修 士 論 文

Generative Semantic Augmentation Based on Depth-Aware Rendering for Low-Data Image Classification

九州工業大学　大学院情報工学府
情報創成工学専攻　情報・通信工学分野

Alexander Gerald Weismann

2025年度

指導教員：Mario Köppen

Abstract

Image classification remains a cornerstone of computer vision, yet traditional augmentation techniques often fall short in low-data environments and struggle to keep pace with the demands of increasingly complex model architectures. In this thesis we investigate the use of generative AI (specifically Generative Adversarial Networks (GANs) and Diffusion Models) to enhance image dataset augmentation. We introduce a novel pipeline that integrates image depth estimation, 3D modelling workflows, and synthetic image generation to produce enriched training data. Our experiments on a dataset of an inital 63 images demonstrates that combining this synthetic data with classical augmentations improves classification accuracy by over 25% in low-data settings. Evaluation of the generated data yields an average LPIPS (Learned Perceptual Image Patch Similarity) score of approximately 0.58 per class, indicating successfully diverse enrichment. Furthermore, CLIP similarity scores average around 0.87 per class, confirming strong semantic alignment with the original class identities. These results highlight the promise of generative AI as a powerful tool for enhancing datasets low-data scenarios.

Contents

List of Abbreviations

API	Application Program Interface
CLIP	Contrastive Language-Image Pre-training
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
HDRI	High Dynamic Range Image
IDE	Integrated Development Environment
LPIPS	Learned Perceptual Image Patch Similarity
SD	Stable Diffusion
UI	User Interface
VAE	Variable Autoencoder
W&B	Weights & Biases

Chapter 1

Introduction

The field of image classification has continuously evolved, progressing from simple rule-based methods in the 1950s to today's sophisticated machine learning architectures. Early image classification systems relied on template matching, where pixel data from an input image was compared to predefined templates. These systems used straightforward heuristics, analyzing edges, shapes, textures, and colours. While basic by modern standards, these techniques laid the groundwork for today's advanced AI models.

Dataset augmentation techniques also have a long history, with early methods like rotation, scaling, translation, flipping, and colour space shifting being widely used to artificially expand datasets. These approaches remain fundamental, but their limitations become apparent in the context of modern machine learning. As models grow in size and complexity, so too does the demand for expansive and diverse training data. This has placed a significant burden on data collection, making it increasingly difficult for individual researchers or small organizations to compete in developing state-of-the-art computer vision models. Or, simply utilize the technology for themselves in more niche scenarios.

Enter Generative Adversarial Networks (GANs). A transformative development in synthetic data generation. First introduced by Ian Goodfellow, GANs have demonstrated a unique ability to generate realistic synthetic images that mirror the distribution of an original dataset while introducing subtle variations. This capability has proven invaluable for augmenting datasets, particularly in scenarios where data scarcity or sensitivity presents a challenge.

Early experiments with GAN-based augmentation techniques underscore their potential. For example, a study of facial emotion datasets showed that GANs could improve classification accuracy by up to 10% by generating diverse facial expressions [34]. Similarly, GANs have been applied in medical imaging, with studies demonstrating improvements in classification accuracy for liver lesion datasets [11] and brain MRI scans [3]. These studies highlight how GAN-generated data can complement traditional augmentation methods to improve model performance, particularly in data-scarce domains. Recent advancements have extended the frontier of genera-

tive AI well beyond GANs, with diffusion models emerging as a powerful alternative. Diffusion models generate images through an iterative denoising process that allows for greater control, stability, and quality. Models like *DALL-E*, *Imagen*, and *Stable Diffusion* have showcased the ability to synthesize photorealistic and semantically accurate images from natural language prompts.

Building on these advancements, this thesis explores the use of synthetic data to augment computer vision models in scenarios beyond the medical field. Specifically, it addresses the question: How can an individual effectively create sufficient data to train and test a model without the prohibitive costs and time associated with large-scale data collection and labelling? Leveraging recent improvements in image generation platforms and techniques, we aim to develop a pipeline for accessible and efficient data augmentation.

Our main contributions include:

- A novel combination of perceived image depth mapping and original source data as latent image data.
- Hybrid augmentation techniques utilizing 3D modeling software (*Blender*) for creating enriched training datasets.
- Dynamic environment augmentation leveraging *Blender* to simulate varied lighting conditions.
- A unique image generation pipeline employing ComfyUI to maximize the quality and clarity of generated images.

By pushing the boundaries of synthetic data augmentation, this research aims to democratize access to effective machine learning tools, empowering researchers and practitioners in low-data environments.

The remainder of this thesis is organized as follows. Chapter 2 provides an overview of related works in generative image augmentation and identifies existing gaps in the literature. Chapter 3 presents the proposed methodology, including the data processing pipeline, depth estimation, and generative model integration. Chapter 4 outlines the experimental setup, datasets, and evaluation metrics. Chapter 5 presents and discusses the quantitative and qualitative results of the experiments. Chapter 6 concludes the thesis with a summary of contributions, while Chapter 7 outlines future work and possible extensions of this research.

Chapter 2

Background

2.1 The Emergence of Diffusion Models

In 2021 we were introduced to the publication "Diffusion Models Beat GANs on Image Synthesis"[7]. This publication showcased how diffusion based models beat GANs in the Frechet Inception Distance (FID) metric. This metric is used to evaluate the overall quality of images generated by generative models. It measures the similarity between two sets of images, typically being the real images from the dataset and those generated from the model.

Lower scores in the metric represent that the group of images are of higher quality and better diversity. Whereas higher scores typically represent the opposite. Testing GANs vs Diffusion Models on the `ImageNet128`, `ImagetNet256`, and `ImageNet512` it was shown that diffusion models receive noticeably better FID scores of 2.97, 4.59, and 7.72 compared to the GAN based model `BigGAN-deep's` 6.02, 6.95, and 8.43. This publication signalled the eventual rise in popularity of the diffusion based model approach. With a much easier to adjust model and better qualitative and quantitative results when compared to GAN models.

"Synthetic Image Datasets with Stable Diffusion and Data Augmentation"[4] is one of the earlier attempts to combine *Stable Diffusion* (SD) image generation and classical image augmentation approaches using the LAION-Aesthetics subset dataset as base to generate a new and improved combined one. The results were then tested on the *ImageNet* Large Scale Visual Recognition Challenge (ILSVRC) dataset, finding that while the top-1 and top-5 accuracy scores remained high at around 89%, it was discovered that "*162 animal classes have no correctly recognized images at all*" revealing that early models lacked generalizability in many different classification problems.

Jumping forward from 2021 to 2023 we can see many more attempts to utilize SD models for augmentation. "Diversify Your Vision Datasets with Automatic Diffusion-based Augmentation" created an Automated Language-guided Image Augmentation (ALIA) technique using `SD 1.5` and `GPT-4` which allowed for image description generation [9]. This technique was then applied to the Caltech-UCSD Birds

200 (CUB) dataset (a commonly used fine-grained classification dataset), iWildCam dataset (a trail camera style dataset that presents unique challenges such as imbalanced data, variance in image quality, and occlusion of the image subject), and a subset of the CUB dataset named Waterbirds (taking bird images from it and combining them with either water or land backgrounds). Each of these 3 datasets presents unique challenges. And as shown in the results, `SD1.5` generated image helped greatly increase the accuracy metrics in all three:

- iWildCam Subset: 67.51% -> 84.87%
- CUB 69.16% -> 72.17%
- Waterbirds 62.24% -> 71.40% accuracy.

Another study from the same year examined the potential of *Stable Diffusion* for improving few-shot object detection (FSOD) [17]. This study utilized SD in a much more complex fashion, generating new entities with it. These entities were then dropped onto new background images, creating almost completely new training data. Using a `ResNet-50` base model and the `ImageNet-100` dataset, the models achieved 70% top-5 accuracy and 43% top-1 accuracy with simply using class names. When using diverse backgrounds with the new images generated, the models achieved a 76% top-5 accuracy and 50% top-1 accuracy. We can see that clearly stable diffusion is a capable image generation tool and within recent years has become more generalizable to different problem sets. In "Is synthetic data from generative models ready for image recognition"[14] the authors perform a similar experiment, but also extrapolates to test whether diffusion based models are capable of creating synthetic data that is usable for large scale model pre-training for transfer learning. Using both the Vision Transformer - Base (`ViT-B`) and Residual Network - 50 layers (`ResNet-50`) models on augmented versions of over 17 datasets (notably `CIFAR-10, CIFAR-100, Caltech101, ImageNet, CUB,` etc...), it was found that in the zero-shot image recognition tasks, all datasets performed better with additional images generated from the diffusion based models. They also found that using synthetic data for pre-training models worked surprisingly well. Though the performance gains were not infinite. Eventually beginning to diminish as a saturation point is reached.

Further publications sought to determine the usefulness of Diffusion style models. "Towards Understanding Generative Data Augmentation"[32] analyzes the differences in GAN and Diffusion based models to create synthetic data for small datasets. This study found that diffusion models seemed to "*enjoy faster convergence rates*" than GAN's. Making them a more promising model type for general data augmentation. "Effective Data Augmentation With Diffusion Models"[27] attempts a novel technique of image-to-image transformation using a baseline training image as a latent image, and transforming it by way of a text to image diffusion model. This allowed for much more fine grained image manipulation whilst also

retaining the overall structure of the source images. "Data Augmentation for Image Classification using Generative AI"[24] also attempts to generate synthetic data in a similar way. Instead opting to preserve the foreground subject through a cutout method via segmentation mask, generating a unique background through text-to-image diffusion, and then superimposing the cutout overtop. This enhanced the diversity and generalizability significantly of the dataset, showcasing accuracy improvements of 15.6% to 23.5% for in and out-of-distribution data when compared to the examined baseline model performance. Though this method does seem to suffer from semantic issues with the subject being imposed over the wrong background, and is an area that the authors wished to further investigate potential solutions for. "Stable Diffusion Dataset Generation for Downstream Classification Tasks" tested more recent diffusion models (`Stable Diffusion 2.0`) and their ability to generate synthetic data [18]. The results showcased that in one third of cases, using purely synthetic data, they were able to achieve better results than training on the original datasets themselves.

"A Convolutional Neural Network Approach to Classifying Urban Spaces Using Generative Tools for Data Augmentation" examines the use of generative AI to balance a wildly unbalanced dataset via synthetic image generation [20]. Using a small seed dataset of under 700 images. Each class was then augmented until they contained 200 training images via two methods: *Midjourney* and the *Deep Dream Generator* image generation models. It was found that *Midjourney* was able to produce better training images overall when compared to the CNN based Deep Dream Generator. This is unsurprising considering the popularity of diffusion based models in the field of generative AI at this point in time. Using the newly augmented dataset, the authors reported that an overall 30% improvement in model performance was noticed. Asserting again that synthetic data is highly capable of balancing data and improving performance across numerous mode types and classification problems.

Further recent publications also explore novel techniques for image augmentation using diffusion models. "DiffuseMix: Label-Preserving Data Augmentation with Diffusion Models" [15] leverages text-to-image style prompts to preserve important image features whilst adding rich visual appearance changes to the input image. This is combined with fractal blending to achieve the final results. The style-transfer type augmentation resulted in *DiffuseMix* achieving better performance metrics compared to all other techniques (*AdaAug, CutMix, SnapMix, etc...*). Showing the power of the base diffusion model for numerous new augmentation techniques and their ability to beat current state of the art methods. "Data Augmentation for Object Detection via Controllable Diffusion Models" [10] explores another unique approach to synthetic image generation. This time a bounding box image generation method is employed. Consisting of a visual prior generator, a prompt constructor, a diffusion model, and a post filter with category-calibrated CLIP rank. "Dream DA: Generative Data Augmentation with Diffusion Models" explores another unique technique with a diffusion model base for image augmentation [12]. This DreamDA

method consists of perturbing (adding noise) each step of the image reverse diffusion to achieve more diverse and useful training data. In doing so, they achieved between a 0.7% - 4.6% top 1% accuracy increase on pre-trained models and 3.6% - 11.7% increase on models "trained from scratch".

2.2 Recent Innovations in Workflow Design

We can see clearly that some of the recent literature has begun to focus more on unique and novel techniques for improving the performance of diffusion based models. However, less work has been done on improving the diffusion step itself. With this in mind, many new SD based front end applications have recently presented themselves to the public. *Automatic1111*, *Forge*, and most significantly *ComfyUI* allow for much higher fine control over how the images are generated, what happens in the intermediate steps of the diffusion process, and gives users the ability to create much more complex image generation pipelines.

Recent publications like "OmniGen: Unified Image Generation"[28] give us some insight into the potential future of image augmentation pipeline and workflows. Exploring the idea of a unified diffusion model architecture that can be used for various image generation tasks. These include text-to-image, image-editing, controllable generation, and image restoration. The idea being to eventually have a "diffusion assistant" that can help with tasks like current GPT style models do. As of now, "*users typically need to load multiple models and perform multi-step processing to ultimately generate a satisfactory image, making the workflow very cumbersome and costly*". *Omnigen* shows that it is very much possible to achieve this sort of architecture with a "one stop" solution for multiple unique generation tasks and problems. This sets a future baseline for the next generation of diffusion models and services. A unified model that supports further downstream tasks and problems, a simplified architecture that allows for user-friendly design adjustments to images, and multi-task knowledge transfer. The future of image generation will hopefully be made of "General Purpose" image generation models such as this.

Alongside GANs and diffusion models, VAEs have also been considered for image synthesis and augmentation tasks. They' re known for their stable training and well-structured latent spaces, which make them useful for certain controlled generation tasks. However, one of the main drawbacks is that the images they produce often lack the fine-grained detail and realism seen in more recent models like diffusion-based approaches. That said, VAEs played an important role in shaping the direction of generative AI and still form a key part of many hybrid architectures used today—including within Stable Diffusion itself.

New model architectures are also being presented which could radically change the new use cases for generative AI data augmentation. The *Mamba* architecture for example (using its much faster inference time and performance) can theoretically train on much higher data volumes and therefore tackle significantly more difficult

vision problems. Some of these include generation of full 3D objects or large scale environments [13][33]. Thus the need for even more training data is required.

2.3 Systematic Literature Review of Related Works

To better understand the field of Generative Images for Dataset Augmentation, a systematic literature review of 20 publications was conducted to observe the trends for image augmentation, and technologies utilized. We began by asking what types of techniques are used for generative image augmentation? The inclusion criteria for a publication was as follows. First, it must contain some generative method for image generation/augmentation. Second, it must be no older than 10 years and be published in English. Third, the studies must offer qualitative results that are readily apparent or quantitative results that are explainable. Each of these publications was then categorized into one of the three following technique categories:

Simple

- Methods that involve minimal or well-established augmentations, such as basic transformations (e.g., flipping, rotating, colour changes, etc...).
- Use of traditional models or straightforward application of pre-trained models without complex architectures.

Intermediate

- Methods that incorporate more advanced but commonly understood techniques, such as using existing generative models (GANs or diffusion models) with slight modifications or applying well-known neural networks.
- Methods that use pre-trained models but with custom fine-tuning.

Complex

- Methods that propose novel architectures, involve hybrid approaches (e.g., combining multiple models or introducing multi-stage pipelines), or use advanced techniques like automatic generation of augmentations, significant changes to the model architecture, or training from scratch on complex models.
- Approaches that incorporate cutting-edge techniques, such as multimodal data (e.g., text-to-image generation), or involve heavy computational resources and long training times.

ID	Name	Methods / Techniques	Datasets	Gen AI Model Base	Model Type	Method Complexity	Year
1	Data Augmentation in Emotion Classification Using Generative Adversarial Networks	Base GAN Generation, Traditional Augmentation	Facial Expression Recognition Database (2013), SFEW, JAFFE	Custom GAN	GAN	Simple	2017
2	Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification	Base GAN Generation, Traditional Augmentation	Custom Lesion Liver Dataset - Sheba Medical Center	GAN	GAN	Simple	2018
3	GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks	Base GAN Generation, Traditional Augmentation	Fluid-Attenuated Inversion Recovery (FLAIR)	PGAN	GAN	Intermediate	2018
4	Classification Accuracy Score for Conditional Generative Models	BigGAN Generation, VQ-VAE Generation	ImageNet,CIFAR-10	BigGAN,VQ-VAE	GAN	Intermediate	2019
5	A survey on Image Data Augmentation for Deep Learning	DCGAN Generation, Traditional Augmentation	Custom Compound Tomography (CT) scans of 182 liver lesions	DCGAN	GAN	Intermediate	2019
6	Diffusion Models Beat GANs on Image Synthesis	Stable Diffusion Generation, Base GAN Generation	ImageNet 256, ImageNet 512,	Early Stable DIffusion Prototype	Diffusion	Intermediate	2021
7	Synthetic Image Datasets with Stable Diffusion and Data Augmentation	Stable Diffusion 1.4 Generation,	LAION-Aesthetics subset	Stable Diffusion	Diffusion	Intermediate	2021
8	Synthetic Data from Diffusion Models Improves ImageNet Classification	Imagen Generation (Diffusion)	ImageNet1K, ResNet50	Imagen	Diffusion	Intermediate	2023
9	Diversify Your Vision Datasets with Automatic Diffusion-based Augmentation	GPT-4 Caption Generation, Stable Diffusion Generation, CLIP and Confience Based Semantic Filter.	CUB, iWildCam, Waterbirds	Stable DIffusion 1.5	Diffusion	Complex	2023
10	Explore the Power of Synthetic Data on Few-shot Object Detection	Stable Diffusion Generation, Background Replacement	PASCAL VOC, MSCOCO	Stable Diffusion	Diffusion	Complex	2023
11	Fake it till you make it: Learning transferable representations from synthetic ImageNet clones	Stable Diffusion Generation	ImageNet-100	Stable Diffusion	Diffusion	Complex	2023
12	Is synthetic data from generative models ready for iamge recognition	GLIDE Image Generation (Diffusion)	17 datasets: CIFAR-10, CIFAR-100, Caltech101, Clatech256, ImageNet, CUB, Flower, Food, Pets, DTD, ImageNet-Sketch, etc...	GLIDE-txt2im	Diffusion	Intermediate	2023
13	Towards Understanding Generative Data Augmentation	cDCGAN, StyleGAN2-ADA, EDM Image Generation (GAN)	CIFAR-10	cDCGAN, StyleGAN2-ADA, EDM	GAN	Simple	2023
14	Effective Data Augmentation With Diffusion Models	Stable Diffusion 1.4 Generation,	PascalVOC, MSCOCO, LeafySpurge	CompVis/stable-diffusion-v1-4	Diffusion	Intermediate	2023
15	Data Augmentation for Image Classification using Generative AI	Stable Diffusion XL Generation	ImageNet10	Stable Diffusion XL	Diffusion	Complex	2024
16	A convolutional neural network approach to classifying urban spaces using generative tools for data augmentation	Segment Anything Model (SAM), LLM for Background Caption Generation, Stable Diffusion XL Generation	Custom image dataset from free Adobe Stock images.	Deep Dream Generator, Midjourney	CNN,Diff	Simple/Intermediate	2024
17	DiffuseMix: Label-Preserving Data Augmentation with Diffusion Models	Deep Dream Generator, Midjourney Generation (DIffusion)	ImageNet, CIFAR100, Tiny-ImageNet-200, Oxford-102 Flower, Stanford Cars, Aircraft, Caltech-UCSD Birds-200-2011 (CUB)	InstructPix2Pix diffusion Model	Diffusion	Complex	2024
18	Data Augmentation for Object Detection via Controllable Diffusion Models	Visual Prior Generator, Stable Diffusion Generation, Post Filter w/ Category-Calibrated CLIP Rank	PascalVOC, MSCOCO	Stable Diffusion	Diffusion	Complex	2024
19	DreamDA: Generative Data Augmentation with Diffusion Models	ImageNet Generation, DreamDA Custom Perturbation	Caltech101, Oxford-IIT Pet, Stanford Cars, STL-10, Shenzhen Tuberculosis	Stable Diffusion	Diffusion	Complex	2024
20	Stable DIffusion Dataset Generation for Downstream Classification Tasks	Stable Diffusion 2.0 Generation	CIFAR10, CIFAR100, PathMNIST, DermaMNIST, BloodMNIST, RetinaMNIST	Stable Diffusion 2.0	Diffusion	Complex	2024

Table 2.1: Analysis of State of the Art Publications in the field of Generative Image augmentation and Generative Image Generation.

We can see in Table. 2.1 that the first early iterations of this type of image augmentation utilize GANs. Most of these studies involved using the base GAN model with additional steps taken on top to bolster the quality thresholds or ensure minimal data leakage. These early datasets focused mostly on medical imaging but would eventually move to general case image datasets. These studies provide promising results. However, from a qualitative standpoint the images generated lack the photorealistic qualities that modern diffusion based architectures provide the user.

More of the recent publications opt to utilize diffusion based models for their generation. In this case, the most popular of these being the *Stable Diffusion* based models including its base `1.0, 1.4, 1.5, 2.0`, and `XL` variants. These models (along with additional techniques) are compiled into pipelines for image generation at a higher quality with a more comprehensive list of datasets tested and positive results acknowledged. We can see also that the general complexity of these pipelines increases as we progress through the literature in linear time. However, there does appear to be some lack of modern diffusion based workflows in the overall literature. These workflows allow users to easily adjust hyper-parameters, and inter-step functionality of the models while also applying nodes which can radically alter image generation procedures. Before the formation of these styles of workflows, making complex model generation pipelines would be a much more difficult process and require greater knowledge of available repositories and addons. Now, centralized hubs of these nodes are easily accessible to anyone and has subsequently lowered the barrier of entry to the field of generative AI images.

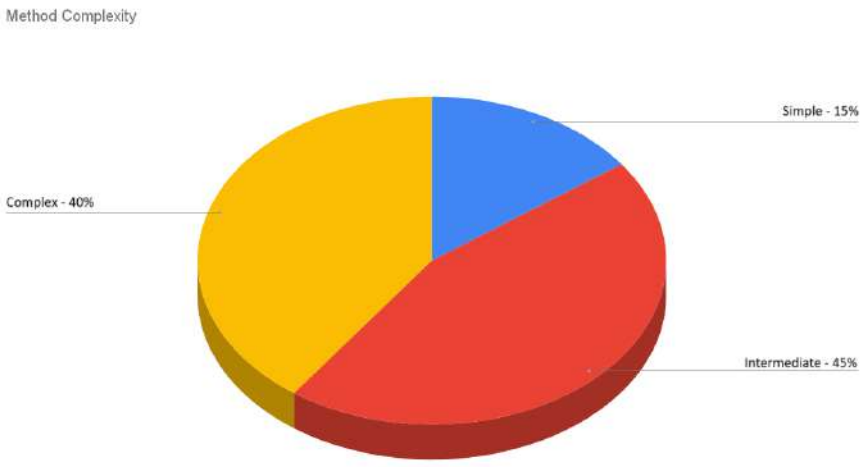


Figure 2.1: Results of method complexity analysis of the state of the art in the image generation and augmentation field.

2.4 Identified Gaps in Current Literature

The reviewed literature demonstrates that diffusion-based models (particularly *Stable Diffusion* variants `1.4` through `XL`) are consistently effective for the task of image augmentation. Enhancing classification performance across a wide range of domains.

However, most recent studies continue to treat the diffusion model as a "black-box" within their augmentation pipelines. They often do not exploit the growing availability of modern, modular front-ends such as *ComfyUI*, *Automatic1111*, or *OmniGen*. As noted in Table 2.1 and Fig.3.1, much of the generation and augmentation done in recent literature stays firmly in the simple and intermediate domains. Missing out on these tools and their abilities for extensive customization such as: Hyperparameter tuning, fine-grained noise control, prompt chaining, modular node-based design, as well as integrating these tools into multi stage pipelines for improved results.

This gap is significant as it overlooks the potential for highly targeted and efficient dataset generation. Which is particularly vital for small-scale or domain-specific applications where traditional data collection is impractical. Additionally, these workflows lower the technical barrier to entry making generative augmentation more accessible to individual researchers, small organizations, and under-serviced domains such as agriculture or field-based monitoring.

This thesis addresses the identified gap by proposing a flexible, modular pipeline that combines Blender-based depth mapping, 3D environment simulation, and ComfyUI-based diffusion workflows to generate semantically consistent synthetic images. While our work does not aim to benchmark against all existing methods, we evaluate the pipeline's effectiveness in improving classification performance within our defined scope of low-data scenarios, and explore its potential as an accessible, customizable solution for data augmentation in these resource-limited contexts.

Chapter 3

Methodology

This section outlines the methodological framework employed to generate synthetic image augmentations using generative AI and 3D rendering techniques. The pipeline consists of five primary stages: data collection, depth map generation, 3D scene rendering, text prompt synthesis, and generative augmentation as per Fig.3.1. Each component was designed to enhance data diversity and realism in scenarios characterized by limited training data.

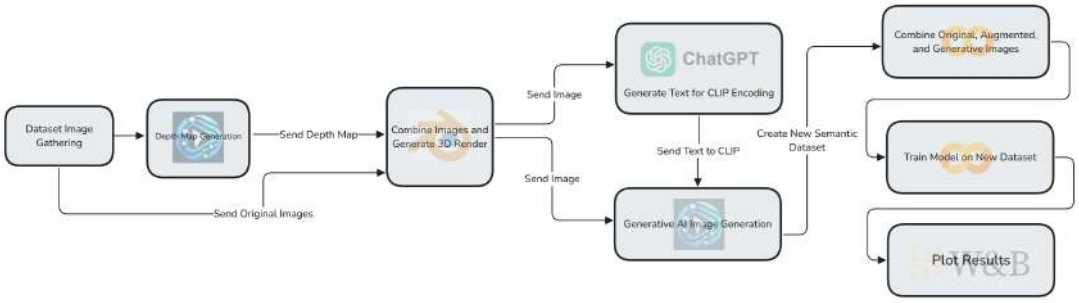


Figure 3.1: Showcase of the full methodological framework. An original dataset is captured. Next being sent through the pipeline where depth maps are generated, 3D renders are taken, image prompts are created, and then combined together to generate new semantic images. These images are then added into the dataset and a computer vision model is trained.

Aspect	Description
Time & Location	Summer, Afternoon, Cocoa farm in Sulawesi, Indonesia
Purpose	Used as a baseline dataset for classification and for further augmentation experiments
Task Type	Binary classification of cocoa plant varieties (Local vs Hybrid)
Camera Used	NIKON D5100 DSLR
Number of Images	30 images per class (60 total)
Collection Context	Simulates a low-resource scenario typical of a single individual's data-gathering effort over a short timespan

Table 3.1: Overview of Dataset Collection Process.

The dataset used for this research was a custom sourced cocoa plant dataset from Sulawesi, Indonesia. The dataset is comprised of 65 RGB images divided into two semantic classes: "Local" and "Hybrid" plants per Tab.3.1. As seen in Fig.3.2, there are clear distinguishing factors between each class. The local variety in Fig.3.2a has a more yellow hue to the pod itself, is more elongated, and generally contains more warts. The hybrid variety in Fig.3.2b is a more reddish hue, with a generally uniform shape and stout appearance. These clear differences make for an excellent ground-truth test for the pipeline itself. As the differences between classes are quite apparent, and thus likely only more data is needed to train a fairly accurate and generalizable model to identify the differences. These images served as the foundation for subsequent augmentation processes. Prior to the augmentation, these images were standardized in resolution and format to ensure compatibility with the depth estimation and rendering stages.

(a) Local Variety (b) Hybrid Variety

Figure 3.2: Example pictures taken of both the local and hybrid varieties of cocoa plants.

To asses the impact of performing different augmentations to the data itself, the original dataset was transformed into three variants, each increasing in size over the other as seen in Table 3.2. Dataset 1 contains the original 65 images without any

form of augmentation applied, serving as the control group for further comparison. Dataset 2 contains data that has undergone classical augmentation techniques including random rotation, random horizontal flipping, and random resized cropping as seen in Table 3.3. This process expanded the size of our dataset to 325 images. Dataset 3 integrates both the classical techniques and our semantic augmentations. A subset of the original data was passed through a custom Blender-based rendering pipeline where a depth map and random lighting effects were applied to generate novel 2D perspectives. These rendered images were then described using the OpenAI `GPT-4o API`, which would generate both positive and negative text prompts. Formatted specifically to be input into the CLIP-based generative text encodings. The resulting text-image pairs were then input into a *Stable Diffusion* model via our custom *ComfyUI* pipeline, producing semantically aligned, synthetic images. Each of these images was then used to generate between 10-50 image variants via the *ComfyUI* workflow. These images were then merged with the original dataset, and subjected to the previous classical augmentation techniques, ultimately producing a dataset comprising approximately 1000 images (500 per class). Leading to a significant size increase from our original dataset.

Dataset	**Techniques**	**Size (# Images)**
1	No Augmentation	65
2	Basic Augmentation	325
3	Basic + Semantic Augmentation	>1000

Table 3.2: Different techniques used in each dataset.

Technique	**Description**
Random Rotation	Rotates the image around its center point by a random number of degrees, within a specified range and probability.
Random Horizontal Flip	Horizontally flips the image with a given probability to simulate mirrored samples.
Random Resized Crop	Randomly crops a section of the image and resizes it to the desired output dimensions.

Table 3.3: Basic augmentation methods used in Dataset 2.

These three unique datasets are then used to test on multiple different computer vision models using various training techniques which is further discussed in section 3.8 culminating in feeding the training and testing data achieved using the *Google Colab IDE* through to the *Weights & Biases API*. Producing graphed results and information to compare different between different runs.

To ensure reproducibility across all experiments, every random operation (including image transformations, and model training initialization) was seeded using

a fixed random seed of 42. All training runs were conducted with consistent hyper-parameters unless otherwise stated, and the *ComfyUI* nodes were locked to their current specific versions throughout the entire testing process to avoid pipeline drift (subtle changing of results through updates to software, random seeds, or inconsistent pipeline integration). Each batch of images generated with *ComfyUI* were however given their own unique seed as this is an inherent requirement for diverse image generation.

3.1 ChatGPT - Text Prompt Generation

Open AI's *ChatGPT* [21] system has in recent years become a very popular choice for people looking for help in task completion in many fields. An area that we looked to take advantage of the technology for our pipeline was `GPT-4o`'s ability to take both visual and textual data (multi-model data) as input and formulate a response based on the two.

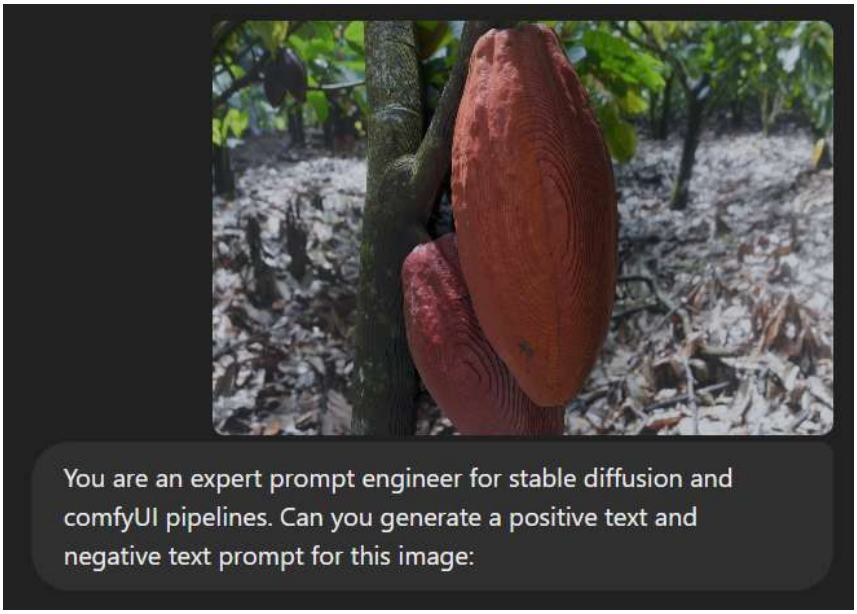


Figure 3.3: Example prompt given to ChatGPT to instruct it to perform prompt engineering on the included image.

A seen in Fig.3.3, the model is given positive reinforcement before instruction. This is based on research performed by [16], and [29] which showcases a positive correlation with encouraging the model and receiving better results. Once the instructions have been processed, we receive an output similar to the one below:

> **Positive Prompt:** "A close-up view of two ripe, textured cocoa pods attached to a tree trunk in a tropical plantation. The cocoas pods are rough

and earthy, displaying deep reddish-brown hues with natural variation. The tree bark shows a rugged surface with patches of moss, providing an organic contrast to the pods. The background features a cocoa farm setting with green foliage, leaves, and dappled sunlight casting soft shadows on the ground, which is covered with dry, scattered leaves and twigs. The scene captures the rich, natural texture and environment of a cocoa farm with depth and clarity."

Negative Prompt: "Out-of-focus image, artificial colors, overly bright or washed-out background, smooth textures on the cocoa pods, lack of natural detail, unrealistic lighting, digital noise, blurry background, overly saturated or desaturated colors, distorted shapes, unnatural patterns, unwanted reflections, extra or misplaced objects, low resolution."

As can be seen, these prompts are quite detailed and allow us to generate high quality results in minimal time. This system opened up the ability for us to automate the prompt generation portion of our workflow.

3.2 Depth Aware Image Estimation and Mapping

Depth maps represent the distance between objects in a scene and the camera, encoded as grayscale intensity. Pixels closer to the camera appear brighter, while farther ones are darker. These maps are essential for simulating three-dimensionality and realistic lighting in synthetic image generation.

3.2.1 Classical Depth Estimation Methods

Historically, depth estimation has been achieved through hardware-based systems and geometric techniques. Devices such as Microsoft's **Kinect** used structured light or time-of-flight sensors to infer the depth of their surroundings. These systems would emit infrared patterns and measure the distortion upon return to construct dense 3D depth maps. While accurate, such hardware solutions are expensive, limited to specific environments, and require physical access to the scene, making them a poor choice for our needs.

Software-based classical approaches attempted monocular depth estimation via stereo vision or structure-from-motion. These relied on multi-view geometry to triangulate distances between pixels. However, these methods required multiple images from different angles and were often computationally intensive and sensitive to occlusions, textureless regions, and lighting conditions—once again making them an expensive and difficult tool to utilize. Classical single-image methods also include image processing algorithms such as optical flow estimation, though these often struggled with accurately recognizing depth discontinuities, especially around shadow boundaries, leading to unreliable depth cues.

3.2.2 Deep Learning-Based Depth Mapping

In contrast, modern deep learning-based approaches can estimate depth from a single image, but they come with their own challenges. Notably, the absolute scale of depth cannot be inferred directly from generated predictions, as these models only learn relative spatial relationships. Additionally, high-performing models such as *Marigold* are extremely resource-intensive, with significant VRAM requirements, making them difficult to integrate into lightweight pipelines or consumer-grade hardware environments.

More popular consumer-grade-hardware-friendly models in this space include *MiDaS*, *DPT*, and the more recent *DepthAnything*. These networks are trained on diverse image-depth pairs and can generalize well across most scenes. However, trade-offs exist between model size, inference speed, and depth resolution. Larger models tend to be slower but generate more detailed and consistent depth outputs.

3.2.3 Our Approach - DepthAnything V2

For our pipeline, we chose to investigate the `DepthAnything V2` (DA) model due to its native integration with *ComfyUI* via a dedicated node. This seamless compatibility allowed us to incorporate it directly into our image generation workflow with minimal setup. Additionally, `DA V2` offers multiple checkpoints of varying size, enabling less powerful GPUs to utilize the techniques and speed up generation time on trivial generation tasks that do not require the most precise mapping.

The `DA V2` node gives us the ability to adjust the size of the model used and therefore the time we would like to spend generating each map. As we can see in Fig.3.4, preliminary results of testing the difference in generation quality on an example highly detailed source image shows a negligible difference in per model results. Using a model size larger than the base version provides diminishing returns with no real increase in detail captured. Thus, the base model was chosen for the generation of all subsequent depth maps.

Figure 3.4: Results of different DepthAnything V2 model size outputs. Comparing different depth map outputs quality based on model size shows diminishing returns.

3.3 Blender - Depth Based Image Rendering

Blender [2] is a 3D modelling, texturing, and rendering tool that aids in generating new images from our source image dataset. This is done by combining depth mappings of the source image and applying topology displacement to in a 3D environment. Specifically, we begin by generating the depth map of an image through the use of the `DepthAnything V2` algorithm / node inside of *ComfyUI* as discussed in Sec.3.2.

Our source image is then imported into blender and applied to a plane mesh as a texture. This mesh is subdivided initially into a 8 x 8 grid giving us 64 quads (a shape made from 4 vertices). This plane is then repeatedly subdivided using the Subdivision Surface modifier (Catmull-Clark Algorithm), with a subdivision surface level of 1, allowing us to test the different levels of subdivision. Many different amounts were tested. However, it was eventually decided upon using 5 additional subdivisions. This amount resulted in an appropriate balance between result fidelity and manageable render times. We apply the 5 subdivision surface modifiers to arrive at a final value of 65,536 quads:

- Initial faces with loop cuts: 64 quads
- 1st subdivision: 64 x 4 = 256 quads
- 2nd subdivision: 256 x 4 = 1,024 quads
- 3rd subdivision: 1,024 x 4 = 4,096 quads
- 4th subdivision: 4,096 x 4 = 16,384 quads
- 5th subdivision: 16,384 x 4 = 65,536 quads

This final geometry offers a dense surface that can allow for detailed displacement mapping with smooth deformations. Once this is finished we apply the following modifications to the material output:

Fig.3.5 showcases our texture node workflow. This pipeline takes in our depth map image and source image object (which has been applied to our plane) and utilizes the depth map image values as a displacement map to the source image plane itself. This is done by interpolating the grayscale image of the depth map as values between 0 and 1. With 0 representing a fully black pixel and 1 representing a fully white one. These numbers are applied through our displacement node to the image plane's individual vertices. Doing this causes the whiter portions to be raised (as seen in Fig.3.6b) vertically (Z-axis in *Blender*) while the darker parts stay in place.

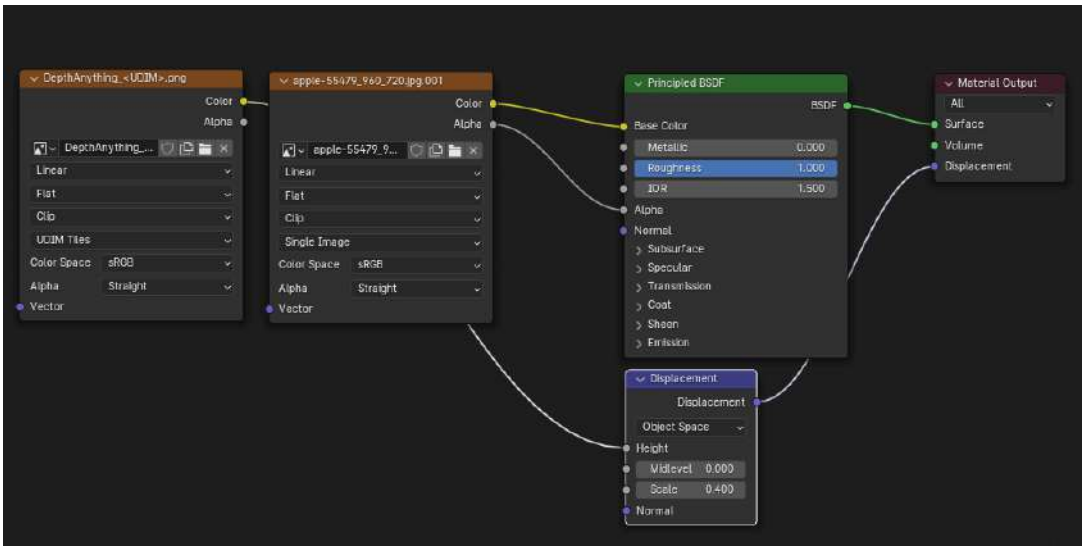


Figure 3.5: Shader node workflow to deform the image plane into a 3D object based on depth map color values.

(a) Top-down view (b) Side view

Figure 3.6: Example 3D image augmentation from the render perspective (a) and side view (b).

We then render an overhead / top-down view of the image (Fig.3.6a) and adjust lighting parameters as we please. This gives the ability to drastically change the lighting conditions of each image and increase diversity of results in future steps. Blender also offers Python scripting which allows us to fully automate the process of loading new images and their depth map based counterparts, rendering images, and saving them in the correct folders. This allows us to ostensibly "click one button" and generate our data.

The rendered scenes utilized *Blender*'s Cycles rendering technique. A physically based path-tracing renderer which simulates light paths and bounces to create photorealistic image renders. The important settings used for our rendering include 4096

samples and a 0.3 noise threshold. Additonally, HDRI (High Dynamic Range Image) lighting was used for environment variation. An HDRI is a 360-degree panoramic image that encompasses the entire 3D scene. When a render is executed, the real world lighting data from the HDRI is used to influence the overall appearance of the scene in the final image output. The final image size of our render is 1920x1080 pixels. This provides us with a high fidelity image with minimal noise, whilst still keeping rendering times low.

3.4 ComfyUI - Model Based Image Generation

ComfyUI [6] is a graphical user interface which allows for the loading of pre-trained stable diffusion style models, adjustment of hyperparameters, and adding of additional nodes which further adjust values of the image generation process. From these workflows, it is possible to then generate images to be used.

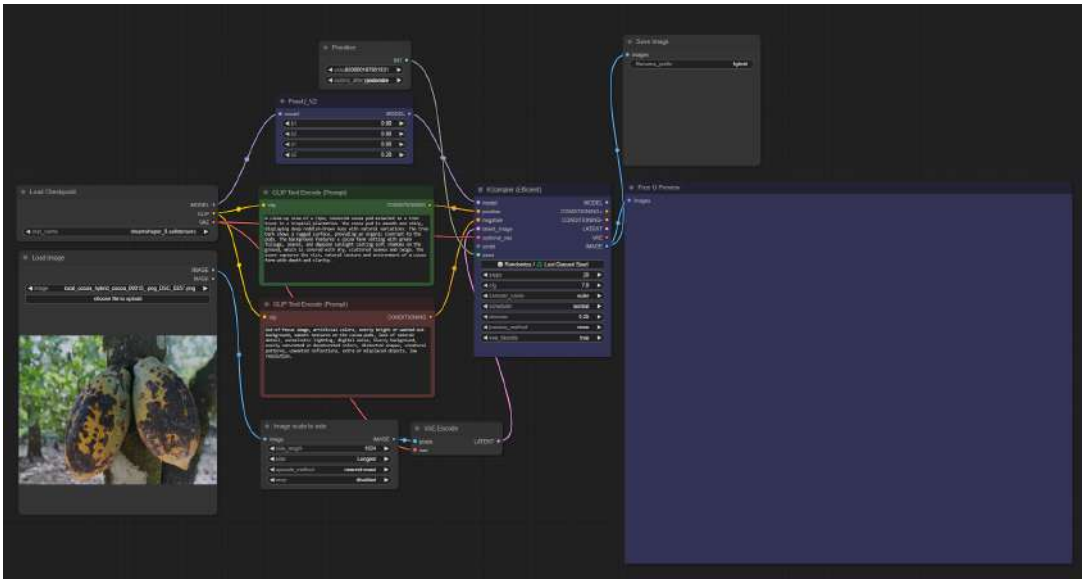


Figure 3.7: Full comfyUI workflow utilized for the image generation.

As shown in Fig.3.7, our full workflow makes use of numerous additional nodes and techniques not offered in a basic workflow setup or straight input -> model -> output workflow. We begin by using our Blender-augmented image as a base for the latent noise in our model. Giving us a structured image base to generate our new semantic augmented images off of (rather than completely random noise). After, we input the positive and negative text prompts generated for us from ChatGPT as our Positive and Negative CLIP text encodings. This data is then sent into the K-Sampler with hyperparameters adjusted to produce the most satisfactory image possible. Additionally, the model chosen is slightly altered using the `FreeU_V2` node. This node strategically re-weights the U-nets skips, connections, and backbone

feature maps to help improve granular detail and sample quality at no additional cost to us. The `FreeU_V2` node is appropriately parameterized (b1= 1.1, b2=1.2, s1=0.9, s2=0.8) to optimize for detail while minimizing the introduction of artifacts. The generated image from this *ComfyUI* workflow is then saved and stored in the appropriate data folder for further processing and investigation.

3.5 Stable Diffusion Model Selection

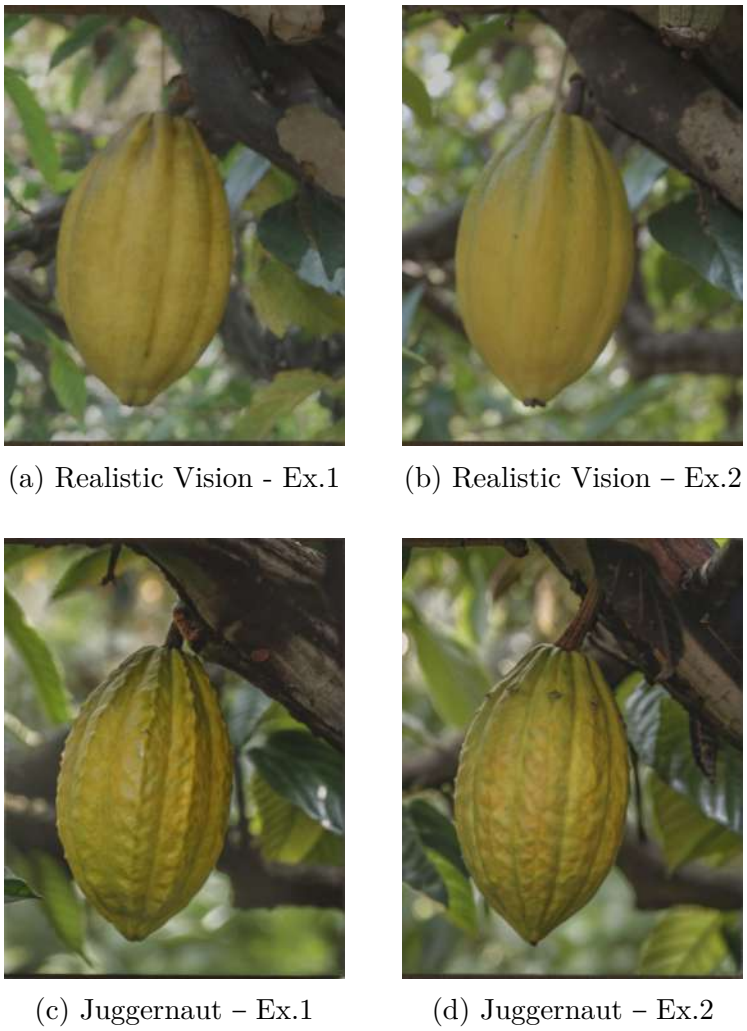
As part of the *ComfyUI* setup, a stable diffusion model is required to begin generating images. This task in itself is somewhat complicated as there exists many different varieties of models to choose from. For our research, we decided upon staying only in the `SD1.5` and `SDXL` family of models. This was done for three main reasons. Firstly, local runtime requirements for these models are either low (**4-6GB VRAM** for `SD1.5`) or moderate (**8-12GB VRAM** for `SDXL`) in terms of requirements. Allowing for us to mimic the hardware specifications that a single user, with their own personal computer could achieve. This also significantly lowered the cost of producing data. As testing can be done on the users own device and one can remain unconcerned about excess API calls to more advanced models. Secondly, there exists already a large swath of fine-tuned models created by users in the generative AI community. Making it easier for us to select a model that can closely achieve the style we wanted. Finally, is the option for fine tuning our models. The `SD1.5` and `SDXL` family of models allows for the use of *LoRA*'s (lightweight fine-tuning), Custom *VAE*s (Variable Auto Encoder), *ControlNets*, and depth/pose/image conditioning, allowing for a model to be extended to more niche tasks without the need for retraining.

Multiple qualitative tests were performed with different `SD1.5` and `SDXL` models to examine the overall quality of image-to-image generation. Using a test image from our gathered dataset we compared the overall image generation capabilities of each model. The full list of models tested includes:

- `Epicrealism_natural (SD 1.5)`
- `Dreamlike-photoreal-2.0 (SD 1.5)`
- `RealisticVision (SD 1.5)`
- `AlbedoBase XL (SDXL)`
- `Juggernaut XL (SDXL)`

Each model was was chosen for its ability to create realistic images. However, some fairly obvious difference can be seen in each models final output. Fig.3.8 showcases the gulf in detail between what `SD1.5` family of models are capable of creating (Fig.3.8a,3.8b) compared to `SDXL`'s (Fig.3.8c,3.8d). With much finer grain

detail, only a moderate increase in rendering time, and numerous reasons in Table 3.4 such as better prompt understanding, and higher quality render resolution contributed to our eventual decision to choose the `SDXL` family and settle upon using the `Juggernaut XL` model.



(a) Realistic Vision - Ex.1 (b) Realistic Vision – Ex.2

(c) Juggernaut – Ex.1 (d) Juggernaut – Ex.2

Figure 3.8: Visual comparison between Realistic Vision (SD1.5) and Juggernaut (SDXL) outputs.

Feature	Stable Diffusion 1.5	SDXL 1.0
Release Year	2022	2023
Base Resolution	512 × 512	1024 × 1024
Architecture	U-Net with CLIP ViT-L/14	Two-Stage: Base + Refiner with CLIP-G
Prompt Understanding	Moderate	Significantly Improved
Detail & Realism	Good	High (especially with Refiner)
Model Size	~4.2 GB	~6.6 GB (Base) + ~6.6 GB (Refiner)
Fine-tuning Support	Widely supported (LoRA, Dreambooth)	Supported but more resource-intensive
Best Use Case	Fast generation, lightweight systems	High-fidelity imagery, production-quality art

Table 3.4: Comparison between Stable Diffusion 1.5 and SDXL 1.0.

3.6 Google Colab - Model Training Environment

Google Colab [25] is an online IDE (Integrated Development Environment) that allows for quick iteration on projects. Using *Jupyter Notebook* style cells, users can quickly edit and test changes made to their code. *Google Colab* also offers the ability to sync to *Google Drive*. Allowing for easy dataset access and storage. Additionally, Colab allows for the free and paid use of numerous GPUs for training and testing models. These three main features are why we decided to use this platform to perform our experiments. While we could use our own hardware, the GPUs offered by Google Colab are free and perform far better. This helps to stop the bottleneck of waiting for GPUs to train models and get us results. The exact GPU utilized for our model training was the **NVIDIA Tesla T4 (16GB)** GPU, a good GPU for deep learning and model training with **16GB VRAM** available to use.

3.7 Weights & Biases - Results Visualization

Weights & Biases (*W&B*) [1] is a popular platform for tracking machine learning experiments, managing models, and collaborating on machine learning projects. The key features that we utilized the service for were experiment tracking, model versioning, and the collaborative workflow. The experiment tracking allowed for us to log each metric, hyperparameter, and compare results with other runs on similar models, showing us clear results through the dashboard as can be seen in Fig.3.9.

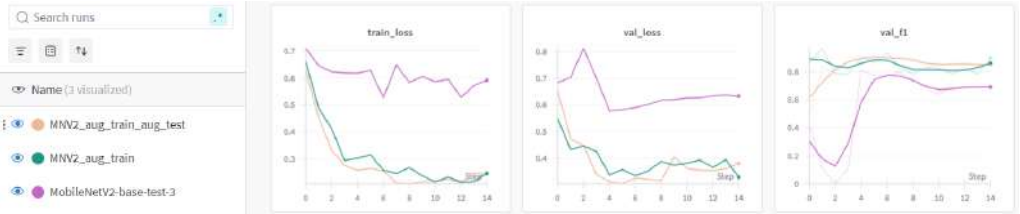


Figure 3.9: Dashboard results of multiple model runs.

To use the service we simply can create a call to the *W&B* with the information and unique *W&B* API key to start the training process. Model versioning was also

important for us. Each run creates a random name and populates the information field with the model's information, which allows for easy comparison and tracking of results between different models per Fig. 3.10.

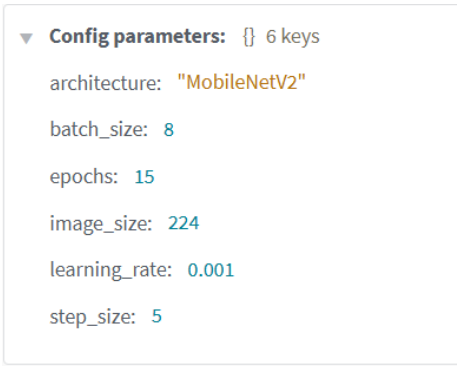


Figure 3.10: Configuration data about the current model being viewed.

Finally, as this experiment was a collaborative process, being able to share results easily to one another and compare was invaluable for quickly iterating on experiments.

3.8 Computer Vision Model Selection and Training Strategy

As mentioned above, from the outset of this research we were interested in testing the possibility of enhancing model performance results in data scarce scenarios. Thus, it was important to choose the correct base model to train on. When in a data scarce scenario, there are numerous important factors to consider:

Firstly, smaller and more lightweight models are generally better suited for these tasks as they contain fewer overall parameters and thus require less training data. Because of this, very large models such as the `ResNet101`, `ResNet50`, or `ResNet18` are generally best avoided as the require significantly more data to generalize well [30]. In our preliminary experiments, we also found that models under 10 million parameters performed the best on the base and augmented data. Additionally, sticking to CNN based models did appear to provide inherently better results as referenced in [8]. Adding an additional dropout layer also seemed to help with over-fitting [26] and was subsequently added to all of our initial model tests.

Secondly, an important technique to make use of in a scenario such as this was transfer learning. Using models with pretrained weights and most of the layers frozen to allow only a very small section of the model (usually the final 1-3 layers including the classifier) to be trained on the new data. With that in mind we created a short list of pretrained model weights (Table **??**) to act as our ground truth testing group.

Model	# Parameters	Description
ResNet18	11.7 million	Shallower version of ResNet with fewer layers. Faster to train with reduced capacity.
MobileNetV3-large	5.4 million	Optimized for mobile devices. Uses depthwise separable convolutions and squeeze-and-excitation blocks.
MobileNetV2	3.5 million	Efficient architecture with inverted residuals and linear bottlenecks.
SqueezeNet	1.25 million	Very compact CNN using Fire modules. Designed for high accuracy with minimal parameters.

Table 3.5: Overview and Evaluation of Lightweight CNN Models.

Finally, it should not be forgotten to use strong augmentation techniques [5] and k-fold cross validation. In data-scarce scenarios, some form of data augmentation is almost always a requirement. As no matter how small the model being trained, when the dataset is too small, overfitting is an unfortunate reality. Overfitting occurs when the model fully familiarizes itself with all of the images in the training data, rather than learning the patterns, shapes, colours, etc. This shows up most frequently with high training scores but mediocre testing results, showing that the model in fact is quite poor at generalizing to new or real-world data. To combat this, a slew of traditional augmentation techniques are available to artificially increase the size of the dataset being utilized. Techniques like the ones described in Table 3.3 are some of the most commonly used. These allow for the creation of a significantly larger amount of data that can help vastly improve a model' s generalization.

However, in addition to these classical augmentations, our pipeline also incorporates a generative augmentation stage as an intentional response to the limitations imposed by small datasets. By leveraging depth-aware 3D rendering and text-to-image synthesis through Stable Diffusion, we create entirely new, semantically consistent samples that go beyond mere spatial transformations. This approach not only increases dataset volume, but also introduces greater visual diversity and richer variation than traditional methods alone can provide. As such, it plays a critical role in improving model generalization under low-data conditions, and represents a key contribution of this work.

To reliably evaluate the effects of these augmentation strategies-both classical and generative—we also apply stratified K-fold cross validation, a widely used re-

sampling technique particularly well-suited for small datasets. The main idea of the technique itself is to partition the data into k equally sized and mutually exclusive subsets (folds). The model is then trained on k-1 of the folds and validated on the remaining one. This process is repeated k times, rotating the validation fold until each fold has been used as the validation dataset. The scores of all validation folds are then averaged, to provide a more accurate estimation of a models performance, compared to only using a single, classical, train-test split method. Based on research conducted in [19] and general field sentiment, we utilize a k values of 2 and 5 for our training. As with a smaller dataset (<5000) using high values of k give a potential for wasted computation / training time.

A final note about the training procedures themselves. Each model was trained using the `Adam` optimizer, a learning rate (lr) of **0.001**, a batch size of **8**, and **15** total epochs, in accordance with most suggestions for training models on small datasets.

Chapter 4

Experiment

To validate the effectiveness of the proposed data augmentation pipeline, we conducted a series of experiments that span across model training, image synthesis, 3D rendering, and evaluation. This section details the computational resources used, the software frameworks adopted, the design of the datasets, and the procedures for model training and evaluation.

Our experiments were structured to progressively assess the impact of classical and semantic augmentation techniques on model generalization under data-scarce conditions. The goal of these experiments were twofold. First, to measure the improvement in model accuracy and generalization across increasingly augmented datasets. Secondly, to assess the semantic and perceptual realism of the synthetically generated images. Together, these experiments provide both quantitative and qualitative evidence of the augmentation pipeline's utility in low-data computer vision tasks.

4.1 Hardware Specifications

The model training, 3D rendering, and image generation components of our pipeline each demanded different GPU hardware configurations to ensure efficient experimentation and testing. Model training was conducted on *Google Colab* using an **NVIDIA Tesla T4 GPU (16 GB VRAM)**. The increased VRAM capacity enabled the use of larger batch sizes and higher input resolutions, which contributed to more stable and effective training.

In contrast, image generation and 3D rendering were performed locally using **NVIDIA RTX 3070** and **RTX 4060 Laptop** GPUs (both equipped with 8 GB of VRAM). Despite the lower memory capacity, we found that 8 GB was sufficient for running the required *Stable Diffusion XL* models. VRAM efficiency was further improved by leveraging optimization nodes such as `FreeU_V2`, as well as by using smaller batch sizes during generation.

For the *Blender*-based 3D rendering, these GPUs also proved capable. Render times were significantly reduced—from several minutes to under one minute per

frame, aided through the intelligent use of denoising algorithms and optimizing Cycles settings such as lower light bounce counts. Additionally, these local GPUs were essential for executing custom Python scripts, performing depth-map generation, and managing *Blender* rendering workflows, all of which required a degree of flexibility and control over a local development environment not available in cloud-based ones.

4.2 Software & Frameworks

All experiments were implemented in `Python 3.10`, using a combination of deep learning libraries, generative AI platforms, and 3D rendering tools. The development environment spanned both cloud-based and local systems, depending on the resource demands of each pipeline stage. Many of the tools and platforms mentioned below have already been introduced in detail in Chapter 3, where their role in the pipeline is discussed more thoroughly.

Model training was conducted on *Google Colab*, which provided access to **NVIDIA TESLA T4** GPUs. The training pipeline was built using `PyTorch 2.6` and `torchvision 0.21`, with supporting libraries such as `NumPy`, `Pandas`, `PIL`, `matplotlib`, and `tqdm` used for preprocessing, visualization, and experiment management. Training runs were monitored using the *Weights & Biases (W&B)* API, which tracked metrics including Accuracy, Loss, Precision, Recall, and F1 Score.

Image generation was carried out locally using the *ComfyUI* platform, a node-based GUI for executing *Stable Diffusion* workflows. As previously detailed in Section 3.5, we tested both `SD 1.5` and `SDXL` variants, ultimately selecting the `Juggernaut XL` checkpoint due to its superior output quality. To improve visual fidelity without increasing VRAM usage, we employed the `FreeU-V2` node with optimized parameters. Prompt engineering was automated through the `OpenAI GPT-4o API`, which generated positive and negative prompts based on image content.

3D rendering was executed using `Blender 4.0`, with depth maps generated via the `DepthAnything V2` model in *ComfyUI* and applied through Blender' s `texture node` system to deform planar meshes. This process, outlined in Section 3.3, allowed for realistic surface generation and lighting simulation using the `Cycles` renderer and `HDRI` lighting. The full rendering workflow was automated using Blender's Python scripting API to batch-process image-depth pairs and export outputs.

All package dependencies were managed using `pip`, with key libraries specified and version-pinned in a `requirements.txt` file to ensure reproducibility across systems.

4.3 Datasets and Evaluation Splits

Three progressively augmented versions of the cocoa pod dataset were used during experimentation to evaluate model performance across different data volumes and

diversity levels. The datasets were preprocessed to maintain a consistent resolution and format, and all models were evaluated using the same fixed stratified k-fold cross validation method across datasets.

- **Dataset 1 – Original (Control)**: This baseline dataset consisted of the 65 un-augmented RGB images, evenly split between the *Local* and *Hybrid* classes. This was used to establish a performance baseline for all candidate models in data-scarce conditions. Results on this dataset informed the selection of models for subsequent evaluations.
- **Dataset 2 – Classical Augmentation**: This dataset contained 325 images generated by applying standard augmentation techniques (random rotation, horizontal flipping, and random resized cropping) Table 3.3 to the original set. This tested how classical augmentation alone affected model generalization. Only the top-performing models from Dataset 1 were evaluated on this expanded dataset.
- **Dataset 3 – Semantic + Classical Augmentation**: This final dataset comprises approximately 1,000 images and incorporates both classical augmentations and additional synthetic samples generated via a semantic augmentation pipeline. These images were created externally via a separate image synthesis process described in Chapter 3. The best-performing model from Dataset 2 was used to evaluate performance on this final, fully augmented dataset.

Each dataset was evaluated using stratified k-fold cross-validation (with $k = 2$ and $k = 5$) to ensure generalizability of the results. The same model configuration, training hyperparameters, and random seed were maintained across all experiments to ensure consistency. Metrics including Accuracy, Precision, Recall and F1-score were computed per fold and averaged for final comparison.

4.4 Model Architectures and Training Procedure

To evaluate the effectiveness of our proposed data augmentation pipeline, multiple CNN architectures were trained and tested across the three constructed datasets. The primary models investigated included `ResNet50`, `ResNet18`, `MobileNetV2`, `MobileNetV3-Small`, and `SqueezeNeet`. All implemented using the *Torchvision* library and initialized with pretrained weights on *ImageNet*. These wide variety of models were selected for their efficiency and established performance in small to medium-sized classification tasks (per Table 4.1).

Model Architecture	Parameter Count
ResNet50	25.6 million
ResNet18	11.7 million
MobileNetV3-Large	5.4 million
MobileNetV2	3.5 million
SqueezeNet	1.25 million

Table 4.1: Model architectures and parameter Counts of the tested models.

All models received input images resized to *224×224 pixels*. This resolution is a widely adopted standard in computer vision, particularly for models trained on the *ImageNet* dataset. Resizing images to this dimension ensures that every model used will not have dimensional misalignment, allows for the use of pre-trained weights without architectual modification, helps standardize the input pipeline, and reduces the computational cost of training.

Each model was trained using a `cross-entropy loss` function and optimized with the `Adam` optimizer. Unless otherwise specified, the default configuration used a learning rate of **1e-3** with **1e-4** weight decay, batch size of **8**, and trained for up to **15 epochs** per fold.

To reduce training time and mitigate the risk of overfitting, we applied Transfer Learning[22, 16]. This technique leverages the pre-trained weights of models trained on large datasets, allowing us to adapt them to our specific task, even in data-scarce scenarios where training from scratch would be impractical.

In our initial experiments, we adopted a consistent strategy across all candidate models. First, all model layers were frozen to preserve their learned representations. Then, we selectively unfroze the final classification layer, either retaining the original configuration or enhancing it with an additional dropout layer to further combat overfitting. Early results from these experiments highlighted a clear limitation: larger models such as `ResNet50`, `ResNet18`, and `MobileNetV3-Large` tended to overfit rapidly due to the small size of our dataset. These findings prompted us to prioritize more lightweight architectures with fewer trainable parameters for subsequent evaluations.

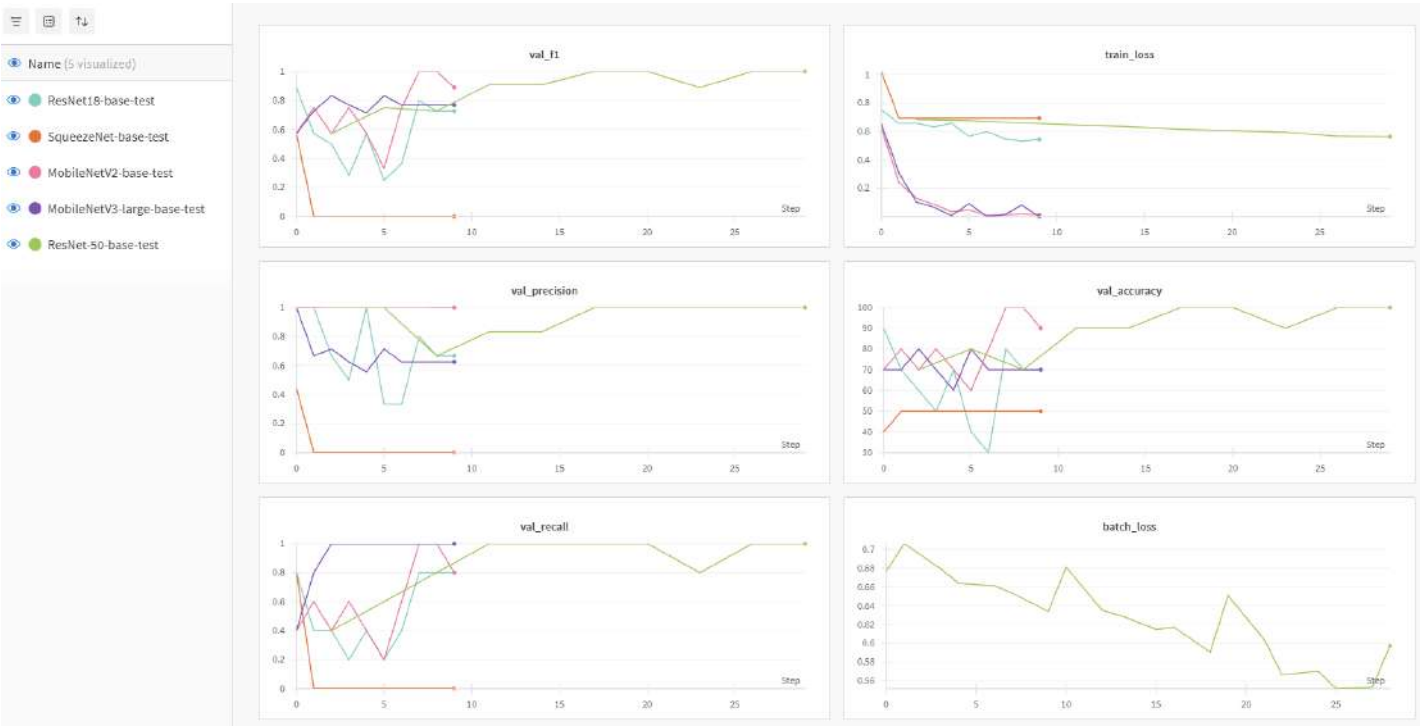


Figure 4.1: Initial results of model testing on base dataset.

Initial baseline tests were conducted with only a small **10** epoch preliminary run. As shown in Fig.4.1 the individual model scores across out testing metrics are quite noisy. One example being the turquoise coloured `RestNet18-base-test` in which we see wild fluctuations in our F1 scores, accuracy, precision, etc... This showcases a classic case of overfitting for most of the models in the tests. Looking at the models used (Table 4.1), we arrived at a set of requirements for achieving passable results.

- Avoid using large models for this style of dataset. Anything over 15 million seems to exacerbate the rate at which overfitting occurs.
- Varying the learning rate during the epoch gives us better results and stops overfitting from occuring as quickly.
- More data is needed to tell realistically how our models perform and generalize once trained on our dataset.

With this in mind, one medium weight model (`ResNet18`) and one light weight model (`MobileNetV2`) were selected for additional data augmentation training. A new dataset was generated using common data augmentation techniques (Table 3.3). transforming our original dataset structure of 38 training, 27 test, 65 total into 190 augmented training, 135 augmented test, 325 total.

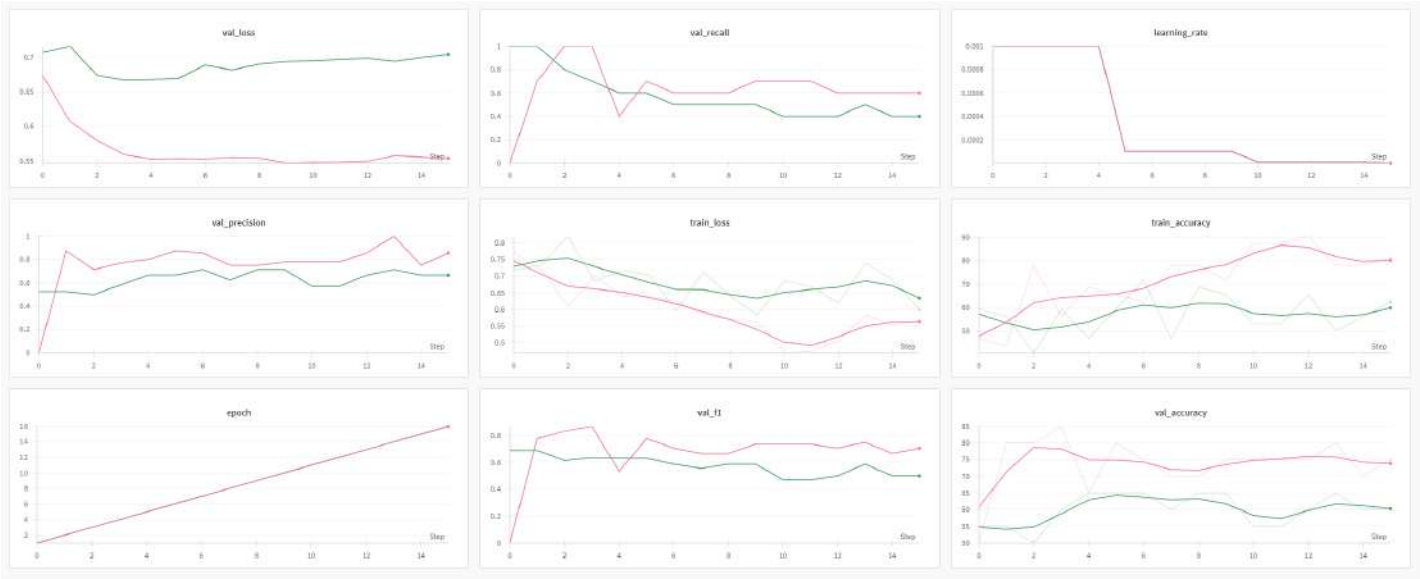


Figure 4.2: Results of testing MobileNetV2 (pink) and ResNet18 (green) with augmented training and testing dataset.

As we can see in Fig.4.2, when run on additional data, the model results begin to smooth out to a more consistent training and validation curve. We can begin to see also that a large difference in results begins to occur between the two models chosen. Counterintuitively the `ResNet18` model while having more parameters, does not generalize as well to the small data size as the `MobileNetV2` model does. Because of this, we take the best model (`MobileNetV2`) and continue it forward in the tests.

As stated in Chapter 3.8, classical model training splits consisting of some percentage of the data broken in to training and testing splits is a less effective means of training a model in a data scarce environment. Because of this, our new found "best model" is then retrained using the stratified k-fold cross validation technique.

Additionally, as described in Chapter 3.8, we employ a progressive unfreezing strategy to gradually expose more of the model to training as dataset size increases. This approach enables the model to learn deeper and more domain-relevant features over time, rather than limiting learning to just the final classification layer.

In the case of the `MobileNetV2` model, we begin by freezing all layers except for the final classifier, to which we add a dropout layer with a rate of 0.7. While this dropout rate is relatively high, it is appropriate in data-scarce scenarios where overfitting is a significant risk. This configuration is used during early testing with smaller datasets (as outlined in Table 3.2) to reduce model capacity and focus on surface-level classification.

As our dataset grows and contains more varied and representative samples, we incrementally unfreeze additional layers to allow the model to better adapt to task-specific features. Specifically, we tested unfreezing the last 2 (Algorithm 1), and then last 5 (Algorithm: 2) layers of the feature extractor block before the classifier.

Algorithm 1 Full 2-Layer Model Adjustment & Layer Freezing Strategy

Input: Pretrained MobileNetV2 model
Output: Partially trainable model with custom classifier
Load `MobileNetV2` with pretrained `ImageNet` weights
▷ Freeze entire base network
for all parameters in `model` **do**
    `param.requires_grad` ← `False`
end for
▷ Unfreeze last 2 convolutional layers
for all parameters in `model.features[-2:]` **do**
    `param.requires_grad` ← `True`
end for
▷ Insert dropout and new linear layer before classification head
`model.classifier` ← `Sequential(Dropout(0.7), Linear(last_channel, 1))`
Move model to computation device (`CPU` or `GPU`)

Algorithm 2 Full 5-Layer Model Adjustment & Layer Freezing Strategy

Input: Pretrained MobileNetV2 model
Output: Partially trainable model with custom classifier
Load `MobileNetV2` with pretrained `ImageNet` weights
▷ Freeze entire base network
for all parameters in `model` **do**
    `param.requires_grad` ← `False`
end for
▷ Unfreeze last 5 convolutional layers
for all parameters in `model.features[-5:]` **do**
    `param.requires_grad` ← `True`
end for
▷ Insert dropout and new linear layer before classification head
`model.classifier` ← `Sequential(Dropout(0.7), Linear(last_channel, 1))`
Move model to computation device (`CPU` or `GPU`)

The final setting of 5 layers was chosen based on it offering consistent improvement in validation accuracy while maintaining stable training loss, suggesting that the model was learning meaningful features without overfitting.

This is supported by the structure of MobileNetV2 itself, which is composed of inverted residual blocks ordered by increasing abstraction. Unfreezing the final 5 layers allows adaptation of the higher-level, more task-relevant feature maps, while still keeping the earlier, general-purpose filters intact. This strikes a balance between training for the specific dataset requirements and generalization. Enabling the network to specialize to our dataset without compromising the stability offered by the pre-trained weights.

4.5 Evaluation Metrics

To properly evaluate both the classification performance and quality of augmented data, a combination of traditional machine learning techniques and perceptual similarity metrics were used. These include Accuracy, Precision, Recall, F1-score, LPIPS (Learned Perceptual Image Patch Similarity), and CLIP similarity.

4.5.1 Classification Metrics

In binary classification problems one of the most common ways to capture model performance is with a confusion matrix. This matrix breaks down predictions that the model makes into four categories:

- **True Positives (TP)** The model correctly predicts the positive class (1 - Hybrid).
- **True Negatives (TN)** The model correctly predicts the negative class (0 - Local).
- **False Positives (FP)** The model incorrectly predicts the positive class (1 - Hybrid).
- **False Negatives (FN)** The model incorrectly predicts the negative class (0 - Local).

These components form the foundation for computing the evaluating metrics used in this study, as detailed below:

- **Accuracy** measures the percentage of correct predictions over the total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision** showcases how many of the predicted positives were actually correct:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** measures how many of the actual positives were correctly predicted:

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score** is the harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

These metrics are commonplace for most model evaluations and help showcase model robustness or potential issues.

4.5.2 Perceptual and Semantic Similarity Metrics

Two perceptual metrics were used in our study to assess the semantic and visual quality of our synthetically generated images:

- **LPIPS (Learned Perceptual Image Patch Similarity)** quantifies perceptual similarity between image pairs using deep features extracted from pretrained neural networks (e.g., *VGG* or *AlexNet*) causing LPIPS to correlate more with human judgment of visual similarity. Lower LPIPS values indicate that augmented images preserve the low-level and mid-level structural characteristics of real samples [31]. A score of 0.0 for example would indicate a perfect match, whereas as score of 0.6 or higher would indicate distinct visual differences.

- **CLIP Similarity** uses embeddings from OpenAI' s *CLIP* model, which jointly learns image-text representations. CLIP similarity measures how semantically aligned a generated image is with its corresponding class identity or prompt. A higher cosine similarity between image and text embeddings indicates stronger semantic consistency [23].

These metrics are particularity meaningful in the case of real world data scarce scenarios. Traditional metrics such as accuracy or F1 allow for the validation of the models performance. However, LPIPS and CLIP similarity allow us to assess whether the augmented / synthetic images maintain visual realism and class specific semantic content. Combined together, these metrics can provide a comprehensive overview of both model generalization and image/data quality in a low-data environment.

Chapter 5

Results

This section presents the outcome of training our best performing model from Chapter 4 on three unique datasets prepared through the augmentation pipeline described in Chapter 3. Performance metrics across datasets, and perceptual and semantic similarity scores are compared to evaluate the effectiveness of combining classical and generative augmentation techniques.

5.1 Model Performance Across Datasets

Test	**Specifications**	**Dataset Size**
1	Base model Classifier layer unfrozen No dataset augmentation	Total: 65
Stratified K-Fold Cross Validation Results		
Fold	Metrics	
Fold 1	Accuracy: 84.21% Precision: 72.73% Recall: 100.00% F1: 84.21%	
Fold 2	Accuracy: 94.74% Precision: 88.89% Recall: 100.00% F1: 94.12%	
K-Fold Avg	Accuracy: **89.47%** Precision: **80.81%** Recall: **100.00%** F1: **89.16%**	
Final Evaluation on Held-Out Test Set		
Test Set Results	Accuracy: 70.37% Precision: 100.00% Recall: 52.95% F1: 69.23%	Test set size: 27

Table 5.1: MobileNetV2 Evaluation on Dataset 1.

Test	**Specifications**	**Dataset Size**
1	Base model Classifier + 5 layers unfrozen Classical augmentation	Total: 325
Stratified K-Fold Cross Validation Results		
Fold	Metrics	
Fold 1	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
Fold 2	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
K-Fold Avg	Accuracy: **100.00%** Precision: **100.00%** Recall: **100.00%** F1: **100.00%**	
Final Evaluation on Held-Out Test Set		
Test Set Results	Accuracy: 80.74% Precision: 100.00% Recall: 69.41% F1: 81.94%	Test set size: 135

Table 5.2: MobileNetV2 Evaluation on Dataset 2.

Test	**Specifications**	**Dataset Size**
1	Base model Classifier + 5 layers unfrozen Classical + synthetic augmentation	Total: 1000
Stratified K-Fold Cross Validation Results		
Fold	Metrics	
Fold 1	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
Fold 2	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
Fold 3	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
Fold 4	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
Fold 5	Accuracy: 100.00% Precision: 100.00% Recall: 100.00% F1: 100.00%	
K-Fold Avg	Accuracy: **100.00%** Precision: **100.00%** Recall: **100.00%** F1: **100.00%**	
Final Evaluation on Held-Out Test Set		
Test Set Results	Accuracy: 98.32% Precision: 96.81% Recall: 99.92% F1: 98.34%	Test set size: 500

Table 5.3: MobileNetV2 Evaluation on Dataset 3.

5.2 Cross-Validation & Held-Out Test Performance

To evaluate the generalizability of our model across training subsets, we conducted 5-fold stratified cross-validation on the augmented datasets. Tables 5.1,5.2,5.3 summarize the performance metrics for our three unique datasets.

Dataset 1 consisted of the original 65 images, divided into "**Local**" and "**Hybrid**" cocoa classes. Additionally, only the final classifier layer was unfrozen for training and augmentation was applied.

As shown in Table 5.1, the results suggest that the model was able to memorize class-distinguishing attributes due to their visual clarity and class separability. Particularly as Recall reached 100% across both folds. However, more moderate precision scores point to a chance of false positives occurring. Suggesting that the classifier occasionally mislabelled the opposite class despite detecting all instances of the class correctly.

When tested against the held-out set of 27 previously unseen images, model performance decreased.

- **Accuracy:** 70.37%
- **Precision:** 100.00%
- **Recall:** 52.95%
- **F1 Score:** 69.23%

The declines in F1 score highlights the models problems with overfitting to the small training dataset. Although the model retained a perfect precision score (zero false positives), it failed to detect nearly half of the actual positive class instances. Showcased by a low recall of 52.95%. These results reinforce the notion that training on such a small dataset without augmentation severely limits a models generalizability.

Dataset 2 incorporates traditional augmentation techniques as per Table 3.3. Here the augmentation transformed the dataset from 65 to 325 images, increasing the class diversity whilst preserving the semantic integrity of all images. The final 5 layers of the `MobileNetV2` model were unfrozen as the larger dataset enabled for this to occur, allowing for the network to adapt more to the given input.

As can be seen in Table 5.2, the results suggest that the classical augmentation techniques were highly effective allowing the model to generalize core class features. The flawless performance on the training data indicates that the class-to-class variance was learned well enough to eliminate misclassification of training splits.

When tested on the held-out test set of 135 previously unseen images. We see the following results:

- **Accuracy: 80.74%**
- **Precision: 100.00%**
- **Recall: 69.41%**
- **F1 Score: 82.4%**

Though accuracy and F1 scores improved substantially over Dataset 1, a wide gap still remains between the cross-validation and real-world performance values. Whilst the model is able to maintain perfect precision (no false positives were present). Recall dropped to 69.41%, indicating that many true instances were still missed.

These findings confirm that classical augmentation techniques significantly improved model generalization. Comparing with Dataset 1, the increase in F1 scores and Accuracy (+12.71% and +10.37% respectively) suggests that classical augmentation alone contributes positively to overcoming the inevitable overfitting spiral and data scarcity bottleneck. The drop in recall also reveals that new methods (ones that introduce new semantic and structural diversity) may be required to achieve a truly generalizable model.

Dataset 3 represents the culmination of our full augmentation pipeline, combining our original data, classical augmentation techniques, and our proposed semantic augmentation strategy based on depth-aware rendering and generative AI synthesis. The dataset was expanded to approximately 1,000 images, with a balanced representation of both the **Local** and **Hybrid** classes. Five layers of the MobileNetV2 architecture were unfrozen to allow for even deeper model adaptation.

As seen in Table 5.3, the consistent performance strongly indicates that the semantic and classical augmentations combine together to provide the model with both sufficient inter-class variation and enhanced data richness, allowing for it to effectively learn and distinguish class features across training folds.

When tested on the held-out test set of 500 images, the below results were achieved:

- **Accuracy:98.32%**
- **Precision: 96.81%**
- **Recall: 99.92%**
- **F1 Score: 98.34%**

These scores further confirm the model's generalizability. Yielding the best results among all of our dataset variations.

Unlike the previous two datasets, where there were trade-offs between recall and precision, Dataset 3 maintained high scores across all major metrics. This near-perfect generalization to previously unseen data samples demonstrates the strength of the synthetic pipeline in creating semantically aligned but also diverse data. Effectively mimicking real world variability. The increase in Accuracy (+27.95% vs Dataset 1 and +17.58% vs Dataset 2) and F1 Scores (29.11% vs Dataset 1 and 16.4% vs Dataset 2) on the held out test set confirms that the proposed pipeline enables robust model generalization in low-data scenarios. Also, the balance between precision and recall (which is often a challenge in low data scenarios) suggests that

the semantic content generated from depth aware rendering and prompt-enabled SD models was both accurate and class-consistent.

In summary, the progression from Dataset 1 to Dataset 3 demonstrates a clear improvement over each iteration, as the datasets become both quantitatively larger and qualitatively more diverse through targeted augmentation techniques and our own custom semantic augmentation. The model generalization improves substantially. Whilst classical augmentation techniques help provide substantial gains, it is the integration with our semantically meaningful data (generated through depth aware images) that yields truly transformative improvements in both recall and overall predictive balance. These findings not-only validate the effectiveness of our depth-aware generative pipeline. But also reinforce the broader idea that in data-scarce domains, the use of intelligent augmentation may rival or even surpass the value of raw data acquisition.

5.3 LPIPS & CLIP Evaluation

Class	CLIP Similarity	LPIPS
Local	0.866	0.578
Hybrid	0.874	0.579

Table 5.4: CLIP Similarity and LPIPS Scores by Class.

To form a better understanding of the perceptual and semantic realism of the synthetically generated images, we computed two widely used metrics. LPIPS (Learned Perceptual Image Patch Similarity) and CLIP similarity. These two metrics provide insight beyond traditional classification accuracy, helping to quantify how visually and semantically aligned our generated samples are to the real counterparts.

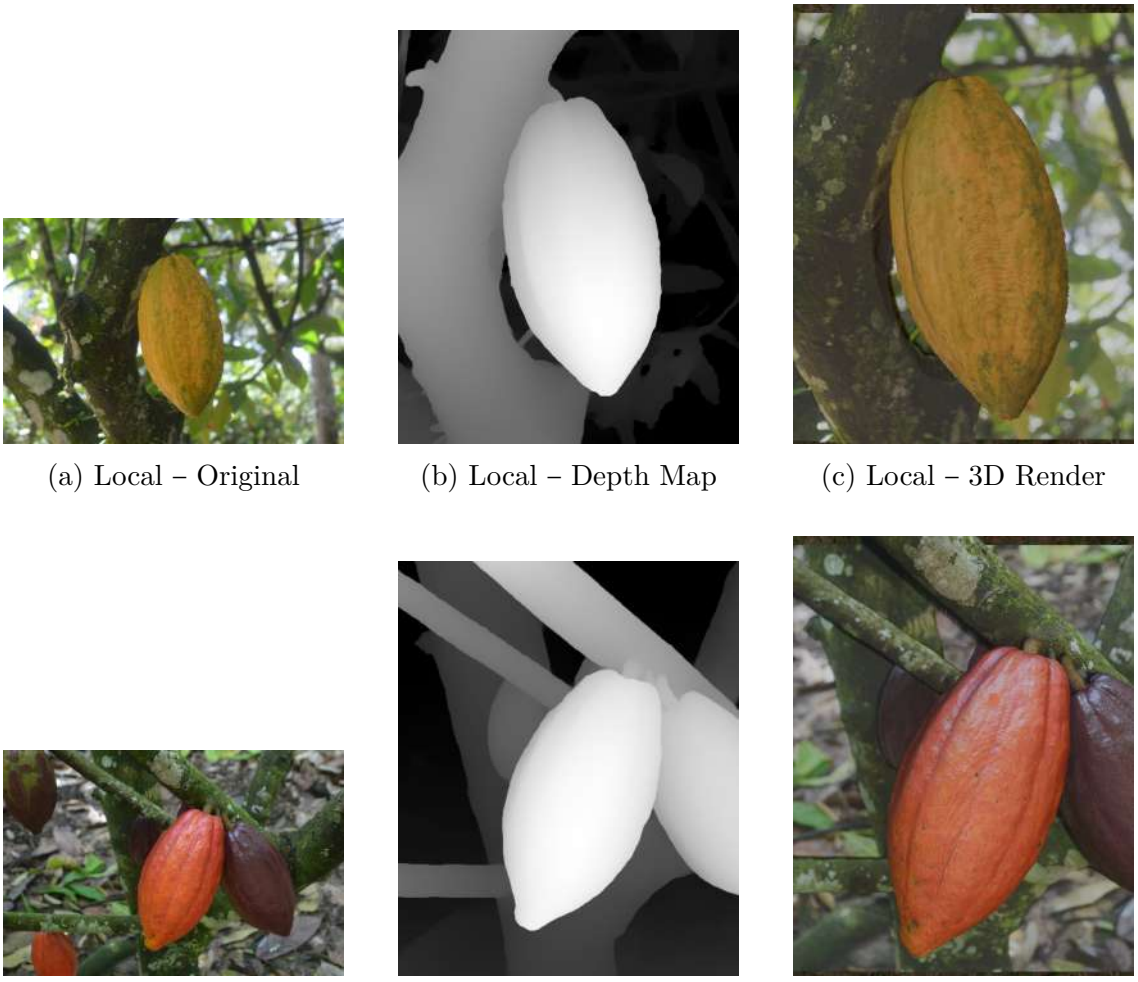
CLIP similarity was used to evaluate the semantic consistency of our newly generated images and its associated class prompt. This metric assesses how well the model's image embeddings align the natural language descriptions. The Local class yielded a CLIP score of **0.866**, whilst the Hybrid class achieved a CLIP score of **0.874** (Table 5.4). These scores demonstrated that the prompt-guided generation using GPT-4o for prompt generation an Stable Diffusion for image synthesis effectively preserved the semantic attributes tied to each cocoa pod type.

LPIPS, which captures perceptual dissimilarity as perceived by a deep neural network, was used to measure the visual closeness of our synthetic images to their source images. Lower LPIPS scores indicated higher perceptual similarity. The average LPIPS score for Local and Hybrid classes were **0.578** and **0.579** respectively (Table 5.4). These moderate values suggest that the rendered and generated images still preserved much of the perceptual structure of the original, despite geometric

deformation in the 3D rendering stage and random noise based mutation in the image generation phase.

Combined, these results support the effectiveness of our hybrid augmentation pipeline. The synthetic images were not only perceptually aligned with their real counterparts. But also retained strong semantic similarity. Making them suitable for use in fine-grained classification tasks with limited baseline data.

5.4 Qualitative Evaluation



(a) Local – Original (b) Local – Depth Map (c) Local – 3D Render

(d) Hybrid – Original (e) Hybrid – Depth Map (f) Hybrid – 3D Render

Figure 5.1: Transformation visualization for both Local and Hybrid cocoa pod classes. Beginning with the original image, followed by a depth map generated with DepthAnything-V2, and concluding with a 3D-rendered image in Blender. These augmentations simulate novel camera viewpoints and lighting while preserving semantic content.

Figure 5.2: Sample of generated image strips for both **Local** and **Hybrid** classes. The 3D rendered images from Fig. 5.1 are taken and noised using the pipeline from Fig. 3.7 to generate new variations. Each row contains 5 semantically aligned synthetic images which help bolster our semantic dataset.

Taken together, the qualitative results underscore the visual cohesion and semantic consistency of the augmented images produced by our pipeline. Figures 5.1 & 5.2 demonstrate the structural integrity of class-specific features (like pod shape, colour, and surface texture) through each stage of the transformation pipeline as well as the the diversity and richness of the final outputs, showcasing the granular variation in multiple images generated from the same source. While some generative artifacts are present among the images generated, their occurrence is sparse and largely non-disruptive to each image's class identity. These visuals further affirm that our augmentation process produces meaningful variability without compromising semantic fidelity, an essential requirement in low-data scenarios such as ours.

Chapter 6

Conclusion

This study introduced a novel generative semantic augmentation pipeline that leverages depth-aware rendering and diffusion-based synthesis to enhance dataset size and diversity in low-data scenarios. Unlike conventional augmentation techniques that rely on basic geometric transformations, our approach produces semantically enriched samples that preserve class integrity while introducing new viewpoints, textures, and lighting conditions. Across all evaluation metrics, the proposed pipeline consistently outperformed both the baseline and classically augmented models. Notably, the final combined dataset achieved an F1 score of 98.34% on a held-out test set, indicating strong generalization capabilities. Qualitative visualizations and perceptual similarity scores (LPIPS and CLIP similarity) further confirmed the semantic and perceptual coherence of the generated samples.

These findings support the use of semantic generative augmentation as a powerful technique in domains where labelled data is scarce, expensive, or logistically difficult to collect — such as agricultural monitoring, biodiversity assessment, or medical imaging. The modular nature of the proposed pipeline also enables flexible adaptation to other vision-based tasks.

However, beyond technical results, it is essential to consider the practical value and real-world deployability of this approach. For example, envisioning a real use case: a cocoa farmer in Sulawesi who wishes to deploy a mobile app that can classify cocoa pods as either local or hybrid. While the final classification model could likely be distilled or quantized to run efficiently on a smartphone, (a practice increasingly common in mobile ML) the upstream processes (depth-based rendering, image generation, and model training) require access to GPU hardware and some degree of technical intervention. Fortunately, these steps can be offloaded to cloud services, where generative image creation and model training can be performed remotely and relatively affordably (e.g., 1–2 per GPU-hour). In this case, the farmer could submit a small number of labelled images to a centralized server, which would process and augment the data, train the model, and return a lightweight mobile-compatible version. Potentially all within a few hours or less, depending on infrastructure. The proposed method therefore has strong potential not only as a research contribu-

tion but also as the foundation for scalable, user-centered solutions in low-resource settings.

Chapter 7

Future Works

While this study has demonstrated the viability of semantic data augmentation using depth-aware rendering and generative models, several promising directions remain for future exploration:

1. **Integration of 3D Scanning Technology**
   A potential enhancement to the current pipeline involves the replacement of estimated depth maps with data obtained from actual 3D scans of plant specimens. The use of structured light or photogrammetry-based 3D scanning could help enable a more accurate reconstruction of the original object into a 3D mesh, thereby allowing for the production of more realistic renderings with physically grounded geometry and lighting. This would help reduce the differences between synthetic and real-world images even further.

2. **Image-to-Video Models**
   Another exciting and recent avenue worth exploring involves extending the current augmentation into the temporal domain. Through recent advances in text-to-video and image-to-video models, it would be possible to simulate dynamic environmental conditions—such as changing lighting, camera motion, or natural plant movement. These variations could further enrich the training distribution and improve classification performance. Moreover, the temporal coherence introduced through video sequence generation could open the pipeline to new applications in activity recognition or video-based agricultural monitoring systems.

3. **Testing on Multi-Class Classification Tasks**
   This work focused on binary classification. However, many real-world classification problems in agriculture, ecology, and healthcare involve multiple classes—such as different plant diseases, species, or developmental stages. Future evaluations should assess the effectiveness of semantic augmentation in multi-class settings, where inter-class boundaries are more nuanced. Performance in these more complex scenarios will be essential for validating the generalizability and scalability of the proposed augmentation strategy.

Acknowledgements

I would like to express my sincere gratitude to everyone who supported my studies and research throughout the course of my Master's degree.

First and foremost, i would like to thank my supervisor, Professor Mario Köppen, whose guidance and insightful suggestions helped transform my ideas into reality. His feedback was invaluable in shaping the direction and quality of this research.

I am also deeply grateful to Sajjad Dadkhah for encouraging me to pursue a Master's in Computer Science and for believing in my potential - long before I believed it myself.

I furthermore extend my thanks to my fellow members of the Human Centered Computing Lab. Your collaboration, discussions, and friendship greatly enriched both my academic and research experience.

Finally, I am profoundly thankful to my friends and family for their unwavering support and encouragement throughout this journey. Without them, none of this would have been possible.

References

[1] Lukas Biewald. Experiment tracking with weights and biases. `https://www.wandb.com`, 2020. Accessed July 2025.

[2] Blender Online Community. Blender - a 3d modelling and rendering package, 2023. Accessed: 2025-07-28.

[3] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[4] Sophia Clark and Ethan Cooper. Synthetic image datasets with stable diffusion and data augmentation.

[5] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[6] ComfyUI developers. ComfyUI: a node - based gui for stable diffusion. Open - source software, 2025. Version 0.3.40, retrieved from GitHub, accessed July 2025.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[9] Lisa Dunlap, Alyssa Umino, Han Zhang, Jiezhi Yang, Joseph E Gonzalez, and Trevor Darrell. Diversify your vision datasets with automatic diffusion-based augmentation. *Advances in neural information processing systems*, 36:79024–79034, 2023.

[10] Haoyang Fang, Boran Han, Shuai Zhang, Su Zhou, Cuixiong Hu, and Wen-Ming Ye. Data augmentation for object detection via controllable diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1257–1266, 2024.

[11] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE, 2018.

[12] Yunxiang Fu, Chaoqi Chen, Yu Qiao, and Yizhou Yu. Dreamda: Generative data augmentation with diffusion models. *arXiv preprint arXiv:2403.12803*, 2024.

[13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[14] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574*, 2022.

[15] Khawar Islam, Muhammad Zaigham Zaheer, Arif Mahmood, and Karthik Nandakumar. Diffusemix: Label-preserving data augmentation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27621–27630, 2024.

[16] Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. Large language models understand and can be enhanced by emotional stimuli. *arXiv preprint arXiv:2307.11760*, 2023.

[17] Shaobo Lin, Kun Wang, Xingyu Zeng, and Rui Zhao. Explore the power of synthetic data on few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 638–647, 2023.

[18] Eugenio Lomurno, Matteo D'Oria, and Matteo Matteucci. Stable diffusion dataset generation for downstream classification tasks. *arXiv preprint arXiv:2405.02698*, 2024.

[19] Bruce G. Marcot and Anca M. Hanea. What is an optimal value of k in k-fold cross-validation in discrete bayesian network analysis? *Computational Statistics*, 36(3):2009 – 2031, June 2020.

[20] Carlos Medel-Vera, Pelayo Vidal-Estévez, and Thomas Mädler. A convolutional neural network approach to classifying urban spaces using generative tools for data augmentation. *International Journal of Architectural Computing*, page 14780771231225697, 2024.

[21] OpenAI. Chatgpt. `https://chat.openai.com`, 2024. Mar 2024 version, accessed July 2025.

[22] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[24] Fazle Rahat, M Shifat Hossain, Md Rubel Ahmed, Sumit Kumar Jha, and Rickard Ewetz. Data augmentation for image classification using generative ai. *arXiv preprint arXiv:2409.00547*, 2024.

[25] Google Research. Google colaboratory. `https://colab.research.google.com`, 2023. Accessed July 2025.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[27] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. *arXiv preprint arXiv:2302.07944*, 2023.

[28] Shitao Xiao, Yueze Wang, Junjie Zhou, Huaying Yuan, Xingrun Xing, Ruiran Yan, Shuting Wang, Tiejun Huang, and Zheng Liu. Omnigen: Unified image generation. *arXiv preprint arXiv:2409.11340*, 2024.

[29] Ziqi Yin, Hao Wang, Kaito Horio, Daisuike Kawahara, and Satoshi Sekine. Should we respect llms? a cross-lingual study on the influence of prompt politeness on llm performance. In *Proceedings of the Second Workshop on Social Influence in Conversations (SICon 2024)*, pages 9–35, 2024.

[30] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.

[31] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[32] Chenyu Zheng, Guoqiang Wu, and Chongxuan Li. Toward understanding generative data augmentation. *Advances in neural information processing systems*, 36:54046–54060, 2023.

[33] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[34] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. *arXiv preprint arXiv:1711.00648*, 2017.