“a car is driving down a road in the mist”,  
“a car driving along a highway in a fog”,  
“a view of impulses of cars on a road”,  
“a view from inside a car of a road”]
```

The generated caption with ten sentences shows many repetitions of the words car or road where excessive word repetition should be avoided.

4.2. Large Language Model (LLM) Application

LLM is used for several processes in DILLEMA, because it is proficient in addressing and solving problems related to natural language [46, 48]. LLM enhances scalability by offering a faster and automated replication process compared to human processes. *Keywords* and *Alternative* generations require understanding the context and semantics of natural language [56]. LLM involves predicting the next word in a sentence in the inference process. This results in the model learning intricate language structures and the importance of different words in diverse contexts [42, 63]. LLM essentially uses its learned knowledge to identify and extract words that hold particular significance within the context of the text to generate *Keyword*. LLM has learned a wide range of synonyms, antonyms, and various ways to express sentences [57]. This semantic knowledge allows LLM to suggest alternative words or phrases that fit the context of the sentence for *Alternative* generation. LLM can consider and disambiguate multiple possible meanings of a word [1], ensuring that the modifications align with the structural natural language of the sentences after applying the alternatives’ words to have *New Caption* from *Alternative* generation. DILLEMA uses three different steps for LLM application because it’s rather easier to have LLM prompt for a specific task instead of combining several tasks into one prompt. It also provides the easiness to manage the result and fine-tune the prompt on specific task.

To optimize the performance of LLM application in DILLEMA, it is essential to consider not only the effectiveness of crafting a suitable prompt but also consider multiple properties for better results. Fine-tuning the control over output randomness stands as a pivotal

property, dictating whether LLM generates a deterministic outcome with repetitive words or more creative result. In the context of the inference process, the performance of LLM is intricately tied to the length of input words. While minimizing the length of the generated result can enhance precision, it introduces challenges when dealing with outputs requiring extensive generation. It's noteworthy that a long-generated result does not necessarily guarantee conciseness, as complexity may still persist in the outcome.

4.2.1. LLM: Generate Keywords

DILLEMA in step 2 uses LLM to identify a set of words that can be altered to align within the description of a caption, ultimately generating a list of keywords that match the intended context. In other words, LLM performs to find a list of words in the caption that can be modified without changing the main objective of the task. For example, if the task is related to the word *car* then it will not give the word *car* in the list of words in the keyword. The prompt to generate *Keyword* with given *Caption* and *Task* can be,

```
"Given the task <TASK> and an image described by this caption <CAPTION>,
what are the elements that I can modify in the image so that the output
corresponding to the image does not change the task."
```

LLM leverages its pre-trained knowledge to discern the contextual relevance of each word. It understands the relationships between words, their syntactic and semantic roles, and the overall coherence of the language. This understanding enables LLM to suggest possible word modifications in the caption, maintaining the intended context and purpose of the task. Moreover, providing the format example result also can help LLM provide a good result. For instance, `result = ["red", "dark", "sunny"]` in this way, LLM provides the result with the given template. Hence, it will not provide a result in arbitrary format.

4.2.2. LLM: Generate Alternatives

In step 3, DILLEMA considers the caption as an initial textual representation of the image input, LLM recognizes the potential *Alternative* for the provided *Keywords*. The outcome of this step is the creation of a formatted string that maps the original *Keywords* to their respective *Alternative* words.

Given *Keyword* from the previous step and the caption that describes the image, the prompt can be,

```
"Given an image described by this caption <CAPTION>, identify the
possible alternatives for these keywords <KEYWORDS>."
```

4.2.3. LLM: Generate New Caption

In step 4, LLM takes *Keywords* and transforms them into selected *Alternatives* within complete sentences. These sentences are designed to convey a new description of the image output, essentially serving as *New Caption* for the image. The process involves replacing *Keywords* with their *Alternatives* to generate new sentences that vividly depict the content and context of the image in a properly structured natural language. Prompt for this step can be,

```
"Modify the caption <CAPTION> by applying some of the following transformations described by <ALTERNATIVES>."
```

4.3. Controlling Image Generation in Diffusion Model

In the step 5, DILLEMA use a controlling Diffusion Model to generate the image from image and text. The aim of Image generation create the test case generation or the data augmentation from the original image. Data augmentation is most effective when it faithfully mirrors real-world scenarios. However, techniques like blurring, rotation, and translation may fall short in capturing diverse and complex real-world environments, particularly in the context of self-driving car vision. Therefore, the traditional transformation needs to be improved. Generative Adversarial Networks (GANs) rely on paired data to learn the mapping from one domain to another in the training process. Consequently, this becomes a challenge when tackling transformations across multiple domains. On the other hand, Diffusion models do not require paired data for training. This presumption to have a mapped dataset may not always be feasible, particularly when dealing with scenarios that have diverse targeted domains. A crucial consideration in self-driving car data augmentation is ensuring that the desired outcome effectively captures the spatial context of the original data. Diffusion Model allows *fine-grained* to have a high level of control or precision over specific details or attributes within an image. In the context of image generation, fine-grained control means being able to make precise adjustments or modifications to various aspects of the image control over the generated images which cannot be effectively done by GANs-based image-to-image transformation.

Controlling Diffusion Model incorporates a conditioning mechanism to control the transformation which uses converted input images into different spatial contexts such as depth-map, canny, segmentation-map, scribble, or HED. The spatial context result recognizes semantic contents in the input. The different spatial contexts provide good results in different specifications and conditions. See Figure 4.3. Image generation results that use

spatial context conditioning mechanisms will be provided in the Implementation section.

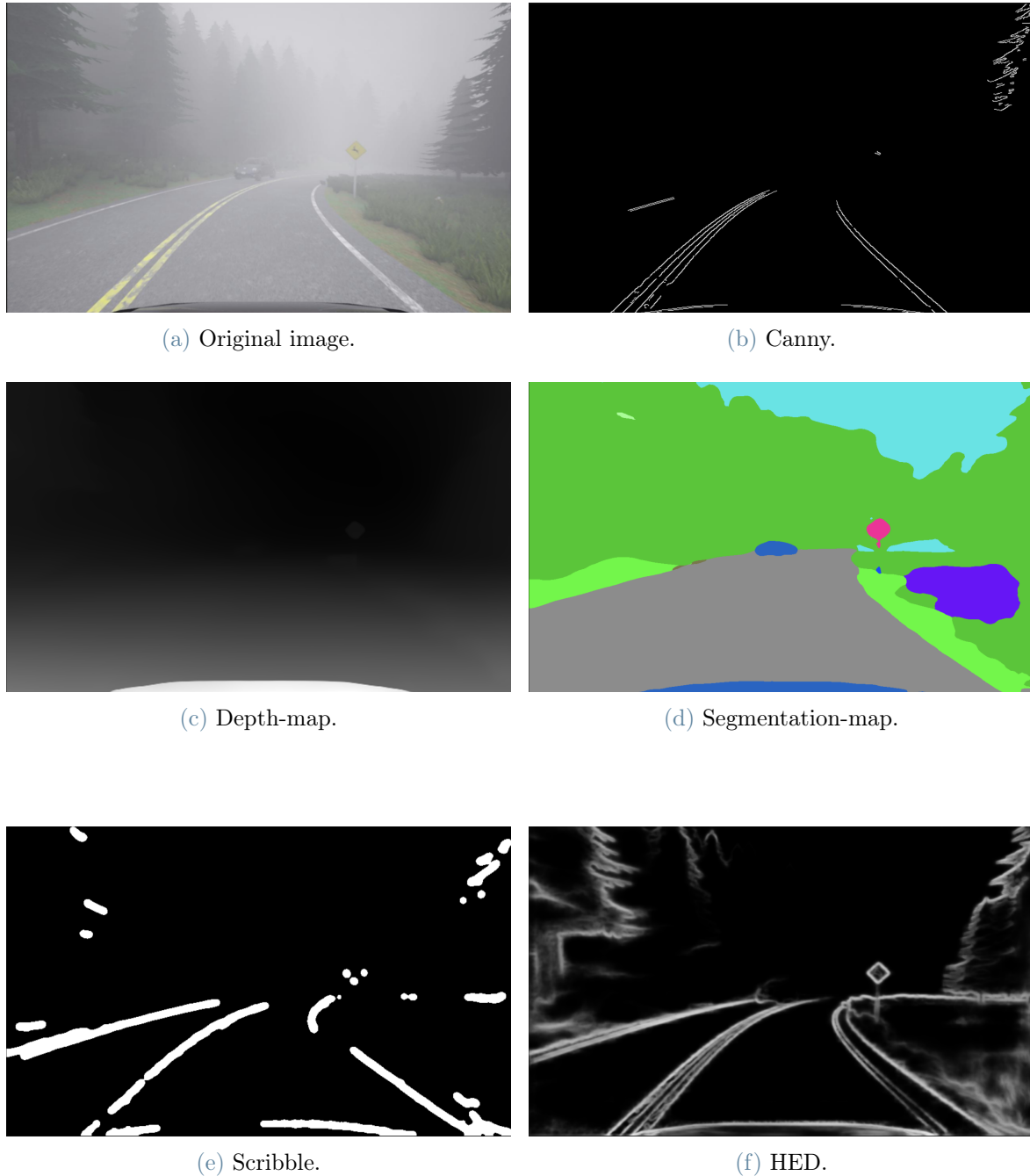


Figure 4.3: Controlling Image with spatial context conversion possibilities.

Depth-map has a good performance when the original input image is essentially represented as a depth image. For canny spatial context, the original input image will convert into an image with canny edge detection. The problem is to find the optimal threshold to have a good spatial context that can capture the original image. Canny edge detection

also encounters challenges in addressing the potential variations in brightness conditions within the original image. Setting the edge detection threshold too low risks losing the spatial context of the original image, while a threshold set too high may introduce noise and alter the contextual information in the converted image i.e. adding the objects that are not in the original image. Hence, it cannot adapt to some variants of input datasets. The segmentation map only works with specific pre-trained classes which make it cannot be used on other task. Another spatial context conversion is HED, it is the optimal option for capturing the spatial context of the original input image. HED performs better compared to other spatial context conversion. It can recognize important semantic content in spatial context conversion without considering any setting. Because of that, HED is used in controlling image generation and combined with *New Caption*. The image generation results are performed by spatial context that is captured from the original image, and text as semantic context from *New Caption* as a prompt.

5 | Implementation

This chapter provides the technicalities of the implementation of DILLEMA. Proposed implementation of Image Captioning Model, Large Language Model, and Diffusion model.

5.1. BLIP-2 Captioning Model

Image Captioning Model uses Bootstrapping Language-Image Pre-training (BLIP), which is a Vision-language pre-training model that refers to the implementation of training models on a large dataset that combines both visual and textual data. This model is designed to learn joint representations of visual and textual information, which can then be used for a variety of tasks that require understanding and generating content that involves both images and text. Implementation of BLIP-2 latest version of BLIP is used for the Captioning Model. In order to have a good result consider another step on the solution process. Applied `nucleus_sampling` provides a better generation of image description. The generation result is more creative and captures the semantics of the image within a sentence. DILLEMA uses pre-trained model without fine-tuning. BLIP-2 has several pre-trained models, `pretrain_opt2.7b`, `pretrain_opt2.7b`, `caption_coco_opt2.7b`, and `caption_coco_opt6.7b` The best option in this case is `caption_coco_opt6.7b`, which provides non-repetitive tokens or sentences. The number of generation sentences also needs to be optimized, because it affects the generating *Alternative* result. Therefore, based on the experiment, two generated sentences are the optimal implementation, because it properly capture the semantics of the image more extensively and does not have the problem in the *Alternative* generation step due to word repetition.

5.2. LLaMA-2 as Large Language Model

Large Language Model Meta AI is an abbreviation of LLaMA, which is a family of large language models. LLM was applied for generating *Keywords*, *Alternatives*, and *New Caption*, specifically using LLaMA-2, the second version of LLaMA. DILLEMA implements a quantized pre-trained LLaMA-2 model without fine-tuning. LLaMA-2 is open-

source but it is as good as GPT-4 (other LLMs family), and much better than GPT-3.5-turbo. This is because LLaMA-2 is trained on a massive dataset of factual information, and it uses a variety of techniques to check the accuracy of its output. LLaMA-2 has three model sizes; 7, 13, and 70 billion parameters. A billion parameters need a huge amount of memory resources. The *quantization* is a technique used to reduce the size of LLM by modifying the precision of their weights. Quantization not only reduces memory needs but also improves time inference. Of course, there is a trade-off in using that, the performance could be decreased. But 13 billion parameters of LLaMA-2 with 5-bit quantized precision provide good performance, with a lower memory cost.

LLaMA-2 needs some settings, `temperature` to manage the output's randomness, reducing the temperature generates responses that are more repetitive and deterministic, whereas increasing the temperature leads to results with greater unpredictability and creativity. `top_p` also controls response randomness, but uses a different method. Lowering `top_p` will narrow the model's token selection to likelier tokens. Increasing `top_p` will let the model choose from tokens with both high and low likelihood. `max_new_tokens` sets a limit on the number of tokens per model response. `repetition_penalty` is a penalty for repeated words in the generated text. `context_length` refers to the total number of tokens permitted input by LLaMA-2. In the device configuration, LLaMA-2 can run with CPU or GPU, if using GPU, it is needed to specify the `gpu_layer`. The number of specified `gpu_layer` contributes to the inference time of LLaMA-2, the higher number induces more time to load into memory, but decreases the inference time. LLM can apply a system prompt, which is an initial prompt to construct the general behavior of LLM.

```
"You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information."
```

The system prompt is constructed as the fundamental behavior of LLaMA-2 for each task in the DILLEMA steps. DILLEMA implements Few-Shot inference because in the prompt provides the example in the prompt to generate *Keyword*.

```
if the task is related to cars, then you cannot put the word car in the list. For example, list = ['red', 'people', 'rainy']
```


Generating *Alternatives* involves more than just providing a simple example; it requires a thorough demonstration to effectively communicate how the response should unfold. This entails not only presenting a straightforward case but also offering a more comprehensive illustration of the desired outcome.

For example "small": ["big", "medium"] which is different type of size, "person": ["woman", "man", "boy", "girl"], "rainy": ["sunny", "snowy", "clear"] which is different condition of weather, "red": ["green", "yellow", "blue"] which is a possible color, or "night": ["day", "lights"]. Provide the result, for example, result = {"small": ["big", "medium"], "person": ["woman", "man", "boy", "girl"], "rainy": ["sunny", "snowy", "clear"], "red": ["green", "yellow", "blue"], "night": ["day", "lights"]}.

New Caption generation needs One-Shot inference because only given one example.

{"night": "morning", "dark": "red"} your response can be result = ["a beautiful road in the morning", "a red street"].

5.3. ControlNet for Image Generation

The control of the Diffusion Model through a conditioning mechanism, which incorporates spatial context and textual descriptions, stands as a crucial component in DILLEMA. The use of GANs-based image-to-image transformation faces limitations as GANs rely on paired data to learn the mapping from one domain to another in order to have specific random noise. In the context of Instruct Pix2Pix, a GANs-based image-to-image transformation approach, the introduction of random noise to the input image poses a challenge by not ensuring the preservation of specific characteristics, such as the spatial context inherent in the input image's structure. The adjustment of the degree of random noise is facilitated through the strength parameter. When applying minimal noise, the resulting image is closely similar to the input image, while adding a significant amount of noise produces a significantly divergent output image. Contrarily, ControlNet stands out by offering a superior level of control, allowing for meticulous adjustments and detailed specifications in the final output. The process involves capturing input images, transforming them into a spatial context, and integrating them with prompts or text that articulate the desired attributes of the augmented image. In the context of DILLEMA, HED spatial context is chosen for its superior performance in addressing this specific problem when compared to alternative methods. ControlNet provides two controlling prompts categories, **prompt** going to infuse the description into an augmentation image, and **negative-prompt** makes

a restriction on the augmented image. Figure 5.1, 5.2, 5.3, 5.4, and 5.5 are the generated images provided by ControlNet starting from the original image in Figure 4.3. These results are generated with prompt: "sunny road, detailed, professional camera", and negative-prompt: "low-resolution". Image generation is able to generate augmented images from spatial context and textual input. The result shows that HED outperforms the other spatial context conversions, see Figure 5.5.

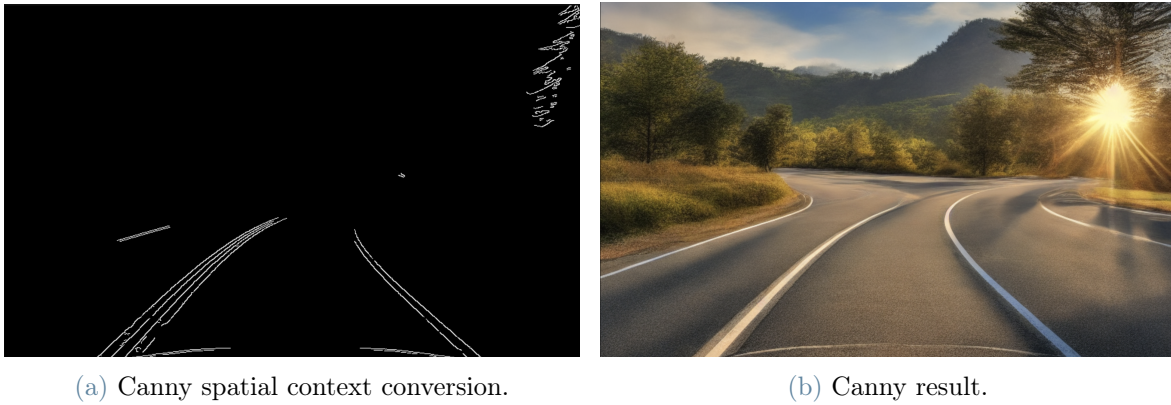


Figure 5.1: ControlNet with Canny spatial context conversion.

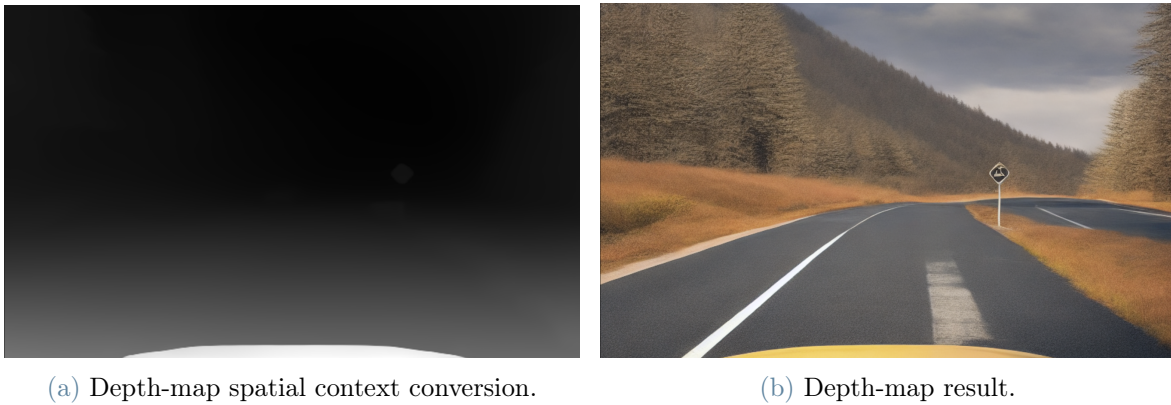


Figure 5.2: ControlNet with Depth-map spatial context conversion.

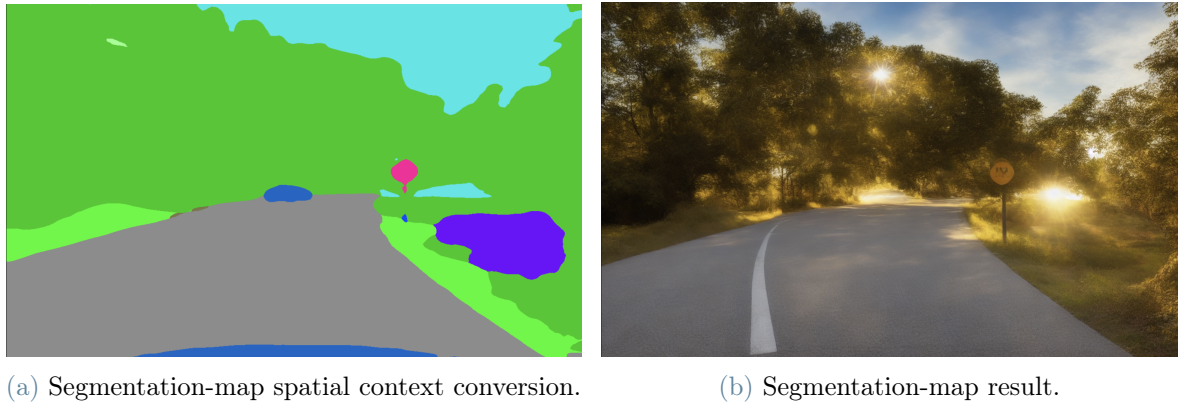


Figure 5.3: ControlNet with Segmentation-map spatial context conversion.

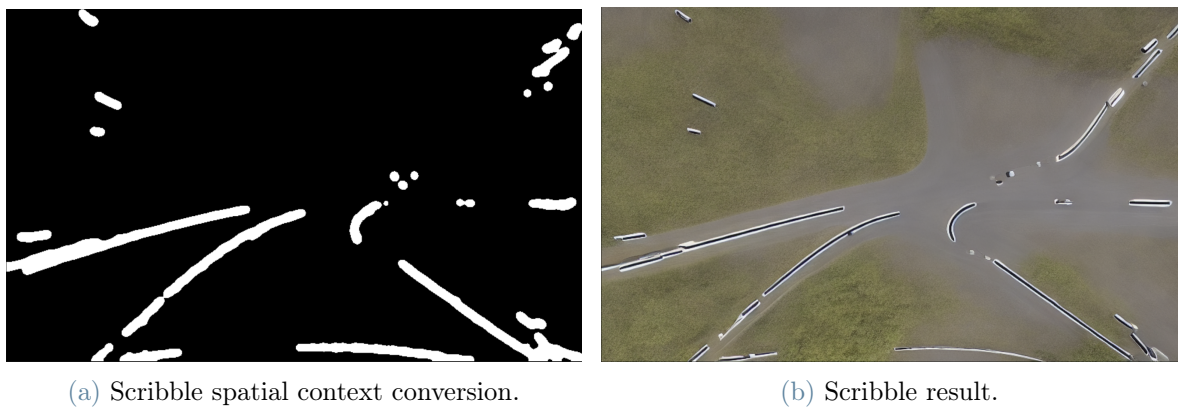


Figure 5.4: ControlNet with Scribble spatial context conversion.

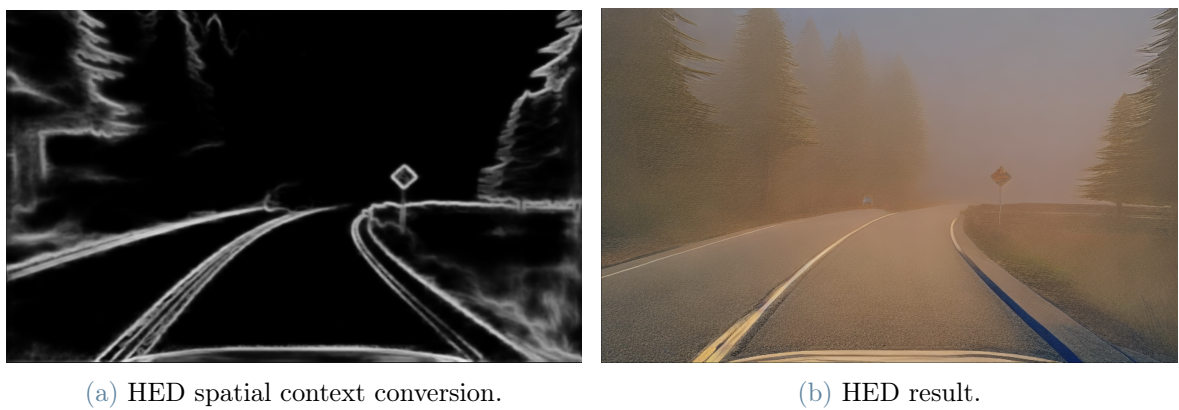


Figure 5.5: ControlNet with HED spatial context conversion.

6 | Evaluation

This chapter provides a critical component where the assessment DILLEMA are presented in detail. To contextualize this evaluation, the methodologies and approaches employed in conducting the evaluation are then thoroughly explained, encompassing the tools, techniques, and datasets. Starting with the experimental description section that describes the specific configurations, parameters, and conditions employed during the evaluation. Experimental setup related to the hardware and software environments in which the experiments were conducted. The Results section, presents evaluations, utilizing tables, and textual descriptions. While also acknowledging the limitations of the evaluation process.

6.1. Experimental Description

In practical implementation, DILLEMA strategically divides its processes into three distinct processes to optimize computational resources. These processes encompass *image captioning*, *alternative generation*, and *image generation*. This structure ensures efficiency by minimizing the overall computational resource requirements. Image captioning uses BLIP-2 which needs 20Gb memory. While LLaMA-2 needs 12Gb memory to process *alternative generation*, and 12Gb is needed by ControlNet to generate images. Based on this experimental specification DILLEMA should be deployed in the proper machine. In the experiment, DILLEMA was applied to two datasets, *ImageNet1K*, is a dataset commonly used for training and evaluating image classification task. It consists of 1.2 million images from 1,000 different classes [50]. *SHIFT* provides multitask synthetic dataset tailored for autonomous driving [54]. SHIFT encompasses both discrete and continuous shifts in driving scenarios, Featuring a comprehensive sensor suite and annotations for several mainstream perception tasks. However, in the context of this thesis, emphasis is placed on leveraging the image segmentation task within the SHIFT dataset. Notably, SHIFT includes a Multi-View RGB camera set, and for the scope of this study, the front-facing camera is employed. This choice aligns with the precedent set by notable benchmarks such as the Udacity Self Driving Car Challenge and Nvidia Dave-2, both of which utilized front-facing cameras to predict steering commands in the domain of self-driving cars [2].

In the results section, a detailed comparison unfolds between the original dataset and its augmented data, using diverse pre-trained models (ResNet18, ResNet50, and ResNet152) for the ImageNet1K classification task. Acknowledging constraints of limited time and available resources, this thesis strategically provides a focused comparison. To optimize efficiency, the comparison involves 25 images per class, each accompanied by 5 augmented versions, forming a representative subset within the ImageNet1K dataset. In contrast, the entire SHIFT dataset is leveraged, ensuring a comprehensive evaluation. The augmentation process for each image takes approximately 6 seconds, resulting in a cumulative processing time of 4 days for the ImageNet1K dataset. Meanwhile, for the SHIFT dataset, the entire augmentation process is completed within 1 day.

In the evaluation phase, this thesis extends the analysis to include the testing results of fine-tuning the pre-trained model. The goal of this retraining process is to show the model's robustness enhancement by incorporating and adapting to the generated data. Fine-tuning involves adjusting the parameters of the pre-trained model using the additional data generated by DILLEMA. This approach is undertaken with the intention of improving the model's ability to handle diverse scenarios, including corner cases and unexpected inputs.





6.2. Experimental Setup

DILLEMA deployed into a machine with installed `pytorch: 2.0.1` and `CUDA: 11.8`. BLIP-2 needs to be installed `salesforce-lavis==1.0.2` which provides the backbone architecture and uses `caption_coco_opt6.7b` for the preferred parameters. We set `nucleus_sampling=True` and define the number of captions that should be generated with `num_caption=2` and then set the device into `"cuda"`. LLaMA-2 is running on top of `ctransformers==0.2.27` with `gpu_layers=500` and with this specification setting, `"temperature": 0.1`, `"top_p": 0.85`, and `"repetition_penalty": 1` to manage the result's randomness. `"max_new_tokens": 2048`, `"context_length": 2048` controlling the result related to context and length of the token result. DILLEMA implements the quantized ControlNet model and it is installed on top of `diffusers[torch]==0.21.4` and `controlnet_aux==0.0.7` which provides the conversion of HED spatial context. Due to constraints posed by the limited processing power of our machine's GPU, DILLEMA employs a quantized ControlNet model. The pretrained quantization is provided by *HuggingFace*, `llama-2-13b-chat.ggmlv3.q5_1.bin`.

6.3. Result

This section provides DILLEMA comprehensive overview of the practical functioning process, delineating its structure across various processes, including *image captioning*, *alternative generation*, and *image generation*. The result shows that leveraging augmented data introduces a valuable aspect of randomness to the output. This is particularly advantageous since it enables the generation of a multitude of distinct images from just a single *alternative generation*. Table 6.1 shows ImageNet1K each original data generates 5 augmented data. The first five images are classified as "Bird". These images have the original caption, *"a yellow bird standing on a wide branch, a yellow bird sitting on a front twig looking off to the front"*. DILLEMA augmented these images and generated an alternative version of the bird with different color (blue), different tree textures, and different backgrounds. Note that both the original and the augmented version should be still classified as "Bird". Table 6.2 shows mapping between a single original data point and its corresponding augmented data in SHIFT dataset. DILLEMA provides the augmented image by changing the environmental condition from the original image (e.g. sunny to snowy, morning to evening, and dark to light).







Table 6.1: DILLEMA images augmentation result on ImageNet1K.


Original Image	New Caption	Augmented Image
	<i>a blue bird standing on a wide branch, a blue bird sitting on a front twig looking off to the front</i>	
	<i>a blue bird standing on a wide branch, a blue bird sitting on a front twig looking off to the front</i>	

Original Image	New Caption	Augmented Image
	<p><i>a blue bird standing on a wide branch, a blue bird sitting on a front twig looking off to the front</i></p>	
	<p><i>a blue bird standing on a wide branch, a blue bird sitting on a front twig looking off to the front</i></p>	
	<p><i>a blue bird standing on a wide branch, a blue bird sitting on a front twig looking off to the front</i></p>	
	<p><i>a vulture is lying beside some bushes, a picture of a bird lying beside some bushes</i></p>	
	<p><i>a vulture is lying beside some bushes, a picture of a bird lying beside some bushes</i></p>	

Original Image	New Caption	Augmented Image
	<p><i>a vulture is lying beside some bushes, a picture of a bird lying beside some bushes</i></p>	
	<p><i>a vulture is lying beside some bushes, a picture of a bird lying beside some bushes</i></p>	
	<p><i>a vulture is lying beside some bushes, a picture of a bird lying beside some bushes</i></p>	
	<p><i>several black chickens and a couple of red and white chickens stand on a paved, a group of chickens stand on a paved</i></p>	
	<p><i>several black chickens and a couple of red and white chickens stand on a paved, a group of chickens stand on a paved</i></p>	



Original Image	New Caption	Augmented Image
	<p><i>several black chickens and a couple of red and white chickens stand on a paved, a group of chickens stand on a paved</i></p>	
	<p><i>several black chickens and a couple of red and white chickens stand on a paved, a group of chickens stand on a paved</i></p>	
	<p><i>several black chickens and a couple of red and white chickens stand on a paved, a group of chickens stand on a paved</i></p>	
	<p><i>a small bird is perched on some smooth rocks, a bird standing on some smooth rocks in the still</i></p>	
	<p><i>a small bird is perched on some smooth rocks, a bird standing on some smooth rocks in the still</i></p>	
	<p><i>a small bird is perched on some smooth rocks, a bird standing on some smooth rocks in the still</i></p>	

Original Image	New Caption	Augmented Image
	<p><i>a small bird is perched on some smooth rocks, a bird standing on some smooth rocks in the still</i></p>	
	<p><i>a small bird is perched on some smooth rocks, a bird standing on some smooth rocks in the still</i></p>	
	<p><i>a large white bird perched on top of a thick tree branch, the white and gray bird is sitting in a short tree</i></p>	

Original Image	New Caption	Augmented Image
	<p><i>a large white bird perched on top of a thick tree branch, the white and gray bird is sitting in a short tree</i></p>	
	<p><i>a large white bird perched on top of a thick tree branch, the white and gray bird is sitting in a short tree</i></p>	

Original Image	New Caption	Augmented Image
	<p><i>a large white bird perched on top of a thick tree branch, the white and gray bird is sitting in a short tree</i></p>	
	<p><i>a large white bird perched on top of a thick tree branch, the white and gray bird is sitting in a short tree</i></p>	

Table 6.2: DILLEMA images augmentation result on SHIFT.

Original Image	New Caption	Augmented Image
	<p><i>a car is driving through a very sunny area, an animated view of several lights in a rural area</i></p>	

Original Image	New Caption	Augmented Image
	<i>the scene of the car driving down a wet sidewalk, a city street with a blue vehicle going on it</i>	
	<i>a couple of cars driving through an alley on a snowy day, evening in the city with sunsets</i>	
	<i>there are some cars is driven in the middle of the real road, several cars driving around in a standstill</i>	
	<i>cars moving down the alley at lights, an image of many cars driving on a bright road at lights</i>	
	<i>a highway with two cars stopped in front of a house, two cars are stopped and a man standing in front of the house</i>	
	<i>a snowy street view from a moving car looking into an empty street, a sunny view of a city from above the street</i>	
	<i>a road with a stop sign in the foreground, a car is driving down a street and a stop sign is gravel</i>	

6.3.1. ImageNet1K Result

This section provides a comparative analysis of testing performance between the original dataset and augmented dataset, employing metrics previously detailed in the Background section 3.2.3: accuracy, precision, recall, and F1 score. Tables 6.3 - 6.6 display results that measure the pre-trained model's generalization and indicate potential corner cases. Table 6.7 presents the faulty cases found in the state-of-the-art of neural networks.

Table 6.3: Accuracy score.

Pre-trained Model	Original Data %	Augmented Data %
ResNet18	78.97	46.70
ResNet50	89.79	54.52
ResNet152	94.11	57.66
Average	87.62	52.96

Table 6.4: Precision score.

Pre-trained Model	Original Data %	Augmented Data %
ResNet18	79.59	50.44
ResNet50	90.49	58.77
ResNet152	94.55	61.25
Average	88.21	56.82

Table 6.5: Recall score.

Pre-trained Model	Original Dat %	Augmented Data %
ResNet18	78.97	46.70
ResNet50	89.79	54.52
ResNet152	94.11	57.66
Average	87.62	52.96

Table 6.6: F1 score.

Pre-trained Model	Original Data %	Augmented Data %
ResNet18	78.69	46.00
ResNet50	89.70	54.00
ResNet152	94.05	57.14
Average	87.48	52.38

Based on the result above, the augmented data are dropped by average 34.66% for accuracy score, 31.39% for precision score, 34.66% for recall score, and 35.10% for F1 score.

Table 6.7: Faulty behavior on ImageNet1K.

Model	Original Data	Augmented Data
ResNet18	5257	66619
ResNet50	2551	56843
ResNet152	1472	52920

Based on data shows that there exists a corner case which means the model is not able to generalize well.

6.3.2. ResNet18 Performance

In this section provides the comparison performance between *Top K* classes which has highest and lowest accuracy on the original data and their counterparts in the augmented data. Where $K : 10, 20, 50, 100$. An additional aspect of the study examines comparison between the original data which is correctly predicted by the model with their augmented data. Table 6.8 illustrates the frequency of augmented data being classified as true (ranging from 0 to 5) when the original data is also true. This table quantifies the instances when the original data is true, with a corresponding scale indicating the number of times augmented data is predicted as true—ranging from 0 (no augmented data predicted as true) to 5 (all augmented data predicted as true). The frequency column denotes how often augmented data is classified as true for a specific number of true predictions.

Table 6.8: Quantifying instances of augmented is true predicted when the original data is true.

True Predicted	Frequency %
0	24.85
1	10.91
2	8.81
3	9.74
4	12.41
5	33.29

Confusion Matrix

This section presents a comparison of the confusion matrix between the predictions for the original data and augmented data, measured in percentages. Table 6.9 specifically illustrates the outcomes for various classes, each belonging to the broader category of *bird species*. Table 6.10 shows the result of augmented data. Likewise with Table 6.11 and Table 6.12.

Table 6.9: Confusion matrix of original data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	96	0	0	0	0
brambling	0	100	0	0	0
goldfinch	0	0	100	0	0
house finch	0	0	0	100	0
junco	0	0	0	0	92

Table 6.10: Confusion matrix of augmented data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	73.6	0	0	0	0
brambling	0	42.4	12	1.6	1.6
goldfinch	0	0.8	58.4	0.8	0
house finch	0	8.8	26.4	14.4	2.4
junco	0	15.2	8.8	0.8	12.8

Table 6.11: Confusion matrix of original data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	100	0	0	0	0
robin	0	100	0	0	0
bulbul	0	0	92	0	0
jay	4	0	0	96	0
magpie	0	0	0	0	92

Table 6.12: Confusion matrix of augmented data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	51.2	0	0	5.6	0
robin	2.4	11.2	2.4	4	2.4
bulbul	0.8	2.4	20.8	6.4	2.4
jay	14.4	0.8	0.8	51.2	0.8
magpie	1.6	0.8	0	12.8	38.4

This outcome indicates that ResNet18 does not exhibit strong generalization capabilities for *bird species*. The performance for these classes falls below the level observed with the original data.

Top-10 Best Accuracy

Table 6.13 shows the Top-10 classes with the highest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance, presented as percentages. The results reveal errors in the model’s inference, indicating a challenge in effectively handling corner cases due to insufficient model generalization. This is evident from the observed decrease in accuracy when assessing the model on augmented data, simulating a specific instance of encountering corner cases.

Table 6.13: Top-10 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Cabbage Butterfly	100	31.2
2	Buckeye	100	34.4
3	Bison	100	61.6

	Label	Original Accuracy %	Augmented Accuracy %
4	Yellow Lady's Slipper	100	41.6
5	Daisy	100	85.6
6	Komodo Dragon	100	26.4
7	American Alligator	100	64.8
8	Orangutan	100	65.6
9	Speedboat	100	76.0
10	Colobus	100	21.6

The average accuracy for the Top-10 predictions on original images stands at an impressive 100%. However, when subjected to data augmentation, the model reveals a notable drop in accuracy, indicating a challenge in maintaining performance comparable to that on original images. This observation suggests that the pre-trained model may not possess robust generalization properties. The fact that the model's performance on augmented data is only 50% in accuracy.

Top-20 Best Accuracy

Table 6.14 is the continuation of Top-10 best accuracy 6.13 presents the Top-20 classes with the highest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance. The results reveal errors in the model's inference, pointing to a challenge in effectively handling corner cases due to insufficient model generalization. This becomes apparent through the observed decline in accuracy when evaluating the model on augmented inputs.

Table 6.14: Top-20 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Torch	100	68.8
12	Triumphal Arch	100	85.6
13	Dandie Dinmont	100	3.2
14	Sulphur Butterfly	100	39.2
15	Prairie Chicken	100	37.6
16	Peacock	100	87.2
17	African Grey	100	23.2
18	Birdhouse	100	91.2

	Label	Original Accuracy %	Augmented Accuracy %
19	Sulphur-Crested Cockatoo	100	33.6
20	Lorikeet	100	17.6

The Top-20 predictions achieve an impressive 100% average accuracy on original images. Yet, upon undergoing data augmentation, the model exhibits a significant decline in accuracy, pointing to a challenge in sustaining performance levels similar to those on the original images. This observation implies that the pre-trained model may lack robust generalization properties. In the result, the model's accuracy on augmented data is limited to 49.8%.

Top-50 Best Accuracy

Table 6.15 extends the Top-10 best accuracy from Table 6.13 and Top-20 best accuracy from Table 6.14 by presenting the Top-50 classes with the highest accuracy scores in the original dataset. The table offers a comparison with their corresponding accuracy scores derived from augmented data performance. In the subsequent analysis, the results underscore errors in the model's inference, signaling a challenge in effectively addressing corner cases due to insufficient model generalization. This challenge becomes evident through the observed decline in accuracy when evaluating the model on augmented inputs.

Table 6.15: Top-50 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Coucal	100	17.6
22	Sea Lion	100	69.6
23	Jacamar	100	28.8
24	Toucan	100	57.6
25	Albatross	100	56.0
26	Red-Backed Sand- piper	100	36.8
27	Ruddy Turnstone	100	14.4
28	European Gallinule	100	21.6
29	Broccoli	100	80.0
30	Jigsaw Puzzle	100	83.2
31	Jack-O'-Lantern	100	87.2

	Label	Original Accuracy %	Augmented Accuracy %
32	Hip	100	44.8
33	Steam Locomotive	100	86.4
34	Chambered Nautilus	100	69.6
35	Croquet Ball	100	48.8
36	Brown Bear	100	87.2
37	Brambling	100	42.4
38	Porcupine	100	76.8
39	Tiger	100	93.6
40	House Finch	100	14.4
41	Radio Telescope	100	84.0
42	Indigo Bunting	100	51.2
43	Robin	100	11.2
44	Snow Leopard	100	36.0
45	Ice Bear	100	90.4
46	Ladybug	100	80.8
47	Dung Beetle	100	57.6
48	Admiral	100	38.4
49	Recreational Vehicle	100	84.0
50	Pool Table	100	96.8

The Top-50 predictions achieve a remarkable 100% average accuracy on original images. However, when undergoing data augmentation, the model experiences a noticeable decrease in accuracy, indicating a challenge in maintaining performance levels comparable to those observed with original images. This observation raises questions about the pre-trained model's ability to generalize robustly. In practical terms, the model's accuracy on augmented data is only 54.8%.

Top-100 Best Accuracy

Table 6.16, which presents the Top-100 classes with the highest accuracy scores in the original dataset, serves as an extension of the Top-10 best accuracy from Table 6.13, Top-20 best accuracy from Table 6.14, and Top-50 highest accuracy from Table 6.15. The table facilitates a comparison with their corresponding accuracy scores derived from augmented data performance. The results emphasize errors in the model's inference, signaling a challenge in effectively addressing corner cases due to insufficient model generalization. This challenge becomes apparent through the observed decline in accuracy when evaluating

the model on augmented inputs.

Table 6.16: Top-100 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Sorrel	100	76.8
52	Zebra	100	96.0
53	Monarch	100	60.8
54	Goldfinch	100	58.4
55	Rhinoceros Beetle	96	41.6
56	Trifle	96	49.6
57	Blenheim Spaniel	96	23.2
58	Manhole Cover	96	44.0
59	Weevil	96	10.4
60	Fox Squirrel	96	85.6
61	Maypole	96	74.4
62	Killer Whale	96	87.2
63	Pekinese	96	40.8
64	Basketball	96	75.2
65	King Penguin	96	88.8
66	Angora	96	64.0
67	Walking Stick	96	65.6
68	American Coot	96	24.8
69	Leafhopper	96	44.0
70	Limpkin	96	16.8
71	Potter'S Wheel	96	73.6
72	Flamingo	96	96.8
73	Knot	96	72.8
74	Jinrikisha	96	60.8
75	Bullet Train	96	59.2
76	Fiddler Crab	96	48.0
77	Cab	96	79.2
78	Maze	96	56.0
79	Warthog	96	28.8
80	Norwegian Elkhound	96	0.8
81	Model T	96	84.8
82	Tibetan Mastiff	96	12.8

	Label	Original Accuracy %	Augmented Accuracy %
83	Lion	96	90.4
84	Saint Bernard	96	24.0
85	Jaguar	96	81.6
86	Snowmobile	96	84.8
87	Leonberg	96	6.4
88	Pomeranian	96	44.8
89	Chow	96	30.4
90	Pinwheel	96	87.2
91	Arabian Camel	96	88.0
92	Plane	96	58.4
93	Cougar	96	64.0
94	Dhole	96	1.6
95	Running Shoe	96	76.0
96	Planetarium	96	76.0
97	Soccer Ball	96	83.2
98	Pickelhaube	96	68.8
99	Chiton	96	30.4
100	Odometer	96	95.2

The Top-100 predictions show that 98.1% average accuracy on original images. Yet, as the model is tested into the data augmentation, there's a discernible dip in accuracy. This observation prompts contemplation regarding the pre-trained model's are not well generalization, the fact that the performance of accuracy is 54.8%

Top-10 Lowest Accuracy

Similar to the previous evaluation, but instead of having the highest accuracy classes, Table 6.17 shows the Top-10 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.17: Top-10 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Maillot	16.0	14.4
2	Lampshade	24.0	36.0

	Label	Original Accuracy %	Augmented Accuracy %
3	Cassette Player	28.0	8.8
4	Black-And-Tan Coonhound	32.0	10.4
5	Night Snake	32.0	15.2
6	Green Lizard	32.0	52.8
7	Band Aid	36.0	22.4
8	Garden Spider	36.0	24.0
9	Purse	36.0	22.4
10	Spatula	36.0	22.4

The average accuracy for the Top-10 lowest predictions on the original images is 30.8%. Conversely, the average accuracy on the augmented data for the Top-10 lowest predictions, relative to the original image results, is 22.8%. Despite the lowest accuracy scenario, these findings indicate that the data augmentation results are closely aligned with those of the original data.

Top-20 Lowest Accuracy

Table 6.18 is the continuation of Top-10 lowest accuracy 6.17 presents the Top-20 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance. The results reveal errors in the model's inference, pointing to a challenge in effectively handling corner cases due to insufficient model generalization.

Table 6.18: Top-20 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Ram	40	48.0
12	Sunglass	40	34.4
13	Plunger	40	13.6
14	Siberian Husky	40	16.8
15	Television	40	32.0
16	Collie	40	21.6
17	Water Bottle	40	31.2
18	Corn	40	26.4
19	Hook	40	22.4

	Label	Original Accuracy %	Augmented Accuracy %
20	Space Bar	40	9.6

The average accuracy for the Top-20 lowest predictions on the original images stands at 35.4%. The result predictions on the augmented data which its classes in the Top-10 lowest accuracy score, the average accuracy score is 24.2%.

Top-50 Lowest Accuracy

Table 6.19 extends the Top-10 lowest accuracy from Table 6.17 and Top-20 lowest accuracy from Table 6.18 by presenting the Top-50 classes with the lowest accuracy scores in the original dataset. The table shows a comparison with the corresponding accuracy scores derived from augmented data performance. Providing the empiric result indicating a challenge in effectively addressing corner cases due to insufficient model generalization.

Table 6.19: Top-50 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Letter Opener	40.0	36.8
22	Lens Cap	44.0	32.8
23	Pole	44.0	38.4
24	Seashore	44.0	30.4
25	Sandbar	44.0	30.4
26	Eskimo Dog	44.0	42.4
27	Dock	44.0	29.6
28	Hand Blower	44.0	30.4
29	Syringe	44.0	10.4
30	Projectile	44.0	35.2
31	Bathtub	44.0	31.2
32	Overskirt	44.0	21.6
33	Ruffed Grouse	44.0	2.4
34	Ice Cream	44.0	29.6
35	Australian Terrier	44.0	15.2
36	Monitor	44.0	51.2
37	Wooden Spoon	44.0	43.2
38	Red Wolf	44.0	4.0
39	Water Jug	44.0	19.2

	Label	Original Accuracy %	Augmented Accuracy %
40	American Staffordshire Terrier	44.0	11.2
41	Hatchet	44.0	14.4
42	Stethoscope	44.0	52.0
43	Cleaver	48.0	36.0
44	Bath Towel	48.0	34.4
45	Breastplate	48.0	18.4
46	Monastery	48.0	19.2
47	Laptop	48.0	26.4
48	Handkerchief	48.0	24.0
49	Cradle	48.0	45.6
50	Patas	48.0	12.0

The average accuracy for the Top-50 lowest predictions on the original images is 41.1%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-50 lowest accuracy scores, yield an average accuracy score of 26.2%.

Top-100 Lowest Accuracy

Table 6.20, which presents the Top-100 classes with the lowest accuracy scores in the original dataset, serves as an extension of the Top-10 lowest accuracy from Table 6.17, Top-20 lowest accuracy from Table 6.18, and Top-50 lowest accuracy from Table 6.19. The table facilitates a comparison with their corresponding accuracy scores derived from augmented data performance.

Table 6.20: Top-100 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Mouse	48.0	54.4
52	Weasel	48.0	19.2
53	Tiger Cat	48.0	29.6
54	Appenzeller	48.0	6.4
55	English Foxhound	52.0	11.2
56	Microphone	52.0	58.4
57	Maraca	52.0	16.8
58	Mailbag	52.0	41.6

	Label	Original Accuracy %	Augmented Accuracy %
59	Library	52.0	56.8
60	Wine Bottle	52.0	47.2
61	Wreck	52.0	13.6
62	Screen	52.0	43.2
63	Miniature Schnauzer	52.0	8.8
64	Mushroom	52.0	51.2
65	Cliff	52.0	31.2
66	Hair Spray	52.0	31.2
67	Church	52.0	61.6
68	Cloak	52.0	37.6
69	Lakeside	52.0	33.6
70	Crate	52.0	28.8
71	Paddle	52.0	48.8
72	Tape Player	52.0	28.0
73	Velvet	52.0	13.6
74	Digital Clock	52.0	36.0
75	Plate Rack	52.0	49.6
76	Toy Poodle	52.0	19.2
77	Radio	52.0	36.8
78	Cardigan	52.0	14.4
79	Barrel	52.0	48.8
80	Hognose Snake	52.0	10.4
81	Restaurant	56.0	43.2
82	Windsor Tie	56.0	48.8
83	Diaper	56.0	34.4
84	English Springer	56.0	4.0
85	Lhasa	56.0	13.6
86	Canoe	56.0	62.4
87	Rifle	56.0	48.0
88	Horizontal Bar	56.0	45.6
89	Harvester	56.0	45.6
90	Passenger Car	56.0	38.4
91	Soap Dispenser	56.0	44.0
92	Chihuahua	56.0	52.0
93	Chime	56.0	52.0

	Label	Original Accuracy %	Augmented Accuracy %
94	Egyptian Cat	56.0	38.4
95	Shih-Tzu	56.0	23.2
96	Bassinet	56.0	36.8
97	Ballpoint	56.0	53.6
98	Breakwater	56.0	31.2
99	Tub	56.0	32.8
100	Lotion	56.0	36.0

The average accuracy for the Top-100 lowest predictions on the original images is 47.2%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-100 lowest accuracy scores, yield an average accuracy score of 30.84%.

6.3.3. ResNet50 Performance

In this section provides the comparison performance between *Top K* classes which has highest and lowest accuracy on the original data and their counterparts in the augmented data. The inference result uses ResNet50 pre-trained model. This section also provide the result of prediction for augmented instance align with instances where the original data is predicted correctly. Table 6.21 depicts the occurrence of augmented data being labeled as true (ranging from 0 to 5) when the initial data is also true. It provides a quantification of situations where the original data is true, indicating the frequency of instances in which augmented data is predicted as true. The scale ranges from 0 (indicating no instances of augmented data being predicted as true) to 5 (representing all augmented data being predicted as true). The frequency column specifies the frequency with which augmented data is classified as true for a given number of true predictions.

Table 6.21: Quantifying instances of augmented is true predicted when the original data is true.

True Predicted	Frequency %
0	22.93
1	9.69
2	8.05
3	8.59
4	11.42
5	39.32

Confusion Matrix

In this section, depict a comparative analysis of the confusion matrices for predictions on both original and augmented data, expressed as percentages. Table 6.22 - 6.25 provides a detailed breakdown of outcomes for various classes within the broader category of *bird species*.

Table 6.22: Confusion matrix of original data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	100	0	0	0	0
brambling	0	100	0	0	0
goldfinch	0	0	100	0	0
house finch	0	0	0	100	0
junco	0	0	0	0	96

Table 6.23: Confusion matrix of augmented data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	84	0	0	0	0
brambling	0	54.4	9.6	0.8	2.4
goldfinch	0	4	54.4	0	0
house finch	0	16.8	16	16.8	1.6
junco	0	20.8	5.6	2.4	4.8

Table 6.24: Confusion matrix of original data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	100	0	0	0	0
robin	0	100	0	0	0
bulbul	0	0	100	0	0
jay	0	0	0	100	0
magpie	0	0	0	0	96

Table 6.25: Confusion matrix of augmented data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	58.4	0	0	4.8	0
robin	2.4	12	4.8	3.2	3.2
bulbul	1.6	0.8	45.6	8	3.2
jay	5.6	0	4	69.6	1.6
magpie	0	0.8	2.4	10.4	44.8

For *bird species*, this observation follows a similar process, revealing the limitation in ResNet50’s capacity to effectively generalize. Its performance falls short of the level seen in the original data; however, there is a noteworthy improvement when contrasted with the performance of ResNet18.

Top-10 Best Accuracy

Table 6.26 shows the Top-10 class of data augmentation result with respect to the Top-10 highest accuracy score. The findings expose inaccuracies in the model’s inference, highlighting a difficulty in adeptly addressing corner cases owing to inadequate model generalization. This is apparent in the noticeable decline in accuracy when evaluating the model on augmented data, replicating a particular scenario of encountering corner cases.

Table 6.26: Top-10 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Dishrag	100	48.8
2	Pomeranian	100	53.6
3	Hyena	100	85.6
4	Rotisserie	100	58.4
5	Dhole	100	0.0
6	Rugby Ball	100	60.8
7	Disk Brake	100	87.2
8	Harmonica	100	70.4
9	Chow	100	28.0
10	Newfoundland	100	11.2

The average accuracy for the Top-10 highest class predictions with ResNet50 on original images stands at an impressive 100%. However, when subjected to data augmentation,

the model reveals a notable drop in accuracy, indicating a challenge in maintaining performance comparable to that on original images. This observation suggests that the pre-trained model may not possess robust generalization properties. In practice, the model's performance on augmented data is only 50.4% in accuracy. This observation suggests that the inclusion of ResNet50 does not yield a significant enhancement in the model's generalization especially in the case of Top-10 highest accuracy class.

Top-20 Best Accuracy

Table 6.27 is the continuation of Top-10 best accuracy 6.26 presents the Top-20 classes with the highest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.27: Top-20 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Recreational Vehicle	100	92.8
12	Pug	100	72.0
13	Dalmatian	100	78.4
14	School Bus	100	77.6
15	Scoreboard	100	87.2
16	Great Dane	100	27.2
17	French Bulldog	100	40.8
18	Tibetan Mastiff	100	15.2
19	Arctic Fox	100	40.0
20	Radio Telescope	100	81.6

The Top-20 predictions achieve an impressive 100% average accuracy on original images. Yet, upon undergoing data augmentation, the model exhibits a significant decline in accuracy, pointing to a challenge in sustaining performance levels similar to those on the original images. This observation implies that the pre-trained model may lack robust generalization properties. In practical terms, the model's accuracy on augmented data is limited to 55.8%.

Top-50 Best Accuracy

Table 6.28 extends the Top-10 best accuracy from Table 6.26 and Top-20 best accuracy from Table 6.27 by presenting the Top-50 classes with the highest accuracy scores in the

original dataset. The table offers a comparison with their corresponding accuracy scores derived from augmented data performance.

Table 6.28: Top-50 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Convertible	100	60.0
22	Meerkat	100	60.8
23	Leafhopper	100	54.4
24	Cicada	100	36.0
25	Pool Table	100	93.6
26	Rhinoceros Beetle	100	36.8
27	Potter'S Wheel	100	84.8
28	Ladybug	100	87.2
29	Projector	100	83.2
30	Puck	100	55.2
31	Racer	100	76.8
32	Ice Bear	100	92.0
33	Quill	100	88.0
34	Brown Bear	100	88.0
35	Cheetah	100	75.2
36	Tiger	100	96.8
37	Lion	100	92.0
38	Snow Leopard	100	54.4
39	Screw	100	84.0
40	Hard Disc	100	71.2
41	Hamper	100	60.8
42	Toilet Seat	100	68.8
43	Tank	100	86.4
44	Thatch	100	50.4
45	Boston Bull	100	28.8
46	Dandie Dinmont	100	3.2
47	Theater Curtain	100	88.8
48	Airedale	100	2.4
49	Go-Kart	100	74.4
50	Torch	100	88.8

The Top-50 predictions achieve a remarkable 100% average accuracy on original images. However, when undergoing data augmentation, the model experiences a noticeable decrease in accuracy, indicating a challenge in maintaining performance levels comparable to those observed with original images. This observation raises questions about the pre-trained model's ability to generalize robustly. In practical terms, the model's accuracy on augmented data is only 62.8%.

Top-100 Best Accuracy

Table 6.29, which presents the Top-100 classes with the highest accuracy scores in the original dataset, serves as an extension of the Top-10 best accuracy from Table 6.26, Top-20 best accuracy from Table 6.27, and Top-50 highest accuracy from Table 6.28. The table enables a comparison of their accuracy scores derived from the performance on augmented data. The results highlight errors in the model's inference, signaling a challenge in effectively addressing corner cases due to insufficient model generalization. This challenge becomes apparent through the observed decline in accuracy when evaluating the model on augmented inputs.

Table 6.29: Top-100 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Gondola	100	85.6
52	Gong	100	51.2
53	Grand Piano	100	94.4
54	Hamper	100	60.8
55	Hard Disc	100	71.2
56	Harmonica	100	70.4
57	Honeycomb	100	87.2
58	Jeep	100	73.6
59	Jigsaw Puzzle	100	92.0
60	Jinrikisha	100	92.0
61	Knee Pad	100	59.2
62	Microwave	100	88.0
63	Missile	100	60.0
64	Model T	100	90.4
65	Mosquito Net	100	52.0
66	Mountain Bike	100	88.8

	Label	Original Accuracy %	Augmented Accuracy %
67	Moving Van	100	62.4
68	Oboe	100	48.0
69	Ocarina	100	53.6
70	Pool Table	100	93.6
71	Potter'S Wheel	100	84.8
72	Projector	100	83.2
73	Puck	100	55.2
74	Quill	100	88.0
75	Racer	100	76.8
76	Radio Telescope	100	81.6
77	Recreational Vehicle	100	92.8
78	Rotisserie	100	58.4
79	Rugby Ball	100	60.8
80	School Bus	100	77.6
81	Scoreboard	100	87.2
82	Screw	100	84.0
83	Slide Rule	100	42.4
84	Soccer Ball	100	91.2
85	Speedboat	100	86.4
86	Steam Locomotive	100	87.2
87	Stove	100	78.4
88	Streetcar	100	78.4
89	Stupa	100	77.6
90	Sundial	100	74.4
91	Swimming Trunks	100	64.0
92	Swing	100	89.6
93	Tank	100	86.4
94	Thatch	100	50.4
95	Theater Curtain	100	88.8
96	Toilet Seat	100	68.8
97	Torch	100	88.8
98	Trombone	100	72.8
99	Unicycle	100	92.8
100	Upright	100	85.6

The Top-100 predictions show that 100% average accuracy on original images. It shows that ResNet50 provide an improvement result compare to the ResNet18. Yet, as the model is tested into the data augmentation, there's a discernible dip in accuracy. This observation prompts contemplation regarding the pre-trained model's are not well generalization, the fact that the performance of accuracy is 63.8%

Top-10 Lowest Accuracy

Table 6.30 shows the Top-10 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.30: Top-10 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Garden Spider	20	5.6
2	Eskimo Dog	28	27.2
3	Soup Bowl	36	27.2
4	Weasel	36	32.0
5	Projectile	36	30.4
6	Cassette Player	36	16.0
7	Maillot	36	33.6
8	Ram	44	51.2
9	Green Lizard	44	56.0
10	Ruffed Grouse	44	1.6

The average accuracy for the Top-10 lowest predictions on the original images is 36.0%. Conversely, the average accuracy on the augmented data for the Top-10 lowest predictions, relative to the original image results, is 28.0%. Despite the lowest accuracy scenario, these findings indicate that the data augmentation results are closely aligned with those of the original data.

Top-20 Lowest Accuracy

Table 6.31 is the continuation of Top-10 lowest accuracy 6.30 presents the Top-20 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.31: Top-20 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Hog	48	46.4
12	Sandbar	48	30.4
13	Mushroom	48	51.2
14	Church	48	61.6
15	Collie	48	21.6
16	Tub	52	32.8
17	Sidewinder	52	8.0
18	Corn	52	26.4
19	Space Bar	56	9.6
20	Wolf Spider	56	36.8

The average accuracy for the Top-20 lowest predictions on the original images stands at 43.4%. The result predictions on the augmented data which its classes in the Top-10 lowest accuracy score, the average accuracy score is 31.6%.

Top-50 Lowest Accuracy

Table 6.32 extends the Top-10 lowest accuracy from Table 6.30 and Top-20 lowest accuracy from Table 6.31 by presenting the Top-50 classes with the lowest accuracy scores in the original dataset.

Table 6.32: Top-50 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Night Snake	56	15.2
22	Ice Cream	56	29.6
23	Dock	56	29.6
24	Sunglasses	56	46.4
25	Lampshade	56	36.0
26	Wine Bottle	56	47.2
27	English Foxhound	60	11.2
28	Cliff	60	31.2
29	Tiger Cat	60	29.6

	Label	Original Accuracy %	Augmented Accuracy %
30	American Staffordshire Terrier	60	11.2
31	Purse	64	22.4
32	Mortarboard	64	60.0
33	Laptop	64	26.4
34	Seashore	64	30.4
35	Typewriter Keyboard	64	40.8
36	Toy Poodle	64	19.2
37	Cricket	68	50.4
38	Handkerchief	68	24.0
39	Hook	68	22.4
40	Digital Clock	68	36.0
41	Groom	68	56.0
42	Plate	68	32.0
43	Green Snake	68	70.4
44	Computer Keyboard	68	67.2
45	Wok	68	46.4
46	Monitor	68	51.2
47	Tusker	68	38.4
48	Appenzeller	68	6.4
49	Cloak	68	37.6
50	Miniature Poodle	68	12.0

The average accuracy for the Top-50 lowest predictions on the original images is 55.6%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-50 lowest accuracy scores, yield an average accuracy score of 35.8%.

Top-100 Lowest Accuracy

Table 6.33, which presents the Top-100 classes with the lowest accuracy scores in the original dataset, serves as an extension of the Top-10 lowest accuracy from Table 6.30, Top-20 lowest accuracy from Table 6.31, and Top-50 lowest accuracy from Table 6.32.

Table 6.33: Top-100 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Paddle	68	48.8
52	Switch	68	20.0
53	Barrel	68	48.8
54	Radio	68	36.8
55	Analog Clock	68	44.0
56	Television	68	32.0
57	Notebook	68	61.6
58	Printer	68	60.8
59	Spider Monkey	72	30.4
60	Egyptian Cat	72	38.4
61	Australian Terrier	72	15.2
62	Silky Terrier	72	12.8
63	Cradle	72	45.6
64	Doberman	72	10.4
65	Cornet	72	38.4
66	Band Aid	72	22.4
67	Car Wheel	72	63.2
68	Cuirass	72	53.6
69	Black-Footed Ferret	72	14.4
70	Toilet Tissue	72	36.8
71	Grille	72	45.6
72	Coral Reef	72	60.0
73	Red Wine	72	42.4
74	Wool	72	30.4
75	Water Bottle	72	31.2
76	Wall Clock	72	61.6
77	Tape Player	72	28.0
78	Spatula	72	22.4
79	Sliding Door	72	61.6
80	Gown	72	52.0
81	Pitcher	72	72.0
82	Picket Fence	72	42.4
83	Passenger Car	72	38.4

	Label	Original Accuracy %	Augmented Accuracy %
84	Moped	72	44.0
85	Crane	72	38.4
86	Hoopskirt	72	37.6
87	Military Uniform	72	56.0
88	Maillot	72	46.4
89	Overskirt	76	21.6
90	Velvet	76	13.6
91	Lens Cap	76	32.8
92	Cardigan	76	14.4
93	Siberian Husky	76	16.8
94	Ipod	76	73.6
95	Water Jug	76	19.2
96	Wooden Spoon	76	43.2
97	Desk	76	58.4
98	Briard	76	9.6
99	English Springer	76	4.0
100	Comic Book	76	43.2

The average accuracy for the Top-100 lowest predictions on the original images is 63.9%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-100 lowest accuracy scores, yield an average accuracy score of 40.2%.

6.3.4. ResNet152 Performance

In this section provides the comparison performance between *Top K* classes which has highest and lowest accuracy on the original data and their counterparts in the augmented data. The inference result uses ResNet152 pre-trained model. This section additionally presents the outcomes of prediction for augmented instances, with respect to instances where the original data is predicted correctly, see Table 6.34

Table 6.34: Quantifying instances of augmented is true predicted when the original data is true.

True Predicted	Frequency %
0	21.90
1	9.43
2	8.05
3	8.56
4	10.89
5	41.17

Confusion Matrix

In this section, we present a comparative analysis of the confusion matrices, showcasing predictions on both original and augmented data in percentage terms. Tables 6.35 provides a detailed breakdown of outcomes for various classes within the broader category of bird species, mirroring the observations made with ResNet18 and ResNet152.

Table 6.35: Confusion matrix of original data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	100	0	0	0	0
brambling	0	100	0	0	0
goldfinch	0	0	100	0	0
house finch	0	0	0	100	0
junco	0	0	0	0	96

Table 6.36: Confusion matrix of augmented data.

	ostrich	brambling	goldfinch	house finch	junco
ostrich	91.2	0	0	0	0
brambling	0	56.8	11.2	2.4	0
goldfinch	0	4.8	58.4	0.8	0
house finch	0	12.8	20	19.2	2.4
junco	0	17.6	6.4	1.6	6.4

Table 6.37: Confusion matrix of original data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	100	0	0	0	0
robin	0	100	0	0	0
bulbul	0	0	100	0	0
jay	0	0	0	100	0
magpie	0	0	0	0	100

Table 6.38: Confusion matrix of augmented data.

	indigo bunting	robin	bulbul	jay	magpie
indigo bunting	53.6	0	0	8	0
robin	1.6	8.8	4.8	1.6	4.8
bulbul	0	3.2	28.8	11.2	2.4
jay	8	0	3.2	68.8	0.8
magpie	0	1.6	0	12.8	48

This observation points to a limitation in ResNet152’s ability to generalize effectively for bird species, with its performance not reaching the level seen in the original data. However, there is notable improvement compared to the performance of ResNet18 and ResNet50.

Top-10 Best Accuracy

Table 6.39 shows the Top-10 class which has the highest accuracy score which is represented as percentages and shows the data augmentation result with respect to the original data.

Table 6.39: Top-10 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Ashcan	100	69.6
2	Angora	100	76.0
3	Wood Rabbit	100	24.0
4	Sea Cucumber	100	43.2
5	Sea Urchin	100	44.0
6	Loafer	100	75.2
7	Hotdog	100	90.4
8	Sulphur Butterfly	100	46.4
9	Cabbage Butterfly	100	24.0
10	Monarch	100	73.6

The average accuracy for the Top-10 highest class predictions with ResNet152 on original images stands at an impressive 100%. However, when subjected to data augmentation, the model reveals a notable drop in accuracy, indicating a challenge in maintaining performance comparable to that on original images. This observation suggests that the pre-trained model may not possess robust generalization properties. In practice, the model's performance on augmented data is only 56.6% in accuracy. This observation suggests that the inclusion of ResNet152 improve the accuracy performance in the model's generalization especially in the case of Top-10 highest accuracy class compare to others pre-trained model.

Top-20 Best Accuracy

Table 6.40 is the continuation of Top-10 best accuracy 6.39 presents the Top-20 classes with the highest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.40: Top-20 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Ringlet	100	61.6
12	Admiral	100	39.2
13	Tray	100	77.6
14	Head Cabbage	100	80.0

	Label	Original Accuracy %	Augmented Accuracy %
15	Lacewing	100	42.4
16	Leafhopper	100	45.6
17	Trailer Truck	100	80.0
18	Loupe	100	47.2
19	Cockroach	100	54.4
20	Walking Stick	100	84.8

The Top-20 predictions achieve an impressive 100% average accuracy on original images. Yet, upon undergoing data augmentation, the model exhibits a significant decline in accuracy, pointing to a challenge in sustaining performance levels similar to those on the original images. This observation implies that the pre-trained model may lack robust generalization properties. In practical terms, the model's accuracy on augmented data is limited to 58.9%.

Top-50 Best Accuracy

Table 6.41 extends the Top-10 best accuracy from Table 6.39 and Top-20 best accuracy from Table 6.40 by presenting the Top-50 classes with the highest accuracy scores in the original dataset. The table offers a comparison with their corresponding accuracy scores derived from augmented data performance.

Table 6.41: Top-50 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Broccoli	100	85.6
22	Magnetic Compass	100	76.8
23	Mailbox	100	83.2
24	Toyshop	100	71.2
25	Fly	100	81.6
26	Weevil	100	18.4
27	Rhinoceros Beetle	100	44.0
28	Lipstick	100	81.6
29	Liner	100	83.2
30	Manhole Cover	100	64.0
31	Porcupine	100	83.2
32	Orangutan	100	71.2

	Label	Original Accuracy %	Augmented Accuracy %
33	Three-Toed Sloth	100	84.8
34	Armadillo	100	80.0
35	Badger	100	80.8
36	Skunk	100	56.0
37	Trifle	100	67.2
38	Ice Lolly	100	66.4
39	Arabian Camel	100	97.6
40	Library	100	76.8
41	Hartebeest	100	31.2
42	Ibex	100	72.0
43	Bighorn	100	37.6
44	Lighter	100	51.2
45	Bison	100	79.2
46	Bagel	100	63.2
47	Pretzel	100	74.4
48	Hippopotamus	100	82.4
49	Warthog	100	34.4
50	Wild Boar	100	60.0

The Top-50 predictions achieve a remarkable 100% average accuracy on original images. However, when undergoing data augmentation, the model experiences a noticeable decrease in accuracy, indicating a challenge in maintaining performance levels comparable to those observed with original images. This observation raises questions about the pre-trained model's ability to generalize robustly. In practical terms, the model's accuracy on augmented data is only 64.3%.

Top-100 Best Accuracy

Table 6.42, which presents the Top-100 classes with the highest accuracy scores in the original dataset, serves as an extension of the Top-10 best accuracy from Table 6.39, Top-20 best accuracy from Table 6.40, and Top-50 highest accuracy from Table 6.41. The table facilitates a comparison of their accuracy scores obtained from augmented data performance. The findings underscore errors in the model's inference, indicating a notable challenge in effectively handling corner cases due to insufficient model generalization. This challenge is evident in the observed decrease in accuracy when assessing the model using augmented inputs.

Table 6.42: Top-100 the highest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Hartebeest	100	31.2
52	Arabian Camel	100	97.6
53	Skunk	100	56.0
54	Badger	100	80.8
55	Armadillo	100	80.0
56	Three-Toed Sloth	100	84.8
57	Orangutan	100	71.2
58	Ashcan	100	69.6
59	Library	100	76.8
60	Lighter	100	51.2
61	Liner	100	83.2
62	Lipstick	100	81.6
63	Loafer	100	75.2
64	Loupe	100	47.2
65	Magnetic Compass	100	76.8
66	Mailbox	100	83.2
67	Manhole Cover	100	64.0
68	Marimba	100	80.0
69	Matchstick	100	55.2
70	Maypole	100	80.8
71	Model T	100	92.8
72	Mosque	100	91.2
73	Motor Scooter	100	78.4
74	Mountain Bike	100	86.4
75	Moving Van	100	76.0
76	Muzzle	100	52.0
77	Nail	100	43.2
78	Neck Brace	100	42.4
79	Nipple	100	56.0
80	Oboe	100	53.6
81	Toilet Seat	100	79.2
82	Toyshop	100	71.2
83	Trailer Truck	100	80.0

	Label	Original Accuracy %	Augmented Accuracy %
84	Tray	100	77.6
85	Washer	100	77.6
86	Trifle	100	67.2
87	Ice Lolly	100	66.4
88	Bagel	100	63.2
89	Pretzel	100	74.4
90	Cheeseburger	100	76.8
91	Hotdog	100	90.4
92	Head Cabbage	100	80.0
93	Broccoli	100	85.6
94	Zucchini	100	63.2
95	Pineapple	100	71.2
96	Jackfruit	100	35.2
97	Custard Apple	100	41.6
98	Pomegranate	100	63.2
99	Hay	100	91.2
100	Carbonara	100	52.8

The Top-100 predictions show that 100% average accuracy on original images. It shows that ResNet152 provide an improvement result compare to the ResNet18. Yet, as the model is tested into the data augmentation, there's a discernible dip in accuracy. This observation prompts contemplation regarding the pre-trained model's are not well generalization, the fact that the performance of accuracy is 63.5%

Top-10 Lowest Accuracy

Table 6.43 shows the Top-10 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.43: Top-10 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
1	Projectile	32	28.8
2	Weasel	40	27.2
3	Cassette Player	44	11.2

	Label	Original Accuracy %	Augmented Accuracy %
4	Hog	48	42.4
5	Garden Spider	48	20.8
6	Green Lizard	48	59.2
7	English Foxhound	52	11.2
8	Soup Bowl	52	37.6
9	Sidewinder	52	12.8
10	Eskimo Dog	52	48.8

The average accuracy for the Top-10 lowest predictions on the original images is 46.8%. Conversely, the average accuracy on the augmented data for the Top-10 lowest predictions, relative to the original image results, is 30.0%. Despite the lowest accuracy scenario, these findings indicate that the data augmentation results are closely aligned with those of the original data.

Top-20 Lowest Accuracy

Table 6.44 is the continuation of Top-10 lowest accuracy 6.43 presents the Top-20 classes with the lowest accuracy scores in the original dataset, comparing them to their corresponding accuracy scores derived from augmented data performance.

Table 6.44: Top-20 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
11	Ruffed Grouse	52	4.0
12	Maillot	56	32.8
13	Sandbar	60	33.6
14	Car Wheel	60	61.6
15	Space Bar	60	36.0
16	Tiger Cat	60	10.4
17	Wine Bottle	64	52.8
18	Tub	64	50.4
19	American Staffordshire Terrier	64	17.6
20	Ram	64	65.6

The average accuracy for the Top-20 lowest predictions on the original images stands at

53.6%. The result predictions on the augmented data which its classes in the Top-10 lowest accuracy score, the average accuracy score is 33.2%.

Top-50 Lowest Accuracy

Table 6.45 extends the Top-10 lowest accuracy from Table 6.43 and Top-20 lowest accuracy from Table 6.44 by presenting the Top-50 classes with the lowest accuracy scores in the original dataset. The table shows a comparison with the corresponding accuracy scores derived from augmented data performance. Providing the empiric result indicating a challenge in effectively addressing corner cases due to insufficient model generalization.

Table 6.45: Top-50 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
21	Grille	64	52.8
22	Typewriter Keyboard	68	47.2
23	Night Snake	68	17.6
24	Mushroom	68	69.6
25	Collie	68	40.0
26	Ice Cream	72	44.8
27	Crane	72	55.2
28	Patas	72	19.2
29	Corn	72	51.2
30	Dock	72	35.2
31	Passenger Car	72	71.2
32	Green Snake	72	60.8
33	Printer	72	65.6
34	Analog Clock	76	55.2
35	Computer Keyboard	76	68.0
36	Paddle	76	51.2
37	Radio	76	61.6
38	Seashore	76	36.0
39	Egyptian Cat	76	50.4
40	Gown	76	63.2
41	Moped	76	59.2
42	Squirrel Monkey	76	0.8
43	Maillot	76	45.6

	Label	Original Accuracy %	Augmented Accuracy %
44	Cricket	76	59.2
45	Desktop Computer	76	67.2
46	Toy Poodle	76	35.2
47	Breastplate	76	39.2
48	Lampshade	76	55.2
49	Hook	80	48.0
50	Cloak	80	42.4

The average accuracy for the Top-50 lowest predictions on the original images is 65.6%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-50 lowest accuracy scores, yield an average accuracy score of 42.6%.

Top-100 Lowest Accuracy

Table 6.46, which presents the Top-100 classes with the lowest accuracy scores in the original dataset, serves as an extension of the Top-10 lowest accuracy from Table 6.43, Top-20 lowest accuracy from Table 6.44, and Top-50 lowest accuracy from Table 6.45. The table facilitates a comparison with their corresponding accuracy scores derived from augmented data performance.

Table 6.46: Top-100 the lowest accuracy of original data and its data augmentation.

	Label	Original Accuracy %	Augmented Accuracy %
51	Tusker	80	42.4
52	Cliff Dwelling	80	39.2
53	Letter Opener	80	47.2
54	Redbone	80	20.8
55	Mouse	80	70.4
56	Church	80	75.2
57	Water Bottle	80	48.8
58	Military Uniform	80	56.8
59	Laptop	80	42.4
60	Cliff	80	55.2
61	Coffeepot	80	64.8
62	Purse	80	51.2
63	Spider Web	80	78.4

	Label	Original Accuracy %	Augmented Accuracy %
64	Japanese Spaniel	80	34.4
65	Stage	80	60.0
66	Screen	80	36.0
67	Titi	80	1.6
68	Cock	80	83.2
69	Leaf Beetle	80	60.8
70	Television	80	47.2
71	Space Heater	84	48.0
72	Otterhound	84	12.8
73	Monitor	84	56.0
74	Dungeness Crab	84	40.8
75	Wallet	84	65.6
76	Sports Car	84	77.6
77	Park Bench	84	70.4
78	Palace	84	73.6
79	Leopard	84	73.6
80	Safe	84	57.6
81	Miniature Poodle	84	12.0
82	Rifle	84	61.6
83	Quilt	84	69.6
84	Tabby	84	46.4
85	Appenzeller	84	16.8
86	Catamaran	84	53.6
87	Power Drill	84	64.0
88	Red Wolf	84	4.0
89	Switch	84	41.6
90	Doberman	84	20.0
91	Standard Poodle	84	48.8
92	Tape Player	84	46.4
93	King Crab	84	45.6
94	Loudspeaker	84	72.8
95	Barbershop	84	44.8
96	Artichoke	84	60.8
97	Spider Monkey	84	27.2
98	Barrel	84	52.0

	Label	Original Accuracy %	Augmented Accuracy %
99	Guenon	84	34.4
100	Ipod	84	86.4

The average accuracy for the Top-100 lowest predictions on the original images is 74.0%. In contrast, the predictions on the augmented data, specifically for the classes within the Top-100 lowest accuracy scores, yield an average accuracy score of 46.3%.

6.3.5. Fine-tune Pre-trained Model

Refining the performance of a pre-trained model through retraining process enhance model performance, especially through augmentation, it strengthen the model generalization. Table 6.47 presents a testing comparison between the pre-trained ResNet18 model and the ResNet18 model after fine-tuning on the original data and augmented data with 90 epochs which requires a training duration of 4.5 hours.

Table 6.47: Fine-tune pre-trained model performance.

Epoch	Pre-trained		Fine-tune	
	Original Data %	Augmented Data %	Original Data %	Augmented Data %
10	79.06	47.46	99.05	99.67
20	79.06	47.46	99.10	99.64
50	79.06	47.46	99.40	99.79
90	79.06	47.46	99.45	99.80
Average	79.06	47.46	99.25	99.73

Through the fine-tuning process, the model exhibits a substantial average improvement of 20.19% on the original data and 52.27% on the augmented data. This outcome highlights the model's adeptness and enhance model generalization.

6.3.6. SHIFT Result

This section present the testing performance and showing the faulty behavior related to autonomous driving. Figure 6.1 shows the prediction results through a normalized percentage confusion matrix for each label in order to provide a clear representation of faulty behaviors. The confusion matrix is presented with percentages that are rounded for clarity. As a result, the sum of the percentages in each row may not be exactly equal to 100% but an approximation. True Label is the ground truth of the label, on the other hand, Predicted Label is the result of prediction. For example, there is 78% Building that is truly predicted as Building, and there is 8% misclassified Building as Sky (second row in the confusion matrix).

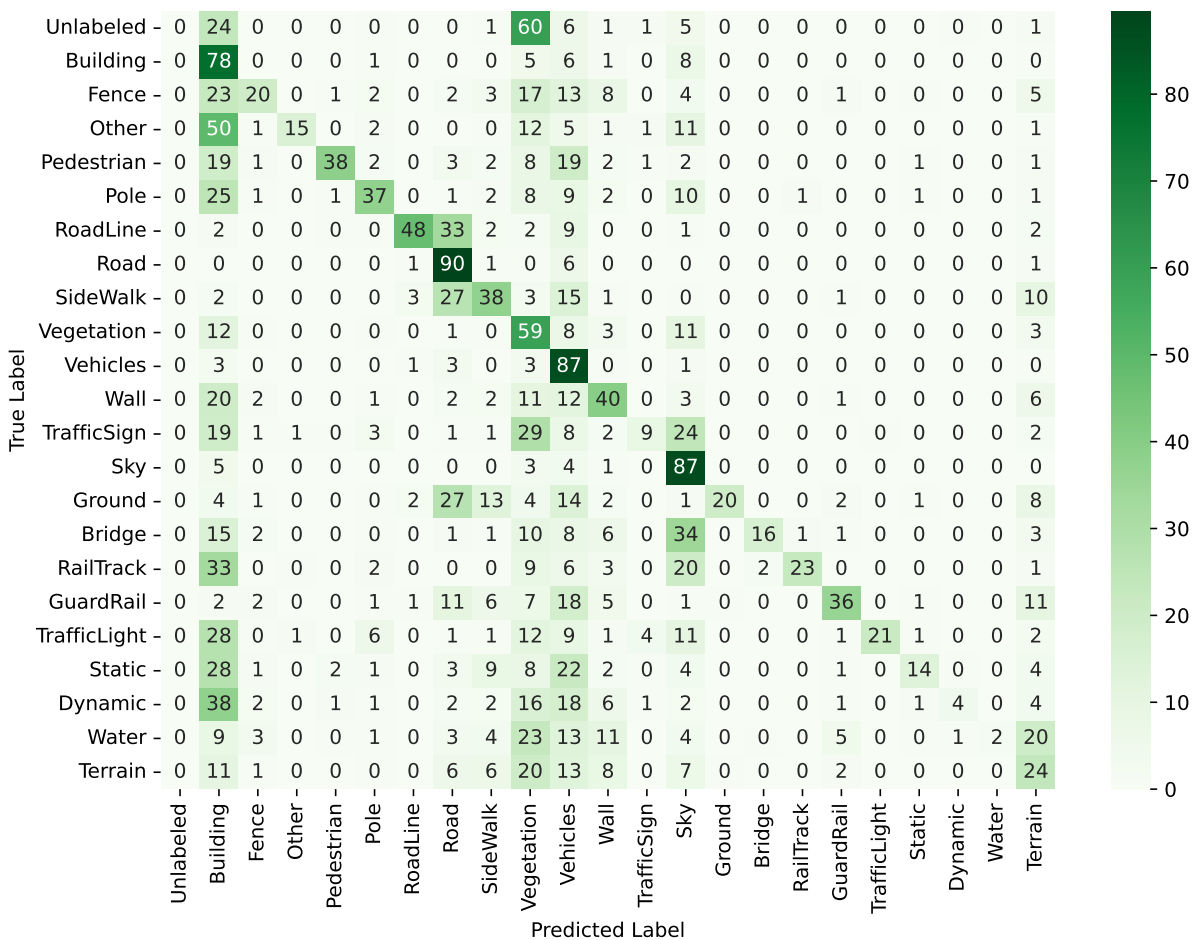


Figure 6.1: Confusion Matrix (% of pixels of a certain class classified as another one).

The fifth row labeled Pedestrian shows a very dangerous misclassification that might lead to fatal accidents, 38% of the pedestrian pixels were correctly classified as Pedestrian, but 3% of them were misclassified as Road. The consequences of a Pedestrian being

misclassified as Road can be severe. Even a low-speed collision with a pedestrian can cause serious injuries or even death.

The result also shows the misclassification example of Water misclassified as Vegetation 23% and Terrain 20% introduces alarming possibilities of dangerous scenarios. This case depicts a situation where an area of Water, such as standing water on the road is inaccurately labeled as either Vegetation or Terrain. This misclassification poses serious risks, ranging from incorrect navigation decisions to challenges in obstacle avoidance. For instance, if Water is mistakenly perceived as Terrain or Vegetation, an autonomous driving might erroneously attempt to navigate through a Water body, while a car might still be physically capable of traversing Water on the road, which is not possible if Terrain or Vegetation. This scenario leading to potential accidents and safety hazards.

The result also shows misclassification arises with the Ground label, encompassing horizontal ground-level structures shared by both vehicles and pedestrians, as well as flat roundabouts delimited from the road by a curb. The instances of labeled as Ground are most mistakenly classified as Road by 27%. Misclassifying a flat roundabout as part of the road could result in the autonomous driving misunderstanding the intended path and potentially navigating inappropriately. Similarly, horizontal ground-level structures shared by vehicles and pedestrians, if mislabeled as Road might lead to the autonomous driving is misjudging the usage of the space and making decisions that compromise safety.

This evaluation presents Metamorphic Testing with DILLEMA data augmentation results of more than 2 billion misclassified pixels that reflect the faulty behavior in the autonomous driving scenario.

7 | Conclusion and Future Works

This chapter is dedicated to present the conclusive remarks based on the outcomes of our study and delving into promising directions for future research.

7.1. Conclusion

In conclusion, the proposed DILLEMA framework represents a step forward in addressing the challenges associated with corner cases of deep learning model with Metamorphic Testing. By leveraging the power of Large Language Model and incorporating Diffusion Models, DILLEMA aims to generate test cases that more accurately reflect the intricacies of real-world environments. As the field of deep learning application continues to advance, the findings and proposed methodologies presented in this thesis contribute to the ongoing discourse on ensuring the robustness of deep learning systems.

7.2. Future Works

1. DILLEMA implementation with different Captioning Model, Large Language Model, and Diffusion Model, as there are various potential alternative implementations to explore.
2. Enhancing DILLEMA robustness through dataset incorporation. Task that specified through DILLEMA process requires specialized handling, for instance ImageNet1K should know the label of the dataset in order to have good result. Hence, towards refining the robustness of DILLEMA by exploring methodologies to enhance the incorporation of diverse datasets it improve comprehensive and effective testing framework.
3. Acceleration of test case generation. The essential aspect of DILLEMA's efficacy lies in the speed at which it generates data augmentation. Future research can focus on optimizing and accelerating the generation process.

Bibliography

- [1] D. Adiwardana, M. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le. Towards a human-like open-domain chatbot. *CoRR*, abs/2001.09977, 2020. URL <https://arxiv.org/abs/2001.09977>.
- [2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL <http://arxiv.org/abs/1604.07316>.
- [3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*, pages 177–186. Physica-Verlag, 2010. doi: 10.1007/978-3-7908-2604-3_16. URL https://doi.org/10.1007/978-3-7908-2604-3_16.
- [4] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL <https://openreview.net/forum?id=B1xsqj09Fm>.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>.

- [6] P. Chandra and Y. Singh. An activation function adapting training algorithm for sigmoidal feedforward networks. *Neurocomputing*, 61:429–437, 2004. doi: 10.1016/J.NEUCOM.2004.04.001. URL <https://doi.org/10.1016/j.neucom.2004.04.001>.
- [7] T. Y. Chen, F. Kuo, H. Liu, P. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Comput. Surv.*, 51(1):4:1–4:27, 2018. doi: 10.1145/3143561. URL <https://doi.org/10.1145/3143561>.
- [8] T. Y. Chen, S. C. Cheung, and S. Yiu. Metamorphic testing: A new approach for generating next test cases. *CoRR*, abs/2002.12543, 2020. URL <https://arxiv.org/abs/2002.12543>.
- [9] J. Ding, X. Kang, and X. Hu. Validating a deep learning framework by metamorphic testing. In *2nd IEEE/ACM International Workshop on Metamorphic Testing, MET@ICSE 2017, Buenos Aires, Argentina, May 22, 2017*, pages 28–34. IEEE Computer Society, 2017. doi: 10.1109/MET.2017.2. URL <https://doi.org/10.1109/MET.2017.2>.
- [10] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503:92–108, 2022. doi: 10.1016/J.NEUCOM.2022.06.111. URL <https://doi.org/10.1016/j.neucom.2022.06.111>.
- [11] L. Dunlap, A. Umino, H. Zhang, J. Yang, J. E. Gonzalez, and T. Darrell. Diversify your vision datasets with automatic diffusion-based augmentation. *CoRR*, abs/2305.16289, 2023. doi: 10.48550/arXiv.2305.16289. URL <https://doi.org/10.48550/arXiv.2305.16289>.
- [12] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In F. Tip and E. Bodden, editors, *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, Amsterdam, The Netherlands, July 16-21, 2018*, pages 118–128. ACM, 2018. doi: 10.1145/3213846.3213858. URL <https://doi.org/10.1145/3213846.3213858>.
- [13] M. A. Fahmideh and M. E. Scheurer. Pediatric brain tumors: Descriptive epidemiology, risk factors, and future directions. *Cancer Epidemiology, Biomarkers & Prevention*, 30:813 – 821, 2021. URL <https://api.semanticscholar.org/CorpusID:232104225>.
- [14] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In

- G. J. Gordon, D. B. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 315–323. JMLR.org, 2011. URL <http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf>.
- [15] P. Godefroid, A. Kiezun, and M. Y. Levin. Grammar-based whitebox fuzzing. In R. Gupta and S. P. Amarasinghe, editors, *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 206–215. ACM, 2008. doi: 10.1145/1375581.1375607. URL <https://doi.org/10.1145/1375581.1375607>.
- [16] P. Godefroid, M. Y. Levin, and D. A. Molnar. SAGE: whitebox fuzzing for security testing. *Commun. ACM*, 55(3):40–44, 2012. doi: 10.1145/2093548.2093564. URL <https://doi.org/10.1145/2093548.2093564>.
- [17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society. doi: 10.1109/CVPR.2016.90. URL <https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90>.
- [18] J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning - the MIT press, 2016, 800 pp, ISBN: 0262035618. *Genet. Program. Evolvable Mach.*, 19(1-2):305–307, 2018. doi: 10.1007/s10710-017-9314-z. URL <https://doi.org/10.1007/s10710-017-9314-z>.
- [19] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS’20, Red Hook, NY, USA, 2020*. Curran Associates Inc. ISBN 9781713829546.
- [20] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6(2):107–116, 1998. doi: 10.1142/S0218488598000094. URL <https://doi.org/10.1142/S0218488598000094>.
- [21] R. Hussain and S. Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Commun. Surv. Tutorials*, 21(2):1275–1313, 2019. doi: 10.1109/COMST.2018.2869360. URL <https://doi.org/10.1109/COMST.2018.2869360>.
- [22] Y. Jia and M. Harman. An analysis and survey of the development of mutation

- testing. *IEEE Trans. Software Eng.*, 37(5):649–678, 2011. doi: 10.1109/TSE.2010.62. URL <https://doi.org/10.1109/TSE.2010.62>.
- [23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL <http://arxiv.org/abs/1412.6980>.
- [24] J. Konečný, J. Liu, P. Richtárik, and M. Takác. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE J. Sel. Top. Signal Process.*, 10(2):242–255, 2016. doi: 10.1109/JSTSP.2015.2505682. URL <https://doi.org/10.1109/JSTSP.2015.2505682>.
- [25] S. Kong and M. Takatsuka. Hexpo: A vanishing-proof activation function. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 2562–2567. IEEE, 2017. doi: 10.1109/IJCNN.2017.7966168. URL <https://doi.org/10.1109/IJCNN.2017.7966168>.
- [26] R. Kuhn, Y. Lei, and R. Kacker. Practical combinatorial testing: Beyond pairwise. *IT Prof.*, 10(3):19–23, 2008. doi: 10.1109/MITP.2008.54. URL <https://doi.org/10.1109/MITP.2008.54>.
- [27] G. Kwon and J. C. Ye. One-shot adaptation of GAN in just one CLIP. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(10):12179–12191, 2023. doi: 10.1109/TPAMI.2023.3283551. URL <https://doi.org/10.1109/TPAMI.2023.3283551>.
- [28] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404. Morgan Kaufmann, 1989. URL <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network>.
- [29] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nat.*, 521(7553):436–444, 2015. doi: 10.1038/NATURE14539. URL <https://doi.org/10.1038/nature14539>.
- [30] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168, 2011. doi: 10.1109/IVS.2011.5940562.

- [31] J. Li, D. Li, C. Xiong, and S. C. H. Hoi. BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 12888–12900. PMLR, 2022. URL <https://proceedings.mlr.press/v162/li22n.html>.
- [32] J. Li, D. Li, S. Savarese, and S. C. H. Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742. PMLR, 2023. URL <https://proceedings.mlr.press/v202/li23q.html>.
- [33] M. Lin, Q. Chen, and S. Yan. Network in network. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL <http://arxiv.org/abs/1312.4400>.
- [34] H. Liu, F. Kuo, D. Towey, and T. Y. Chen. How effectively does metamorphic testing alleviate the oracle problem? *IEEE Trans. Software Eng.*, 40(1):4–22, 2014. doi: 10.1109/TSE.2013.46. URL <https://doi.org/10.1109/TSE.2013.46>.
- [35] D. Lo, S. Khoo, and C. Liu. Efficient mining of iterative patterns for software specification discovery. In P. Berkhin, R. Caruana, and X. Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 460–469. ACM, 2007. doi: 10.1145/1281192.1281243. URL <https://doi.org/10.1145/1281192.1281243>.
- [36] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. doi: 10.1109/CVPR.2015.7298965. URL <https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298965>.
- [37] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang. Deepmutation: Mutation testing of deep learning systems. In S. Ghosh, R. Natella, B. Cukic, R. S. Poston, and N. Laranjeiro, editors, *29th IEEE International Symposium on Software Reliability Engineering, ISSRE 2018, Memphis,*

- TN, USA, October 15-18, 2018*, pages 100–111. IEEE Computer Society, 2018. doi: 10.1109/ISSRE.2018.00021. URL <https://doi.org/10.1109/ISSRE.2018.00021>.
- [38] T. M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN 978-0-07-042807-2. URL <https://www.worldcat.org/oclc/61321007>.
- [39] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010. URL <https://icml.cc/Conferences/2010/papers/432.pdf>.
- [40] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Comput. Surv.*, 43(2): 11:1–11:29, 2011. doi: 10.1145/1883612.1883618. URL <https://doi.org/10.1145/1883612.1883618>.
- [41] A. N. S. Njikam and H. Zhao. A novel activation function for multilayer feed-forward neural networks. *Appl. Intell.*, 45(1):75–82, 2016. doi: 10.1007/S10489-015-0744-0. URL <https://doi.org/10.1007/s10489-015-0744-0>.
- [42] OpenAI. Chatgpt: A large-scale transformer-based language model for conversation, 2021. URL <https://openai.com/research/chatgpt>.
- [43] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. L. Traon, and M. Harman. Chapter six - mutation testing advances: An analysis and survey. *Adv. Comput.*, 112:275–378, 2019. doi: 10.1016/BS.ADCOM.2018.03.015. URL <https://doi.org/10.1016/bs.adcom.2018.03.015>.
- [44] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org, 2013. URL <http://proceedings.mlr.press/v28/pascanu13.html>.
- [45] K. Pei, Y. Cao, J. Yang, and S. Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 1–18. ACM, 2017. doi: 10.1145/3132747.3132785. URL <https://doi.org/10.1145/3132747.3132785>.
- [46] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pretraining. Technical report, OpenAI, 2018. URL

https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

- [47] M. A. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993*, pages 586–591. IEEE, 1993. doi: 10.1109/ICNN.1993.298623. URL <https://doi.org/10.1109/ICNN.1993.298623>.
- [48] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y. Boureau, and J. Weston. Recipes for building an open-domain chatbot. In P. Merlo, J. Tiedemann, and R. Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 300–325. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EACL-MAIN.24. URL <https://doi.org/10.18653/v1/2021.eacl-main.24>.
- [49] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01042. URL <https://doi.org/10.1109/CVPR52688.2022.01042>.
- [50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. doi: 10.1007/S11263-015-0816-Y. URL <https://doi.org/10.1007/s11263-015-0816-y>.
- [51] S. Segura, G. Fraser, A. B. Sánchez, and A. R. Cortés. A survey on metamorphic testing. *IEEE Trans. Software Eng.*, 42(9):805–824, 2016. doi: 10.1109/TSE.2016.2532875. URL <https://doi.org/10.1109/TSE.2016.2532875>.
- [52] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen. Metamorphic testing: Testing the untestable. *IEEE Softw.*, 37(3):46–53, 2020. doi: 10.1109/MS.2018.2875968. URL <https://doi.org/10.1109/MS.2018.2875968>.
- [53] S. Squartini, A. Hussain, and F. Piazza. Preprocessing based solution for the vanishing gradient problem in recurrent neural networks. In *Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS 2003, Bangkok, Thailand, May 25-28, 2003*, pages 713–716. IEEE, 2003. doi: 10.1109/ISCAS.2003.1206412. URL <https://doi.org/10.1109/ISCAS.2003.1206412>.

- [54] T. Sun, M. Segù, J. Postels, Y. Wang, L. V. Gool, B. Schiele, F. Tombari, and F. Yu. SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 21339–21350. IEEE, 2022. doi: 10.1109/CVPR52688.2022.02068. URL <https://doi.org/10.1109/CVPR52688.2022.02068>.
- [55] Y. Tian, K. Pei, S. Jana, and B. Ray. Deeptest: automated testing of deep-neural-network-driven autonomous cars. In M. Chaudron, I. Crnkovic, M. Chechik, and M. Harman, editors, *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, pages 303–314. ACM, 2018. doi: 10.1145/3180155.3180220. URL <https://doi.org/10.1145/3180155.3180220>.
- [56] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971. URL <https://doi.org/10.48550/arXiv.2302.13971>.
- [57] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL <https://doi.org/10.48550/arXiv.2307.09288>.
- [58] P. Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule. *SIAM J. Optim.*, 8(2):506–531, 1998. doi: 10.1137/S1052623495294797. URL <https://doi.org/10.1137/S1052623495294797>.
- [59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Ad-*

- vances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [60] J. Vendrow, S. Jain, L. Engstrom, and A. Madry. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *CoRR*, abs/2302.07865, 2023. doi: 10.48550/arXiv.2302.07865. URL <https://doi.org/10.48550/arXiv.2302.07865>.
- [61] J. Wang and B. Malakooti. A feedforward neural network for multiple criteria decision making. *Comput. Oper. Res.*, 19(2):151–167, 1992. doi: 10.1016/0305-0548(92)90089-N. URL [https://doi.org/10.1016/0305-0548\(92\)90089-N](https://doi.org/10.1016/0305-0548(92)90089-N).
- [62] S. Wang and Z. Su. Metamorphic object insertion for testing object detection systems. In *35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020, Melbourne, Australia, September 21-25, 2020*, pages 1053–1065. IEEE, 2020. doi: 10.1145/3324884.3416584. URL <https://doi.org/10.1145/3324884.3416584>.
- [63] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In Q. Liu and D. Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-DEMOS.6. URL <https://doi.org/10.18653/v1/2020.emnlp-demos.6>.
- [64] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In D. Zhang and A. Møller, editors, *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, pages 146–157. ACM, 2019. doi: 10.1145/3293882.3330579. URL <https://doi.org/10.1145/3293882.3330579>.
- [65] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Trans. Software Eng.*, 48(2):1–36, 2022. doi: 10.1109/TSE.2019.2962027. URL <https://doi.org/10.1109/TSE.2019.2962027>.
- [66] L. Zhang and M. Agrawala. Adding conditional control to text-to-image diffusion

- models. *CoRR*, abs/2302.05543, 2023. doi: 10.48550/arXiv.2302.05543. URL <https://doi.org/10.48550/arXiv.2302.05543>.
- [67] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142, 2018. doi: 10.1145/3238147.3238187.
- [68] Q. Zhang, Y. N. Wu, and S. Zhu. Interpretable convolutional neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8827–8836. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00920. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Interpretable_Convolutional_Neural_CVPR_2018_paper.html.
- [69] Z. Zhang, P. Wang, H. Guo, Z. Wang, Y. Zhou, and Z. Huang. Deepbackground: Metamorphic testing for deep-learning-driven image recognition systems accompanied by background-relevance. *Inf. Softw. Technol.*, 140:106701, 2021. doi: 10.1016/J.INFSOF.2021.106701. URL <https://doi.org/10.1016/j.infsof.2021.106701>.
- [70] Z. Zhang, Y. Liu, C. Han, T. Guo, T. Yao, and T. Mei. Generalized one-shot domain adaptation of generative adversarial networks. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/58ce6a4b9c16d11975f11e4a23871041-Abstract-Conference.html.
- [71] Z. Zhou, S. Xiang, and T. Y. Chen. Metamorphic testing for software quality assessment: A study of search engines. *IEEE Trans. Software Eng.*, 42(3):264–284, 2016. doi: 10.1109/TSE.2015.2478001. URL <https://doi.org/10.1109/TSE.2015.2478001>.
- [72] Z. Q. Zhou and L. Sun. Metamorphic testing of driverless cars. *Commun. ACM*, 62(3):61–67, 2019. doi: 10.1145/3241979. URL <https://doi.org/10.1145/3241979>.

List of Figures

2.1	<i>Blue</i> arrow is the expected steering in original images and red arrow is the steering result with six-degree rotation [55].	6
2.2	GAN based generated images, snowy and rainy image [67].	7
2.3	Left image is the original input image that correctly makes a decision result. The right image is the generated image which makes an error with respect to the expected result [45].	7
2.4	Left image is "A photo of <i>< dog – class ></i> in the beach". The right image is "A photo of <i>< dog – class ></i> in the grass" [60].	8
2.5	ALIA augmented image result and the original image [11]	9
3.1	Testing process.	11
3.2	The perceptron.	13
3.3	Feed Forward Neural Network for a (L)-hidden layer with i input units and K output units. w_{ji} is the weight going to neuron j -th from input or neuron i -th.	14
3.4	The architecture of the original convolutional neural network, as introduced by <i>LeCun et al.</i> [28].	17
3.5	Semantic segmentation with Fully-Convolutional Network [36].	20
3.6	Fully Convolutional Network with 8 times upsampled.	21
3.7	U-Net Architecture, the orange layer is convolution, the red color is max-pooling, and the blue color is convolutionl transpose.	21
3.8	The Transformer Encoder-Decoder architecture [59].	23
3.9	Latent Diffusion Model.	25
3.10	Controlling Diffusion Model.	26
3.11	Controlling Image with spatial context.	27
4.1	DILLEMA schema.	29
4.2	Input image for caption generation.	30
4.3	Controlling Image with spatial context conversion possibilities.	34
5.1	ControlNet with Canny spatial context conversion.	40

5.2	ControlNet with Depth-map spatial context conversion.	40
5.3	ControlNet with Segmentation-map spatial context conversion.	41
5.4	ControlNet with Scribble spatial context conversion.	41
5.5	ControlNet with HED spatial context conversion.	41
6.1	Confusion Matrix (% of pixels of a certain class classified as another one). .	90

List of Tables

- 3.1 Non-linear activation functions. 14
- 6.1 DILLEMA images augmentation result on ImageNet1K. 45
- 6.2 DILLEMA images augmentation result on SHIFT. 51
- 6.3 Accuracy score. 53
- 6.4 Precision score. 53
- 6.5 Recall score. 53
- 6.6 F1 score. 54
- 6.7 Faulty behavior on ImageNet1K. 54
- 6.8 Quantifying instances of augmented is true predicted when the original data is true. 55
- 6.9 Confusion matrix of original data. 55
- 6.10 Confusion matrix of augmented data. 55
- 6.11 Confusion matrix of original data. 56
- 6.12 Confusion matrix of augmented data. 56
- 6.13 Top-10 the highest accuracy of original data and its data augmentation. . . 56
- 6.14 Top-20 the highest accuracy of original data and its data augmentation. . . 57
- 6.15 Top-50 the highest accuracy of original data and its data augmentation. . . 58
- 6.16 Top-100 the highest accuracy of original data and its data augmentation. . . 60
- 6.17 Top-10 the lowest accuracy of original data and its data augmentation. . . 61
- 6.18 Top-20 the lowest accuracy of original data and its data augmentation. . . 62
- 6.19 Top-50 the lowest accuracy of original data and its data augmentation. . . 63
- 6.20 Top-100 the lowest accuracy of original data and its data augmentation. . . 64
- 6.21 Quantifying instances of augmented is true predicted when the original data is true. 66
- 6.22 Confusion matrix of original data. 67
- 6.23 Confusion matrix of augmented data. 67
- 6.24 Confusion matrix of original data. 67
- 6.25 Confusion matrix of augmented data. 68
- 6.26 Top-10 the highest accuracy of original data and its data augmentation. . . 68

6.27	Top-20 the highest accuracy of original data and its data augmentation. . .	69
6.28	Top-50 the highest accuracy of original data and its data augmentation. . .	70
6.29	Top-100 the highest accuracy of original data and its data augmentation. . .	71
6.30	Top-10 the lowest accuracy of original data and its data augmentation. . .	73
6.31	Top-20 the lowest accuracy of original data and its data augmentation. . .	74
6.32	Top-50 the lowest accuracy of original data and its data augmentation. . .	74
6.33	Top-100 the lowest accuracy of original data and its data augmentation. . .	76
6.34	Quantifying instances of augmented is true predicted when the original data is true.	78
6.35	Confusion matrix of original data.	78
6.36	Confusion matrix of augmented data.	79
6.37	Confusion matrix of original data.	79
6.38	Confusion matrix of augmented data.	79
6.39	Top-10 the highest accuracy of original data and its data augmentation. . .	80
6.40	Top-20 the highest accuracy of original data and its data augmentation. . .	80
6.41	Top-50 the highest accuracy of original data and its data augmentation. . .	81
6.42	Top-100 the highest accuracy of original data and its data augmentation. . .	83
6.43	Top-10 the lowest accuracy of original data and its data augmentation. . .	84
6.44	Top-20 the lowest accuracy of original data and its data augmentation. . .	85
6.45	Top-50 the lowest accuracy of original data and its data augmentation. . .	86
6.46	Top-100 the lowest accuracy of original data and its data augmentation. . .	87
6.47	Fine-tune pre-trained model performance.	89

Acknowledgements

I extend my heartfelt gratitude to Prof. Luciano Baresi for providing me with the invaluable opportunity to complete my thesis under his expert guidance and supervision.

A special acknowledgment goes to Davide, my co-supervisor and the best person I have ever met at Politecnico. His expertise and collaborative spirit have been indispensable to the success of this thesis. I am truly thankful for his guidance and unwavering support.

I would like to express my deepest gratitude to my mother and father for their unwavering support, encouragement, and sacrifices throughout my academic journey. Their pray and belief in my potential have been a source of my life.

To my beloved partner Rosiana, your love, your love, understanding, and uplifting encouragement have stood as unwavering pillars of strength during this academic odyssey. Your enduring support has not only accompanied me through every step but has also bestowed a profound sense of purpose upon each milestone, transforming this journey into a meaningful and enriching experience.

I am profoundly grateful to the Indonesian Endowment Fund for Education (LPDP) for the invaluable support and scholarship that made it possible for me to pursue and complete my master's studies.

I would also like to express my sincere appreciation to Ardi, Arif, Fariz, Adib, Labiyb, Bayu, Fian, and all my friends, whose names I regrettably cannot include here. Your friendship, encouragement, and shared experiences have profoundly enriched my academic and personal life.

This thesis is a culmination of the collective efforts and support from these remarkable individuals, and I am genuinely grateful for their contributions to this significant milestone in my academic journey.

