INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER, UK

University of Westminster, Coat of Arms

GANSAN: Using Neural Architecture Search with Generative Adversarial Networks for Image Restoration

A Project Specification and Prototype Design Document by

Mr. Ruchira Sahabandu

w1790179 | 2019724

Supervised by

Mr. Guhanathan Poravi

Submitted in partial fulfilment of the requirements for

the BSc (Hons) Computer Science degree at the University of Westminster.

February 2023

ABSTRACT

Restoration of degraded images is a naturally ill-posed problem which has been an open research problem for many years in the domain of computer vision. Whilst analogue images were primarily affected by physical degradations, digital imaging systems still struggle with visual corruptions and degradations such as blurring, image noise, low resolution, and haze. Since imaging systems are used in many other cutting-edge applications such as autonomous vehicles, surveillance systems and robotics, it is essential to develop efficient and robust image restoration systems capable of dealing with different types of visual degradations.

Whilst many different approaches to restoring degraded images exist, recent advances in deep learning have seen the emergence of complex techniques such as Generative Adversarial Networks being applied towards this problem. In particular, the proposed GANSAN is a novel approach towards optimising such restoration models using the power of Neural Architecture Search by focusing on the Generator network architecture in particular. This U-Net based neural network is optimised to restore degradations of any input image by passing it through the stack of down-sampling and up-sampling layers whose features and hyperparameters are further fine-tuned during the adversarial training process.

GANSAN is capable of restoring degraded images using a Pix2Pix GAN architecture as its base. Experiments conducted so far have shown that the model is generalizable and could achieve PSNR scores as high as 29.17 dB and SSIM scores as high as 0.85 after training for 500 epochs on the DIV2K dataset. Optimizing the U-Net based generator network of the image restoration GAN model using NAS is a novelty introduced in this research.

Keywords: Image restoration, Image denoising, Image deblurring, Computer Vision, Deep Learning, Data Science

Subject Descriptors

- Computing methodologies → Artificial intelligence → Computer vision
- Computing methodologies → Machine learning → Machine learning approaches → Neural networks
- Computing methodologies → Machine learning → Machine learning approaches → Bio-inspired approaches → Genetic algorithms
- Computing methodologies → Machine learning → Machine learning approaches → Bio-inspired approaches → Generative and developmental approaches

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Acronym	**Description**
GAN	Generative Adversarial Network
NAS	Neural Architecture Search
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
CAE	Convolutional Autoencoder
SOTA	State of the Art
ML	Machine Learning
DL	Deep Learning
IDE	Integrated Development Environment

1 INTRODUCTION

1.1 Chapter Overview

This chapter aims to present the reader with an outline of the problem domain of image restoration via deep learning approaches such as Generative Adversarial Networks (GANs), the current state of research in this area of computer vision/image processing and their shortcomings, along with a brief overview pertaining to the author's proposal to improve such image restoration GAN models by leveraging Neural Architecture Search (NAS) as a novel approach towards designing effective and optimized image restoration models with increased efficiency.

1.2 Problem Domain

1.2.1 Image Restoration

Image restoration is a well-established problem in low-level computer vision, where the aim is to recover or hallucinate a high-fidelity sharp image from a degraded (e.g., blurred, or low-resolution) input observation. This has been a long-standing area of research interest in the digital imaging space with attempted solutions ranging from simple mathematical models during the early days, to more advanced and complex deep learning (DL) approaches using various types of neural network (NN) based models and algorithms developed within the past decade. (Su, Xu and Yin, 2022)

Image restoration has been identified as a requirement in a multitude of domains such as autonomous driving and navigation, video surveillance systems, medical imaging, astronomy, and microscopy where it is common to receive degraded inputs due to environmental and other external factors which cannot be controlled. (Hu et al., 2020; Su, Xu and Yin, 2022)

1.2.2 Generative Adversarial Networks

GANs, originally introduced for image generation, is an interesting and rapidly evolving area of research within the DL space and has received wide attention across multiple problem domains, with arguably the most significant impact in the field of computer vision and image processing (Wang, She and Ward, 2022). The basis of a typical GAN architecture consists of a generator network G and a discriminator network D, where the generator attempts to generate new samples based on the training dataset whilst the discriminator attempts to estimate the probability that a sample is from the training data rather than G (Goodfellow et al., 2014).

In the context of image restoration, the application of adversarial training via GAN models has proven to achieve better visual performance and efficiency with tasks such as image deblurring and super-resolution (Fu et al., 2020; Zhang et al., 2022). However, similar to other DL based models, traditional GANs make use of manually designed network architectures created through trial-and-error. The difficulty of creating optimal network architectures utilized in GAN models is pronounced further due to the possibility of encountering unique issues such as mode-collapse during training (Goodfellow, 2017).

1.2.3 Neural Architecture Search

NAS, which is a subfield of AutoML, focuses on automating the process of architecture engineering for NNs and is seen as a logical next-step of automating machine learning (Elsken, Metzen and Hutter, 2019). The basis of NAS could be divided into 3 main components: search space, search strategy and performance estimation/evaluation strategy and could be employed to optimize a multitude of DL models based on NNs.

NAS has been successfully utilized in designing efficient image restoration models based on comparatively simple approaches such as Convolutional Autoencoders (CAE), and Convolutional Neural Networks (CNN) /U-Net models (Suganuma, Ozay and Okatani, 2018; Chen et al., 2020; Arican et al., 2022). Furthermore, NAS has also been successfully applied towards the automation of GAN architecture search for computer vision tasks such as conditional and unconditional image generation and image segmentation (Gong et al., 2019; Gao et al., 2020; Ganepola and Wirasingha, 2021a).

1.3 Problem Definition

The core aim of image restoration is to recover clean, latent images from degraded inputs (e.g., blurred/low resolution images, images with high noise) via methods such as deblurring, super-resolution and denoising. Due to the infinite possible mappings between a degraded observation and its corresponding restored image, image restoration is defined as an ill-posed inverse problem and is still considered to be challenging to tackle (Su, Xu and Yin, 2022).

Conventional approaches towards solving this problem have been based on probabilistic models that solve inverse problems. However, within the past decade, rapid developments in the DL space as well as its various applications in the domain of computer vision has given rise to the introduction of various image restoration models based on Neural Networks (NNs). More recently, the popularity of GAN models in the field of image generation has led to their

adoption towards image restoration tasks where a notable increase in performance over existing generic NN based methods has been observed, raising the bar further for State of The Art (SOTA) performance. (Su, Xu and Yin, 2022)

Whilst much progress has been observed in other computer vision tasks (e.g., image generation, image segmentation) employing such GAN models, by optimizing the underlying network architectures using NAS no such efforts have been made towards improving GAN models used for image restoration purposes (Gong et al., 2019; Gao et al., 2020; Ganepola and Wirasingha, 2021a).

1.3.1 Problem Statement

Image restoration, being a well-researched area in classic low-level computer vision consist of problems ill-posed by nature, giving rise to the development of a multitude of image restoration networks, many of which employ adversarial training via GAN models which are built upon NN architectures that are difficult to design manually, to be as optimised and efficient as possible.

1.4 Aims and Objectives

1.4.1 Aims

This research project aims to design, develop, and evaluate a novel approach towards generating an efficient and optimized image restoration network architecture using Neural Architecture Search and Generative Adversarial Networks, to efficiently restore degraded images (E.g.: Deblurring, Super-resolving, Denoising).

To elaborate further, the aim of this project is to develop a novel system based on NAS to search for and generate efficient and optimized network architectures for GAN models used for image restoration purposes. Ideally these would be capable of rivalling and even surpassing such existing human-designed network architectures, some of which employ adversarial processes with complicated loss functions with high resource requirements for training.

1.4.2 Research Objectives

The following table represents a list of atomic activities that are needed to be carried out to achieve the aim of this research.

Research Objectives	Explanation	Learning Outcome	Research Question

Problem Identification	**RO1 –** Brainstorm within domains of interest, with the support of existing literature to identify research areas with existing problems of interest. **RO2 –** Continue further reading into the selected problem domain to identify problems addressable via a research project. **RO3 –** Review all available options and decide on a problem with good research impact	LO1 LO4 LO8	RQ1
Literature Review	A thorough literature survey is conducted within the image processing domain to, **RO1 –** Gain insight into the applications of deep learning networks within the image-processing domain. **RO2 –** Identify the deep learning approaches used towards restoring degraded and blurred images. **RO3 –** Learn about the prominence for the increased usage of GAN based models when approaching image restoration tasks. **RO4 –** Understand the usage and impact of NAS in designing neural networks, its applications towards developing optimized image restoration networks. **RO5 –** Analyze and understand the integration of NAS as an architecture search method to design optimal GAN models.	LO1 LO4 LO5 LO8	RQ1 RQ2 RQ3 RQ4
Data Gathering and Requirement Analysis	Relevant data is gathered, and a thorough requirement analysis is conducted to, **RO1 –** Get an understanding on existing systems and identify the functional requirements of the system being developed. **RO2 –** Gain insights on the requirements and opinions of domain experts and researchers on the	LO1 LO2 LO3 LO6 LO8	RQ3 RQ4

	expectations for the proposed system via questionnaires and interviews. **RO3 –** Understand and define the core features and functionality expected of the prototype software by users through surveys.		
Design	Designing of the proposed image restoration approach by, **RO1 –** Designing a NAS based deep-learning neural network architecture for image restoration tasks. **RO2 –** Designing an optimized NAS based GAN model using such neural networks for image restoration tasks.	LO1 LO2 LO5 LO8	RQ1 RQ2 RQ3
Implementation	Development of the prototype using the proposed NAS based approach towards image restoration based on the architectures and approaches defined in the design phase to, **RO1 –** Effectively restore degraded images using the optimized models. **RO2 –** Build efficient and functional GAN models for such image restoration tasks.	LO1 LO2 LO5 LO7	RQ2 RQ3
Testing and Evaluation	Testing and evaluation of the developed prototype with feedback and input of domain experts to, **RO1 –** Test and verify the functionality of each component of the developed prototype. **RO2 –** Review the performance of the system when it comes to restoring various types of images. **RO3 –** Validate the effectiveness of the developed system in relation to the initially identified system requirements.	LO1 LO4 LO8	RQ4

Table 1-1: Research Objectives

1.5 Novelty of Research

1.5.1 Problem Novelty

Image restoration, being a fundamentally ill-posed problem in computer vision has received the attention of many researchers over the years (Su, Xu and Yin, 2022). This has led to the birth of a wide variety of image restoration methods, with contemporary development trends pushing such restoration models to be more powerful, but also more complex to manually architect. GANs is one such image restoration technique that has had a surge in popularity in recent years. However, such models are notorious for being difficult to successfully architect and require a lot of experience and expertise to optimise (Gao et al., 2020). At the time of this research, no attempts had been made to automate the process of architecting GAN models for image restoration.

1.5.2 Solution Novelty

The solution proposed in this work is to use Neural Architecture Search to automate the process of architecting the generator of a GAN model. Whilst this approach has been explored in other areas of computer vision, starting with the work of (Gong et al., 2019), it is yet to be applied for the purpose of image restoration, making it a novelty in this domain.

1.6 Research Gap

Grounded on the existing work, it is found that adversarial training (via GANs) helps to boost results of image restoration processes (Kupyn et al., 2018) compared to other lightweight deep learning approaches. Similarly, the experimentation of using NAS to create image restoration networks and deep image priors have proved to be successful and capable of surpassing the performance of most human-designed image restoration networks (Suganuma, Ozay and Okatani, 2018; Zhang et al., 2021; Arican et al., 2022). Furthermore, the integrated approaches of NAS and GANs have been experimented with for other image processing tasks such as unconditional image generation (Gong et al., 2019) and semantic image segmentation (Ganepola and Wirasingha, 2021a) and has proved to yield positive results capable of rivalling SOTA approaches in the respective domains. However, a combined approach of NAS with GANs towards image restoration problems, is still an open research problem (Ganepola and Wirasingha, 2021b). Hence a successful completion of such study could aid in improving the efficiency and accuracy of image restoration tasks as proved by the success of NAS in GAN in image synthesis and segmentations tasks as mentioned above.

1.7 Contribution to the Body of Knowledge

1.7.1 Contribution to the Problem Domain

A majority of State of the Art (SOTA) models used for image restoration tasks such as image deblurring and image super-resolution employ some form of GAN based architecture to increase their accuracy and the quality of the output. The neural networks used for such GAN models are designed, architected, and optimized manually, which is quite a time consuming and difficult process.

The introduction of NAS as a means of architecting such networks would help catalyse the development of more optimized neural networks and GAN models for image restoration tasks and would allow future researchers to develop more approaches to tackle such low-level vision problems with greater accuracy and efficiency.

1.7.2 Contribution to the Research Domain

This project aims to explore a means of harnessing the versatility of NAS as an approach to architecting efficient and optimized neural networks, towards the purpose of designing optimized GAN models for image restoration tasks.

Existing research has explored the usage of GANs (Xu et al., 2017; Gilbert, Messingher and Patel, 2018; Kupyn et al., 2018) and NAS (Chen et al., 2020; Arican et al., 2022) (separately for various image restoration tasks, as well as integrated methods of NAS with GANs for other areas of image processing such as image generation and semantic segmentation. However, no such attempt has been made by combining NAS and GANs thus far towards solving image restoration problems. Therefore, automating the construction of GAN architectures for image restoration using NAS would be the primary technological contribution of this research.

Furthermore, extending NAS in GANs towards image restoration and other image-to-image translation tasks (e.g.: Image colour-transfer, Image style-transfer) could be defined as another contribution of this project.

1.8 Research Challenge

This research aims at contributing towards the improvement of GANs through the integration of NAS, for image restoration tasks such as image deblurring and image super-resolution, which are notorious for being ill-posed and challenging by nature (Xu et al., 2017). As such, the following areas have been identified to pose key challenges for this research.

Image restoration via deep learning networks – The continuous research and contributions made towards image restoration tasks such as image deblurring and image super-resolution has led to the creation of a multitude of different types of image restoration networks with varying levels of complexities and vastly different approaches to training and evaluation. Given the rapidly evolving nature of this domain, successfully understanding and identifying the most relevant and efficient approaches would be a challenge.

Generative Adversarial Networks – GANs offer a novel approach to training generative networks through the use of an adversarial process (Goodfellow et al., 2014), where the generator *G* competes against a discriminator *D* to generate results indistinguishable from the ground truth. Since both networks are trained simultaneously to convergence, the process is resource intensive and makes for a challenging training process with possibilities of premature failures such as mode-collapse (Goodfellow et al., 2014; Goodfellow, 2017).

Neural Architecture Search – NAS focuses on automating the architecting process for neural networks (Elsken, Metzen and Hutter, 2019), with the aim of efficiently producing optimal network architectures. Defining the optimal search space and search algorithm for a selected image restoration network would pose its own challenges due to the aforementioned complex nature of SOTA networks.

For the success of this research, the author would have to deal with the above challenges as part of a cohesive system for image restoration as well as the uncertainties and complications that would arise thereof. Further, the author would also have to tackle diverse and challenging learning areas such as math and optimizations throughout the project's duration.

1.9 Chapter Summary

Starting with an introduction to the problem domain, this chapter provided a detailed overview of the project, along with a clear definition of the problem statement, aims and objectives of the research, planned contributions as well as the expected challenges over the course of its lifecycle. The upcoming chapters will go over the requirement specifications of the proposed system, conceptualized design, and initial implementation details.

2 SOFTWARE REQUIREMENT SPECIFICATION

2.1 Chapter Overview

This chapter provides an in-depth view of the entire requirement gathering process of this research project, spanning from the core research component to the development of the prototype. At the outset, stakeholders of to the proposed system are identified and illustrated through graphical means along with their roles and relationships. Next, potential requirement elicitation techniques are discussed and analysed with detailed reasoning for their selection. The gathered requirements are then presented in tabular form and use case diagrams with supporting descriptions are produced. Ultimately, the findings made through the requirement elicitation process are interpreted and summarized.

2.2 Rich Picture Diagram

The purpose of a rich picture diagram is to demonstrate the relationships that would exist between the proposed system and its wider environment. These entities may extend beyond the system's own domain through tertiary relationships. Additionally, this rich picture diagram also demonstrates potential threats and vulnerabilities that may affect the proposed system due to activities of malicious actors and lawful competitors alike.

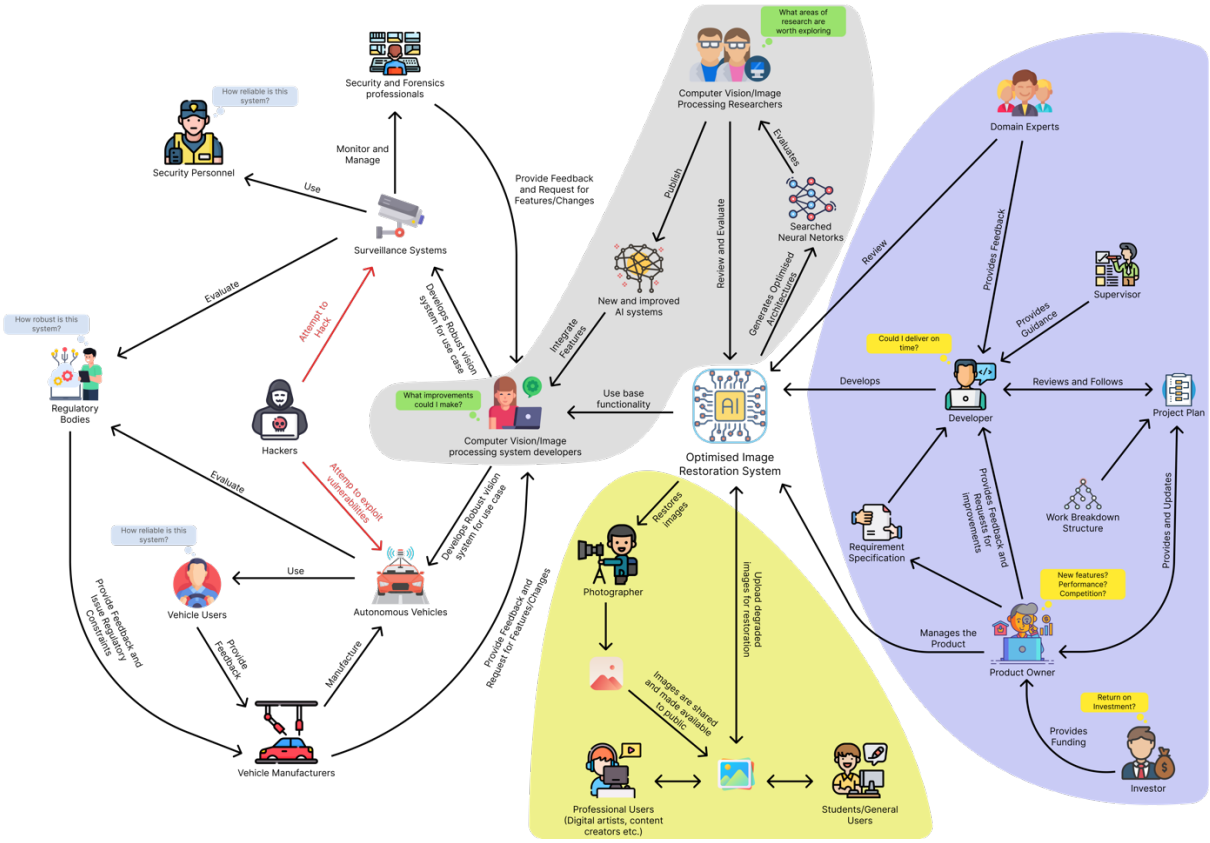


Figure 2-1: Rich Picture Diagram (Self-Composed)

2.3 Stakeholder Analysis

The main stakeholders who participate and interact with the proposed prototype system, identified through the creation of the above rich-picture diagram are presented here within an onion-model diagram. Further, a detailed description of each stakeholder is presented in tabular form for better clarity and understanding.

2.3.1 Stakeholder Onion Model

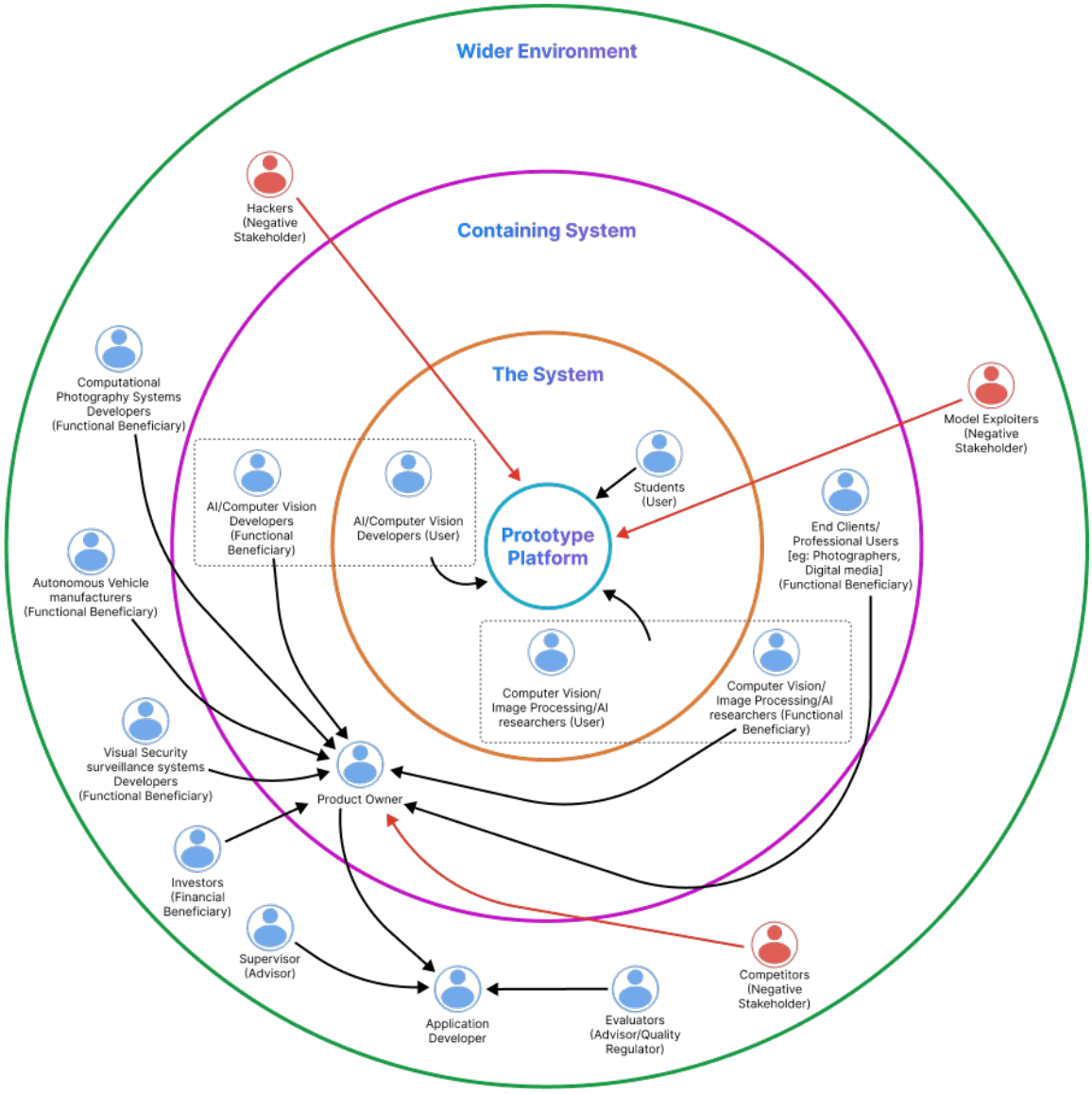


Figure 2-2: Stakeholder Onion Model (Self-Composed)

2.3.2 Stakeholder Viewpoints

Stakeholder	Role	Role Description
Core System		

Students	User/Functional beneficiary	Uses the system and observes how the proposed research project could help improve architecting image restoration neural networks and subsequent GAN models with which they're integrated.
AI/Computer Vision developers	User	Uses the research concepts to develop improved image restoration systems embedded within other complex applications such as autonomous vehicles and visual surveillance systems.
Computer Vision/Image Processing researchers	User	Critically evaluate the research concept and findings and research/provide suggestions on new ways of further enhancing image restoration and related computer vision tasks.
Containing System		
Product Owner	Operational beneficiary Financial beneficiary	Monitor and manage business process and make decision related to the product based on feedback received from other stakeholders.
AI/Computer Vision developers	Functional beneficiary	Provide feedback to the product owner on potential improvements that could be made to the product.
Computer Vision/Image Processing researchers	Functional beneficiary	Provide feedback on improvements that could be made or come up with improved approaches to enhance the core features of the product.

End Clients/Professional Users	Functional beneficiary	Use the product for personal use cases or as a tool for professional requirements.
Investors	Financial beneficiary	Invests in the product and provides funds for further improvements of its features as well as for further research which could help boost performance and efficiency further.
The Wider Environment		
Application Developer	Development and operational staff	Develops the core functionalities of the system and makes improvements/fixes issues wherever possible.
Computational photography systems developers	Functional beneficiary	Uses the research findings to improves capabilities of their own products.
Autonomous Vehicle vision system developers	Functional beneficiary	Uses the research findings to improves capabilities of their own products.
Visual surveillance systems developers	Functional beneficiary	Uses the research findings to improves capabilities of their own products.
Supervisor	Advisor	Guides the developer throughout the conceptualization and development phase and provides feedback as required.
Evaluator	Advisor/Quality Regulator	Provides feedback on potential issues and improvements to the concept and prototype system.
Competitors	Negatives stakeholder	Develops competing image restoration systems using similar or different approaches.

Hackers	Negative stakeholder	Attempts to hack the system.
Model Exploiters	Negative stakeholder	Attempts to exploit the generated image restoration models/network architectures for personal gain or other malicious intent.

Table 2-1: Stakeholder Analysis

2.4 Selection of Requirement Elicitation Methodologies

Requirement Elicitation is a vital component of software development which focuses on gathering the requirements for a software project that allows the developers to obtain a holistic understanding of its purpose and ultimate goals. The tables below summarize the selected requirement elicitation methods along with the relevant justifications, weighing in their pros and cons.

Technique 1 – Literature Review
Exploring, studying, and reviewing existing literature works stands as the initial step in gathering requirements for any research project. This is instrumental in obtaining a clear view of the selected field of research and helps in identifying limitations of the corpus of contemporary research work and available future research directions. Additionally, reviewing literature helps to identify potential approaches and techniques that could help successfully navigate rather complex topics such as NAS and GANs in computer vision.
Technique 2 – Survey
Conducting surveys is one of the simplest, yet most effective forms of gathering requirements for any project or task – obtaining the public's opinion. This allows to gather and quantify data regarding the various requirements, expectations and wishes that end users may have regarding the proposed solution/product. A general questionnaire, being one of the most straightforward means of conducting a survey was selected as it allows for ease of distribution (publicly, since general users are the target demographic) and comprehension for all parties involved.
Technique 3 – Structured Interviews
Since AI/Computer Vision developers and Computer Vision/Image Processing researchers are also part of the target demographic, structured interviews were selected as a requirement gathering instrument. One-to-one structured interview sessions allows for better communication of complex concepts, ideas and points of advice and is arguably the fastest and most efficient means of gathering requirements from such personnel. Furthermore,

obtaining expert feedback to assess and validate the proposed architecture and prototype features is an essential step moving forward. However, features included in the final implementation may vary based on many other factors.
Technique 4 – Prototyping
Exploratory prototyping was selected as the main development approach for the project. This allows for the continuous evaluation, validation, and improvement of the proposed system throughout its development cycle. Since the aim of this research is to develop a novel image restoration model using NAS with GANs, an exploratory prototyping approach would serve well towards identifying potential improvements, optimizations, and adjustments in the design and development of the core research component. Feedback could be obtained from experts throughout the development process once the initial prototype is built.

Table 2-2: Requirement Elicitation Methods

2.5 Discussion of Findings

2.5.1 Literature Review

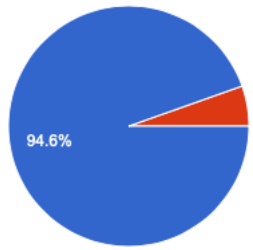
Citation	Findings
(Su, Xu and Yin, 2022)	Most modern methods used for image restoration use deep learning-based approaches to restore images. GANs and other discriminative learning methods have been experimented with
(Chen et al., 2020; Ulyanov, Vedaldi and Lempitsky, 2020)	Deep image priors are capable of efficiently solving standard inverse image restoration problems such as deblurring and super-resolution. Their U-net architecture with Encoder and Decoder blocks play a key role in this process. Moreover, using NAS to architect such U-Net models allows to capture stronger priors from images.
(Kupyn et al., 2018; Pan et al., 2020)	GAN models are equally capable of optimizing image restoration models built for multiple restorations tasks (such as DIP's) as well as addressing specific restoration problems such as de-blurring and de-noising.
(Gong et al., 2019; Ganepola and Wirasingha, 2021b)	The usage of NAS with GAN models designed for computer vision tasks allow for an increase in performance and efficiency. This is reflected both whilst building and testing such deep learning models. The literature has demonstrated that using NAS with GANs for image restoration could, therefore, yield better results.

Table 2-3: Requirement Elicitation - Literature Review

2.5.2 Survey

Question	Do you take photos?
Aim of Question	Identification and filtering of participants

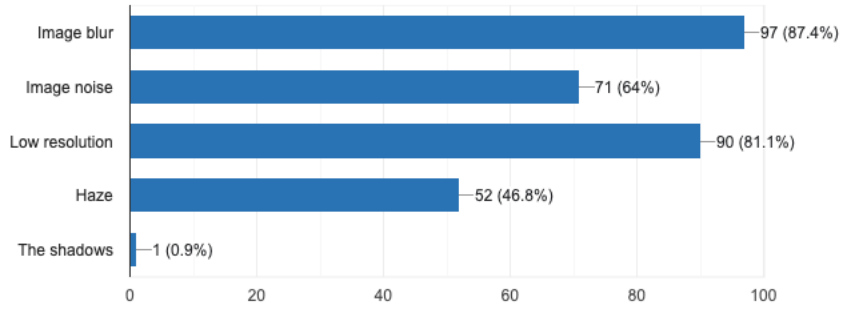
Findings



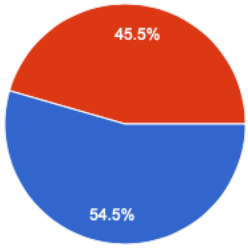
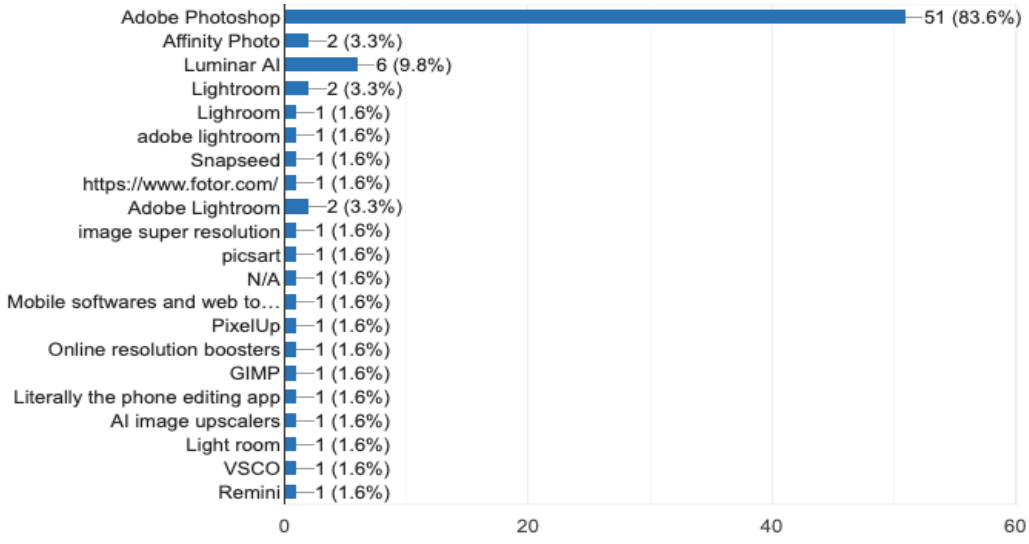
As expected, almost all survey participants (close to 95%) were individuals who took photos at some capacity in their daily lives. This validates the fact that almost all responses received for this survey were through individuals with some form of vested interest in digital images and therefore were familiar with the broader domain to which the proposed system would apply. However, the survey continues for all parties (including those who answered "no") to obtain a more holistic understanding of the general consensus towards the project.

Question	Have you ever noticed any of the following visual degradations in images captured by you or someone else? 1. Image blur 2. Image noise 3. Low resolution 4. Haze
Aim of Question	Validation that the participants were aware of common visual degradations in images and quantification of the most perceived visual degradations.

Findings



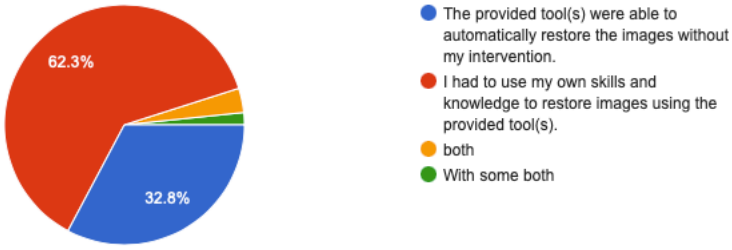
The most common type of image degradation experience by the participants turns out to be image blur (over 87%) whilst low resolution in images trails at a close second (still over 80%). It could be safely assumed that this observation is interlinked with various compressions that images are subject to when being stored and shared between devices.

Image noise and haze also have been selected by a considerable number of respondents as commonly noticed visual degradations. Additionally, one respondent has included shadows in images as a type of visual degradation, which could be the result of the drop in visual clarity in low-light instances.	
Question	Have you ever used any software tools to remove such visual degradations from images?
Aim of Question	Assessing the familiarity of the participants with available image manipulation tools
Findings  The responses to this have been quite evenly matched with a slight majority (almost 55%) showing familiarity with image manipulation tools that could be used to alleviate certain image degradations.	
Question	Which of the following applications have you used to remove such image degradations? 1. Adobe Photoshop 2. Affinity Photo 3. Luminar AI
Aim of Question	Identification of available solutions within the problem domain
Findings 	

An overwhelming majority (83%) of participants who had shown familiarity with image manipulation tools have used Photoshop, one of the most popular applications for digital art and image manipulation to restore images with visual degradations. Image manipulation tools which use AI and other such techniques have also been used by a small minority, which could be indicators of different limitations of said systems.

Question	Were the methods/tools used to restore images using the aforementioned application(s), automatic or manual?
Aim of Question	Assessment of the penetration of AI based approaches towards solving image restoration problems.

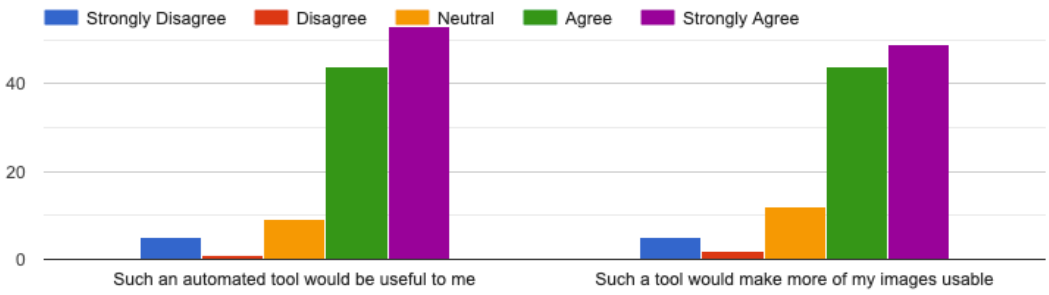
Findings



The majority of users (over 62%) have had to use completely manual methods to rectify degraded whilst a small fraction of participants have used applications where a certain level of human intervention had been required. A sizable portion of the respondents have also had experience with completely automated image restoration systems which is a testament to the capability and potential of applying AI based solutions to this domain.

Question	If there was an automated tool that could remove visual degradations from your images, 1. Would you find it useful? 2. Would it make more of your images usable?
Aim of Question	Validation of the concept of the proposed system

Findings

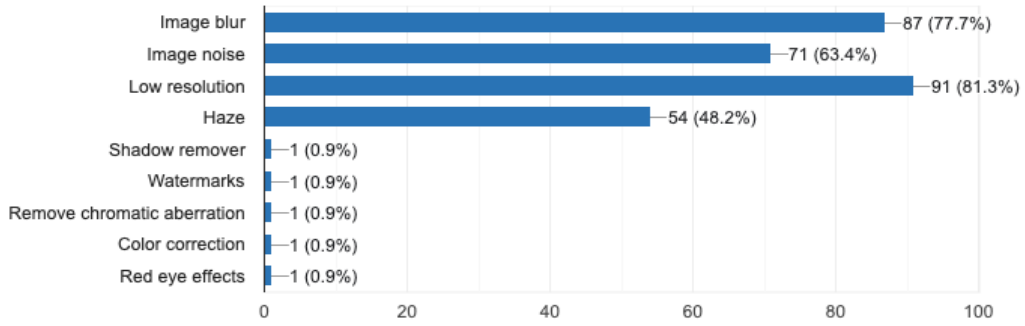


The response to both questions are overwhelmingly positive, with 97/111 participants agreeing or strongly agreeing that an automated image restoration tool of the proposed nature

would be useful to them. Furthermore, 93/111 participants have also responded that they either agree or strongly agree with the fact that such an image restoration tool would make more of their images usable. This validates that the proposed system would be of great use to the vast majority of general users.

Question	Which type of image degradation would you like to see addressed by such an automated image restoration tool?
Aim of Question	Identifying which areas of image restoration should receive more focus, given equal technical feasibility.

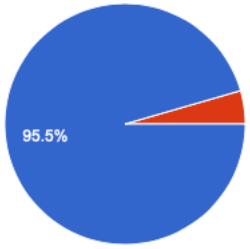
Findings



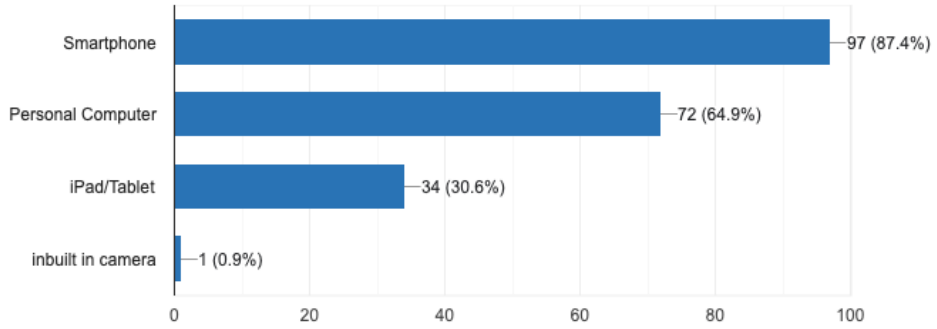
Once again, low resolution (over 81%) and image blur (almost 78%) have been selected as the most requested type of image degradation to be addressed. Image noise and haze have also been requested as areas to which the solution could cater by popular demand. Further, watermarks, chromatic aberration as well as color corrections have been listed as areas of image degradations to which a solution would be welcomed. However, the primary focus would be on addressing the main types of image degradations that have been identified and validated through these responses.

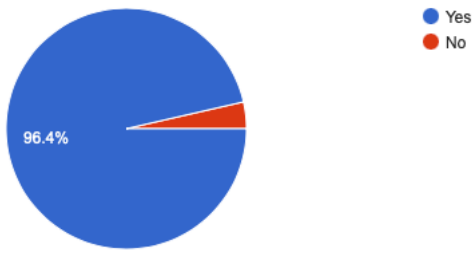
Question	Would you use such an automated image restoration system to restore your own images? (Provided it did not store any of your images or personal data permanently)
Aim of Question	Validation of the usefulness of the prototype to the participants

Findings



Once more, the vast majority of respondents (over 95%) have confirmed that such an automated image restoration system that upholds user privacy would be of personal use to them towards restoring their own degraded images. This validates the core theme

<table>
<tr><td colspan="2">of this research project and confirms the suitability and usefulness of the conceptualized prototype to demonstrate the proposed novel image restoration approach.</td></tr>
<tr><td>Question</td><td>If yes, what device(s) would you like to use such an application on?
1. Smartphone
2. Personal Computer
3. iPad/Tablet</td></tr>
<tr><td>Aim of Question</td><td>Identification of the most suitable candidate as a prototyping platform</td></tr>
<tr><td colspan="2">Findings
 

Smartphones appear to be the predominantly preferred platform to use such an application with over 87% of the participants responding in favour of it. However, personal computers as well as iPads/tablets have also been selected as preferred platforms by a considerable number of respondents, with an aggregate of 95.5% of the user base being in favour. (Multiple selections were allowed). A single participant has shown preference towards such an application being implemented as part of the camera itself. Based on the above diverse distribution of preferences, a web-application appears to be most suitable due to the versatility and platform-agnostic nature afforded through the ability to run the application on common web-browsers.</td></tr>
<tr><td>Question</td><td>Would you like such an application to have a simple user interface without too many distracting options and settings?</td></tr>
<tr><td>Aim of Question</td><td>Identification of end user preferences when making design decisions.</td></tr>
<tr><td colspan="2">Findings
 

With over 95% of the responses being in favor of a simple graphical user interface to interact with the system, a web application with a simple GUI structure is most suitable to demonstrate the core features of the proposed prototype.</td></tr>
</table>

Table 2-4: Requirement Elicitation - Survey

2.5.3 Interview

Codes	Themes	Conclusion
Image Degradations, Visual corruptions	Research Problem	All participants unanimously agreed that the selected research problem is a high-impact area in the domain of computer vision. Further, two of the more experienced participants commended the effort to work on an area of computer vision generally considered to be one of the most complex in the field.
Generative Adversarial Networks, Neural Architecture Search, Image restoration	Research Gap	Using NAS alongside GANs for image restoration was recognized by all interviewees as a highly advanced research gap. They commended the initiative to pursue a project with a heavy research component such as this and highlighted its potential beyond the undergraduate level. However, two of them advised to review existing implementations for both components of the project and to experiment with proven NAS algorithms to avoid unnecessary complications and resource limitations during experimenting and training.
NAS Algorithms, GANs architectures, ML Libraries	Methodology	Due to the inherently resource intensive nature of traditional NAS approaches, it was advised to experiment with memory-efficient NAS algorithms. The definition of the search space and search strategy were highlighted as areas to be experimented with. Since NAS implementations that work with U-net architectures already exist, it was advised to attempt integrating such networks with Pix2Pix or CycleGAN type models which are designed around U-Net architectures. Both PyTorch and Tensorflow were recommended for

		experimenting. SciKit Learn was also indicated as a useful library for this project.
Dataset selection, labelled vs unlabelled data, Evaluation methods	Datasets	All interviewees stressed the importance of using tried and tested datasets for training and testing to ensure the quality of data. Moreover, the usage of such datasets would also assist during evaluation and benchmarking against other available solutions. One participant indicated the importance of finding a dataset with 1:1 matching annotation if experimenting with Pix2Pix GANs. All participants stated that standard evaluation metrics such as PSNR, SSIM and FID (for the GAN model) would be ideal during the testing phase.
User Interface, Application type, data visualizations	Implementation details	Whilst all participants agreed that a web-based UI would be ideal to showcase the proposed system, three of the most senior researchers expressed doubts on the feasibility of developing an end product given the steep learning curve of the research component. However, it was agreed that a simple user interface would be ideal for the task at hand. One participant indicated the integration of Tensorboard with the UI as a potential value addition to the final product.

Table 2-5: Requirement Elicitation – Interviews

- For transcripts of the interview, please refer to APPENDIX A

2.5.4 Prototyping

Prototype Type	Findings
Basic CNN for image denoising	The model is capable of generating a decent output withing a reasonable amount of time. However visual artifacts remain around areas with smooth textures and solid colours.

NAS-DIP with basic U-Net architecture, prior to optimising with NAS.	Quality of outputs for the denoising task were poor, with details being lost mid-way through the training process. Similar results were observed for the super-resolution task. The generated high-resolution image was blurred.
NAS-DIP with basic U-Net based architecture, post NAS optimisation (NAS implementation testing).	The model is capable of generating high-quality outputs. Image denoising takes the least amount of time and resources whilst image inpainting tasks required training for longer.
DeblurGAN (GAN implementation testing)	The model is capable of successfully deblurring images with very high accuracy. However, the GAN training process is difficult and resource/time intensive.
GANSAN Prototype GAN implementation	The U-net architecture of the generator paired with the PatchGAN discriminator is capable of producing decent results. Training time is not nearly as long as some NAS implementations. Convergence is achieved at around 500 epochs, but this number could be reduced by optimising with NAS.

Table 2-6: Requirement Elicitation - Prototyping

2.5.5 Summary of Findings

The below table presents the summary of findings made through all the above-mentioned requirement elicitation methods.

ID	Finding	Literature Review	Survey	Interview	Prototyping
1	Validating the research problem and gap	X	X	X	
2	Novelty of the proposed solution and approach (Using NAS with GANs)	X		X	

3	Significance of using existing NAS algorithms to optimize a neural network (due to the cost of time vs reward when training with limited resources).			X	
4	Significance of using an existing GAN architecture as the basis of the image restoration model.			X	
5	Experimenting with different network and model architectures to find best possible fit			X	X
6	Use tried and tested datasets for model training and testing for fair comparison.			X	X
7	Implement a simple GUI which promotes ease of use for general users.		X		
8	Users should be able to restore degraded images without human intervention.		X	X	
9	If possible, include the ability to train the model on custom datasets through the GUI.			X	
10	If possible, include the ability to download the trained model.			X	
11	It would be beneficial to integrate Tensorboard logs into the GUI if the option to train on a custom dataset is provided.			X	X
12	Show evaluation metrics			X	

Table 2-7: Requirement Elicitation - Summary of Findings

2.6 Context Diagram

The context diagram gives an overview into the system's boundaries, interactions with external entities and the data flows between the system and its primary users. As seen below, there are mainly 3 types of users who would typically interact with this system, namely, General Users/Students, Computer Vision Developers and Computer Vision Researchers.

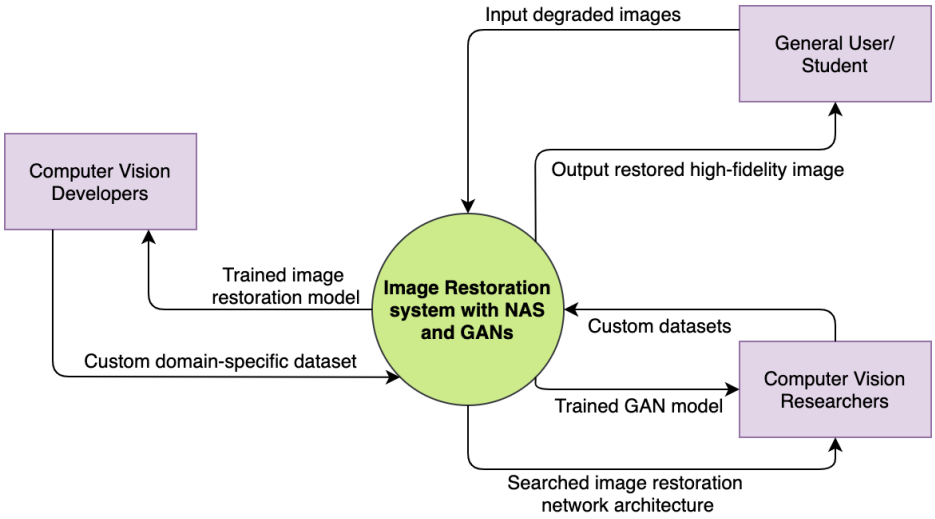


Figure 2-3: Context Diagram (Self-Composed)

2.7 Use Case Diagram

The following diagram illustrates the use-cases bound to the interactions between the system and its most prominent stakeholders. Additionally, relationships between use cases within the system have also been indicated for better understanding.

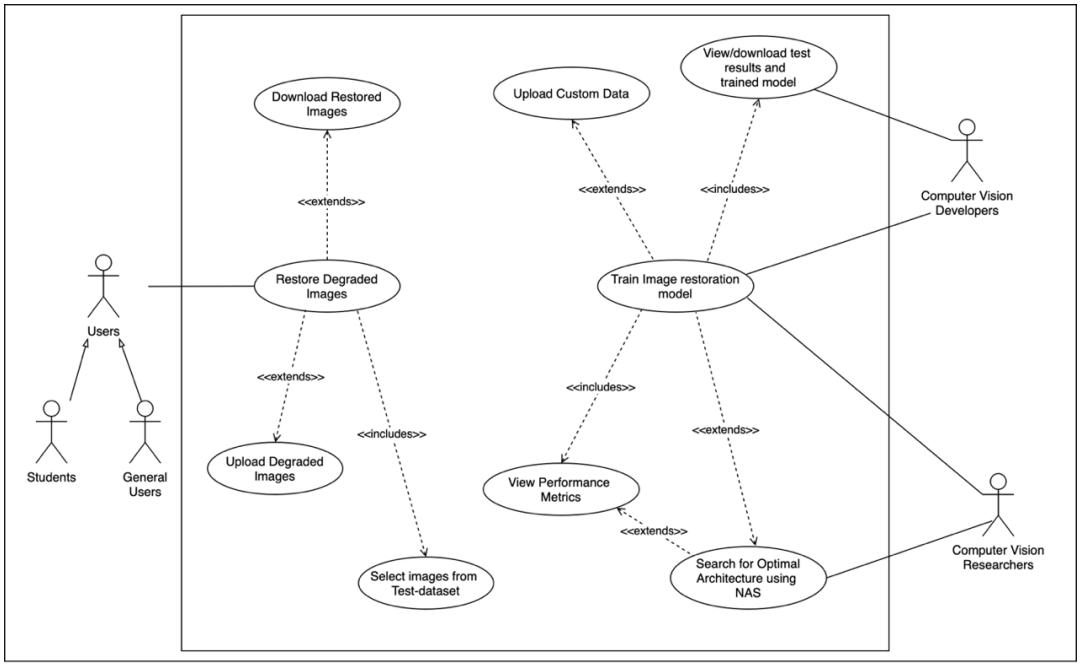


Figure 2-4: Use Case Diagram (Self-Composed)

2.8 Use Case Descriptions

Use Case	**Restore Degraded Images**
Description	This use case describes evaluating the image restoration capabilities of the system.

Participating Actors	General Users	
Pre-Conditions	The image restoration model should be accessible long with the degraded image(s) to be restored.	
Extended Use Cases	1. Upload degraded images. 2. Download restored images.	
Included Use Cases	Selecting degraded images from the test-dataset	
Main Flow	Actor	System
	1. The user activates the system. 2. Select degraded image(s) to be restored. 4. Perform image restoration using the trained model.	3. Load the degraded image to the trained model to obtain a restored result. 5. Display the generated result for viewing and/or downloading. 6. Generate and display performance and evaluation metrics.
Alternate Flows	**AF1 –** If the user requires, they may upload their own degraded images for restoration.	
Exceptional Flows	**EF1 –** Failure to restore the selected image: Display an error message.	
Post-Conditions	1. The restored image output is saved to a directory and remains accessible for download. 2. Testing metrics remain available for evaluation purposes.	

Table 2-8: Use Case Description - 1

Use Case	**Train Image Restoration Model**
Description	This use case describes the process of retraining the image restoration model on a given dataset.
Participating Actors	Computer Vision Developers/Researchers
Pre-Conditions	The required datasets and pre-trained/untrained model should be available on the running system.
Extended Use Cases	1. Upload custom datasets. 2. Search for optimal network architecture using NAS.
Included Use Cases	1. View performance metrics.

	2. View/Download test results and trained model.	
Main Flow	Actor	System
	1. The user must select a dataset to retrain the image restoration model. 3. If the user requires, they may view the performance metrics of the newly trained GAN model. 4. If the user requires, they may download the test results and the trained model.	2. Load the training dataset into the model and start training the model when triggered whilst displaying training and performance metrics.
Alternate Flows	**AF1 –** If the user requires, they may upload a custom dataset to train the restoration model. **AF2 –** If the user requires, they may retrigger a search for the optimal generator architecture using NAS.	
Exceptional Flows	**EF1 –** Unable to train the restoration model: Display an error message.	
Post-Conditions	1. If successful, performance metrics of the model based on test data will be displayed. 2. If the model training is successful, the trained model will be available for download to the user.	

Table 2-9: Use Case Description - 2

2.9 Requirements

Identified functional requirements are categorized based on the "MoSCoW" prioritization based on factors such as projected delivery schedules, value addition to the product and stakeholders, and risk factors involved.

Priority Level	Description
(M) – Must have	Mandatory features which form the core of the project and must be implemented.

(S) – Should have	Important features, which even though not mandatory for the functioning of the prototype, would add significant value if implemented.
(C) – Could have	Supplementary features which would add further value to the system but are not mandatory and essential for the functioning of the prototype. Will be implemented based on availability of time.
(W) – Would not have	These requirements are not important for the implementation of the system and would therefore be omitted altogether.

Table 2-10: Requirement Priority Levels

2.9.1 Functional Requirements

FR ID	Requirement Description	Priority Level	Status
FR1	Users must be able to select a degraded image for restoration.	M	Select restoration model
FR2	Users should be able to upload their own degraded images.	S	Upload degraded image.
FR3	Users must be able to restore images using the trained model.	M	Perform image restoration
FR4	Users should be able to download the restored images.	M	Downloads restore image.
FR5	Computer vision researchers should be able to train the image restoration model on an existing dataset.	S	Retrain the restoration model
FR6	Computer vision researchers could upload their own datasets to train the image restoration model.	C	Upload custom data
FR7	Computer vision researchers could search for an optimized neural network architecture using NAS on an uploaded/available dataset.	C	Search for optimal architecture using NAS
FR8	Computer vision researchers/developers should be able to view performance metrics of the image restoration model.	S	View performance metrics

FR9	Computer vision developers should be able to download the test results and the trained model	S	View/Download test results and trained model
FR10	Computer vision developers and researchers could have the option to tweak parameters related to the model prior to training.	C	Train image restoration model

Table 2-11: Functional Requirements

2.9.2 Non-Functional Requirements

NFR ID	Requirement Description	Specification	Status
NFR1	Users must be able to restore degraded images using the system within an acceptable timeframe from initiation to final output.	Performance	Important
NFR2	Non-technical users must also be able to restore degraded images.	Usability	Important
NFR3	The system should alert the users if an error occurs while executing any task.	Usability	Important
NFR4	The codebase of the project should conform to coding best-practices.	Maintainability	Important
NFR5	Uploading and downloading images/results should be executed efficiently.	Performance	Desirable

Table 2-12: Non-Functional Requirements

2.10 Chapter Summary

This chapter encapsulated the overall process of identifying and gathering requirements for the implementation of the proposed prototype system. Potential stakeholders were initially identified using a rich picture diagram and were filtered and categorized based on the proximity to the system using the Saunder's Onion Model. Next, suitable requirement elicitation instruments were identified, implemented and the gathered findings were presented in a summarized fashion. Subsequently, a context diagram of the system depicting its dataflow as well as a high-level use case diagram pertaining to the interactions between the system and its primary users was presented. Finally, the identified use cases were described in detail and the discovered functional and non-functional requirements of the system were tabulated and presented in-keeping with the "MoSCoW" prioritization method.

3 DESIGN

3.1 Chapter Overview

This chapter will dissect the design decisions made when creating a suitable architecture for the proposed system based on the requirements gathered in the previous chapter. First, the identified design goals will be tabulated after which, the high-level design, low level design, related design diagrams including a system process flow chart and UI wireframes will be presented with comprehensive reasoning for each design choice.

3.2 Design Goals

Design Goal	Description
Performance	**Research Component:** Since the objective of using NAS is to optimize the image restoration network architecture, it is vital that the results produced by the system are highly performant and efficient compared to industry standards. **Prototype Component:** It is important that the proposed system can restore degraded images efficiently within an acceptable timeframe to maximise user satisfaction.
Adaptability	The system should be designed such that it could be adapted to work with images from different domains. Further, it is also important that the prototype is developed such that it could be used on different platforms based on the resources available to the user.
Usability	Since general users are expected to be able to operate the system with minimal guidance, it is important that the UI is elegant and easy to navigate. The provision of an intuitive and lightweight UX is a priority during implementation.
Scalability	In a production environment, the system should be able to work with large datasets and a diverse range of images/visual degradations.
Correctness	Since the proposed system deals with images, the ability to produce results with minimal errors is of paramount importance. Any shortcomings of the system will be evidently visible in generated outputs, causing the purpose of the image restoration system to be lost.

Table 3-1: Design Goals

3.3 High Level Design

3.3.1 Architecture Diagram

Due to the nature of this project, a tiered model, where the presentation, logic, and data tiers could be physically separated was selected as the system architecture. The following diagram depicts the components and modules contained within each tier. The Presentation tier deals with the UI components with which the end users interact, Data tier deals with storing and managing system and user data, whilst the Logic tier bridges the above two and facilitates the functionalities of the system.

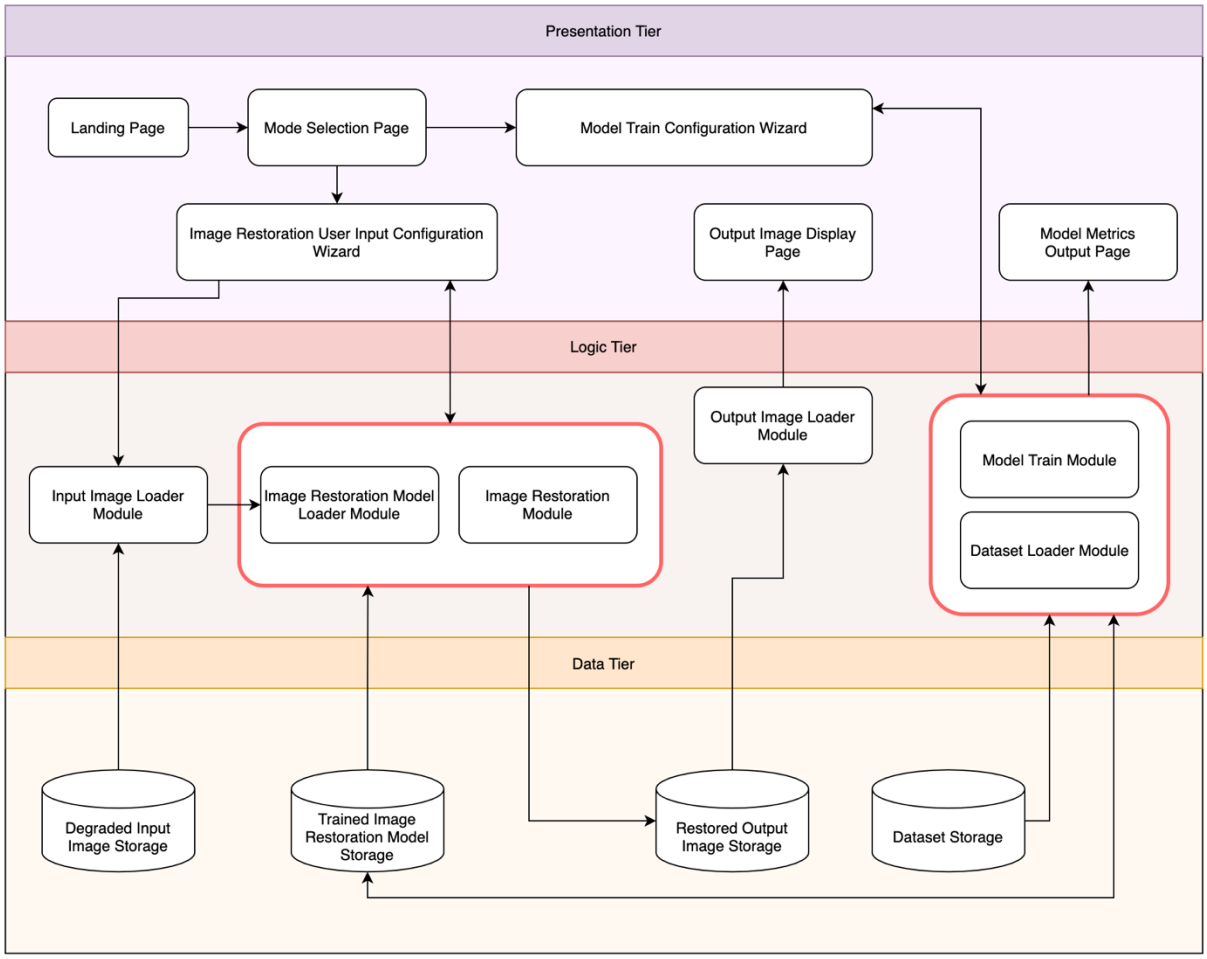


Figure 3-1: High Level Architecture (Self-Composed)

3.3.2 Discussion of Tiers

Data Tier

- **Degraded Input Image Storage –** This module acts as a storage for pre-loaded degraded images which could be selected by the user to test the image restoration capabilities of the system.

- **Trained Image Restoration Model Storage –** This module allows for the storage of the trained optimised image restoration models architected using the approach proposed in this project.
- **Restored Output Image Storage –** This module serves as a temporary storage for images restored using the proposed system, to feed them through to the output image loader module.
- **Dataset Storage –** This module serves as the storage for image datasets used to train the model.

Logic Tier

- **Input Image Loader Module –** The purpose of this module is to load images uploaded by the user or images selected from the degraded input image storage, into the image restoration system.
- **Image Restoration Model Loader Module –** The available image restoration model types are loaded into the system prior to allowing the user to select the desired type of restoration.
- **Image Restoration Module –** This module executes the image restoration process using a saved restoration model, retrieved from the Trained Image Restoration Model Storage.
- **Dataset Loader Module –** This module loads a training dataset into the model, from the dataset storage or user uploads according to the selected configuration.
- **Model Train Module –** This module executes the model training process when triggered by the user via the GUI wizard.
- **Output Image Loader Module –** Images restored and saved to the temporary storage are retrieved and loaded into the system prior to being displayed on the final output page.

Presentation Tier

- **Landing Page –** This will serve as an introduction to the application and will guide the user to try out the different functionalities of the system.
- **Image Restoration User Input Configuration Wizard –** This component will allow the user to configure the system to restore degraded images (self-uploaded or pre-loaded system images).

- **Model Train Configuration Wizard –** The purpose of this component is to allow Computer Vision experts to configure and retrain the restoration models with their own configurations.
- **Output Image Display Page –** This page presents the user with the final output of the system, displaying a restored version of the original degraded input image.

3.4 Low Level Design

3.4.1 Choice of Design Paradigm

A software design paradigm consists of the general philosophy and approach to designing and developing any software system. Since it provides the basic framework and guidance to structure, organize, and maintain the entire project, it is essential to select an appropriate design paradigm for our specific use-case. The two most commonly used design paradigms are **OOAD** (Object Oriented Analysis and Design) and **SSADM** (Structured Systems Analysis and Design Method).

Due to the complex nature of this research project where rapid changes and improvements must be made based on continuous experimentations to achieve a pre-defined goal, **SSADM** was selected as the design paradigm of choice. Additionally, since Python is used as the main programming language, it is not essential to map the system's design to real-world objects.

3.5 Design Diagrams

3.5.1 Component Diagram

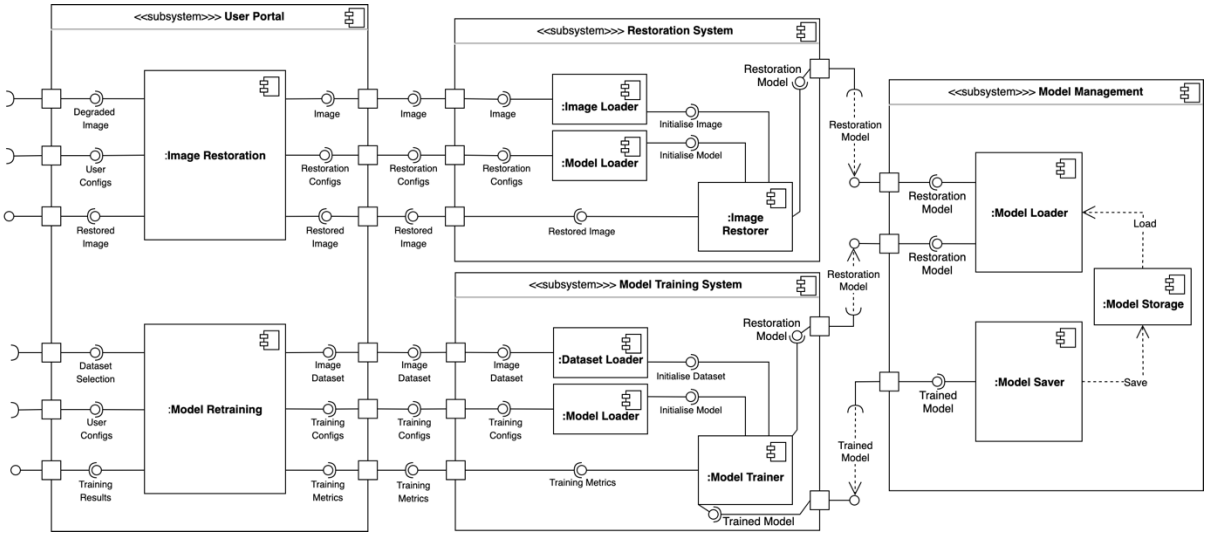


Figure 3-2: Component Diagram (Self-Composed)

For a full-page horizontal view of the Component Diagram, please refer to APPENDIX C

3.5.2 Data Flow Diagram

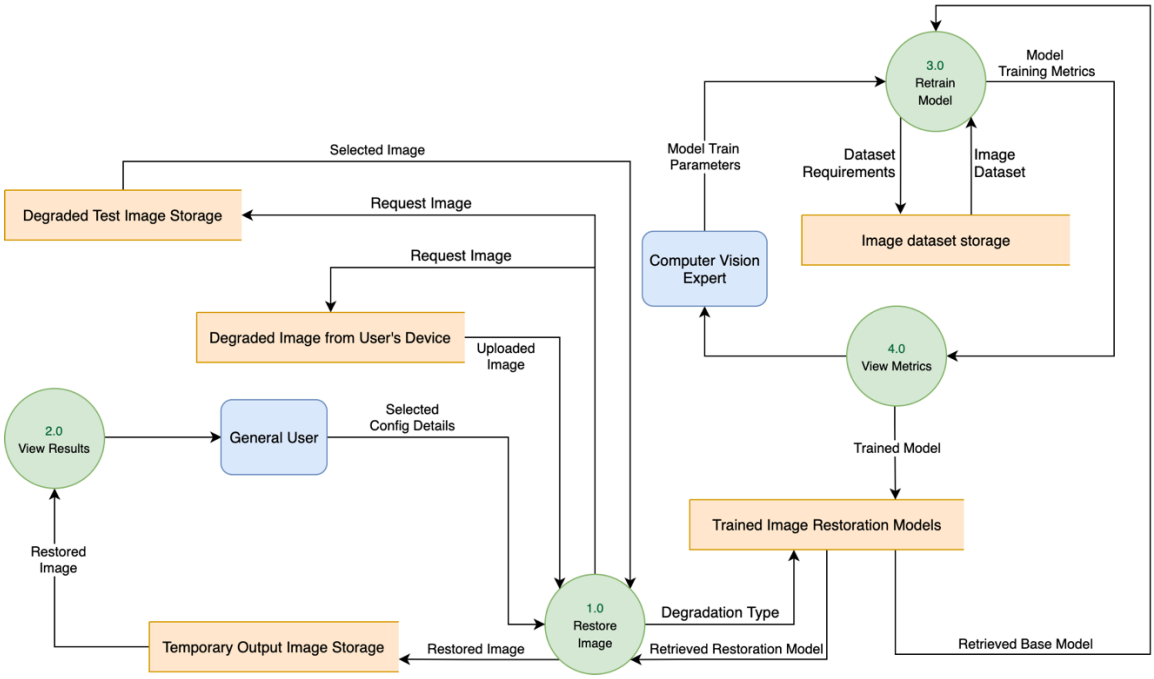


Figure 3-3: Level 1 Data Flow Diagram (Self-Composed)

Depicted through the above data-flow diagram (**Level 1 DFD**) is the relational data flow between various components of the system and external interactors.

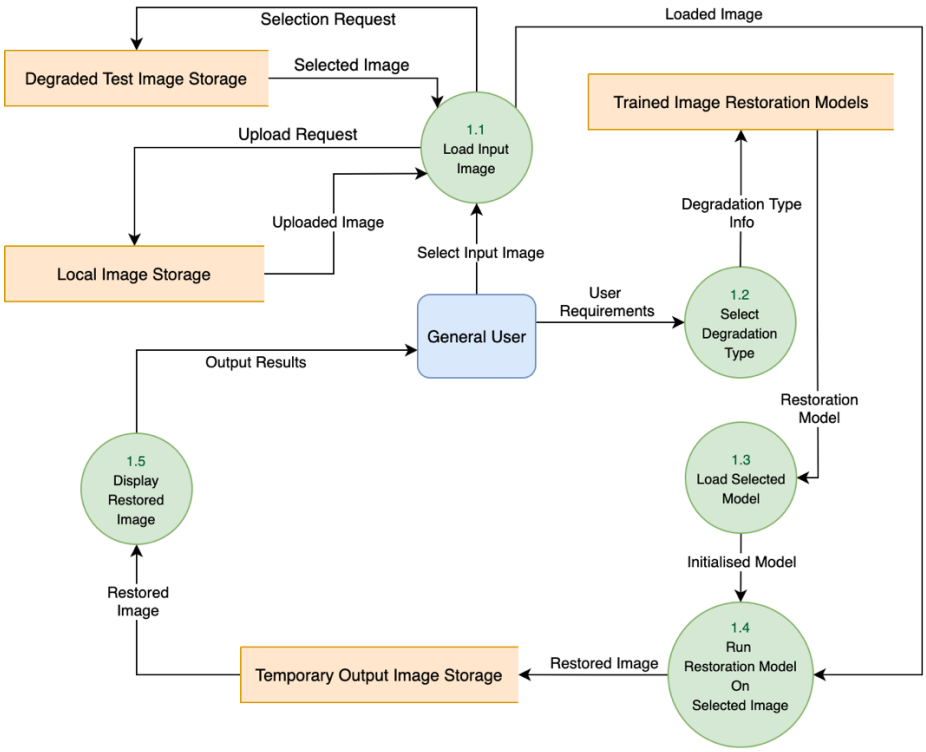


Figure 3-4: Level 2 Data Flow Diagram (Self-Composed)

The above data flow diagram (**Level 2 DFD**) represents an extensive breakdown of the data flow between system components and general users.

3.5.3 System Process Activity Diagram

The below system process activity diagram represents the overall workflow of the prototype system, including inputs, outputs, decision operations, and other processes. It assumes the point of view of a general user of the prototype application.

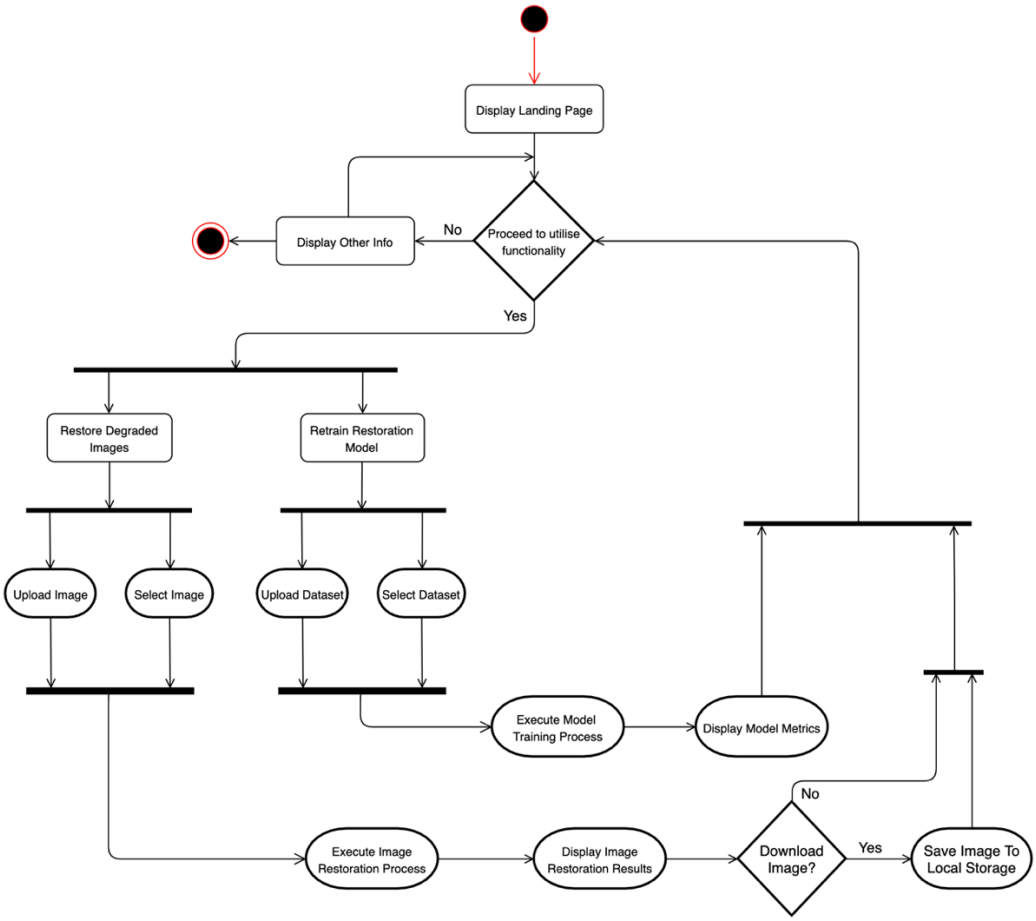


Figure 3-5: Activity Diagram (Self-Composed)

3.5.4 User Interface Design

Below are the UI wireframes pertaining to the degraded image selection page and final output page of the of the proposed web-application to demonstrate the functionality of the research findings.

- For auxiliary UI wireframes please refer to APPENDIX B

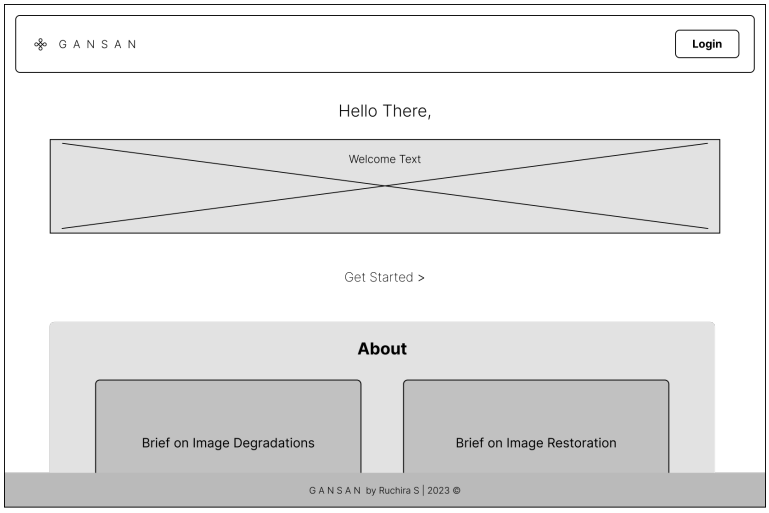


Figure 3-6: Landing page UI wireframe

The landing page is what users will be greeted with, upon launching the application.

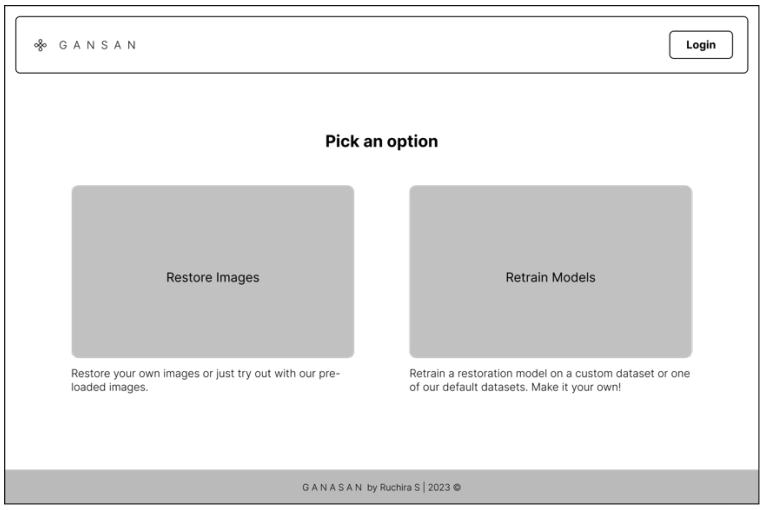


Figure 3-7: Mode Selection page UI wireframe

The mode selection allows users to choose the action they want to perform with the system.

3.6 Chapter Summary

This chapter provided an in-depth run-through of the conceptualized system design and related methodologies utilized prior to the implementation of the proposed solution. Based on the identified design goals, high-level and low-level architectural components have been structured and depicted through relevant design diagrams. Subsequently, the selected software design paradigm has been presented along with sufficient justifications. All areas of the system's design have been elaborately presented through supporting diagrams pertaining to the system's components, data flow, process flow and user interfaces. The following chapter will discuss details related to the system implementation with reference to the deliberated design outline.

4 INITIAL IMPLEMENTATION

4.1 Chapter Overview

This chapter documents in detail, all aspects of the implementation process of the designed prototype. Based on the system architecture created with the knowledge gathered through the literature review and requirement elicitation phases, suitable technologies and development frameworks have been selected. Sufficient justification for all technology selections have been provided along with code snippets supporting implementation details of the core functionality of the prototype.

4.2 Technology Selection

4.2.1 Technology Stack

The following figure represents the technologies selected to for the implementation and functionality of the designed 3-tier architecture.

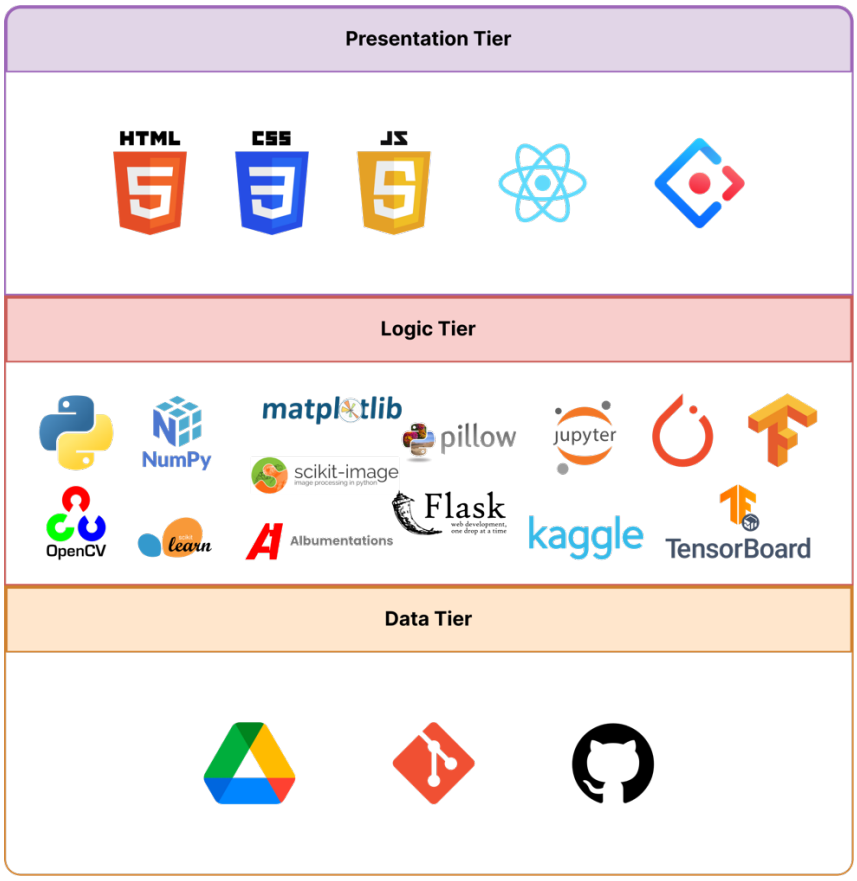


Figure 4-1: Technology Stack (Self-Composed)

4.2.2 Dataset Selection

Based on the knowledge gathered through the literature review and the requirement elicitation process, a mix of generalized and specialised datasets have been selected for this project.

Dataset	Type	Purpose
Div2k	Generalized, high-res images	Used for image restoration GAN and NAS training. Data pre-processing and augmentation methods could be used to generate dataset variants with degradations for training.
GoPro	Blurred and high-res image pairs	Used for image restoration model training. Since image pairs are available, artificial addition of degradations is unnecessary
CIFAR10	Generalized images	Used to train restoration models. Degradations could be added using data augmentation and pre-processing techniques to generate variants as required.
Other Kaggle datasets	Miscellaneous/Images	Since the aim of the project is to generalize the capabilities of the image restoration model, experiments could be conducted with multiple image datasets from reputed sources such as Kaggle.

Table 4-1: Dataset Selection

4.2.3 Programming Languages

The following table depicts the programming languages selected for the implementation of different components of this project along with supporting reasoning.

Programming Language	Reasoning
Python	Python was chosen as the main programming language to build the data-science/deep-learning component of this project due to the extensive availability of supporting libraries, tooling, and community support.
JavaScript	JavaScript was selected as the main front-end scripting language due to its adoption as the industry standard in building responsive, dynamic web applications.

Table 4-2: Programming Language Selection

In addition to the above, HTML and CSS will also be utilized to build and style front-end components of the application.

4.2.4 Development Frameworks

With Python being selected as the main development language, Flask and Django presented themselves as the go to web-development frameworks.

Both contenders were put subjected to a comparative analysis weighing in their individual pros and cons with respect to the nature of this project. **Flask** was selected as the development framework of choice after careful consideration of the rationale summarized in the table below.

Framework	Rationale
Flask	Being a light-weight Python-based micro-framework with no predefined tooling or libraries, Flask is extremely lightweight, flexible, and considerably easier to learn. Additionally, Flask is faster than Django since it has fewer abstraction layers.

Table 4-3: Development Framework Selection

4.2.5 Libraries

Based on the language and development framework selections made for this project, the following libraries were selected to aid in the implementation of the deep learning models and application features.

Library	Rationale
Torch/PyTorch	Torch has been selected since it works alongside Pytorch (which acts as an extensive framework) when building deep learning models for various domains including computer vision.
TensorFlow	Used during early stages of development and prototyping to test image restoration models and approaches.
NumPy	NumPy was selected to address the requirements related to building mathematical and algorithmic functions.
Matplotlib	Commonly used to create data visualisations in python, Matplotlib was selected to display image data and plot testing and evaluation results of the deep learning models.
OpenCV	OpenCV is one of the most popular libraries used for computer vision and machine learning projects. OpenCV was selected to manage the image data used in this project.

Scikit-Learn	These libraries were selected to pre-process and manage image data used in this project. Additionally, Scikit-Image would is used when calculating PSNR and SSIM scores of images when evaluating the performance of the developed model.
Scikit-Image	
Pillow	
Albumentations	
React	Being one of the most used front-end JavaScript libraries, React was selected for this project to help build a responsive and dynamic web application. Its community support and extensibility with other 3rd party libraries would greatly aid in the rapid development of the application's front end.

Table 4-4: Library Selection

4.2.6 IDEs

The following IDE's and code editors were selected for the implementation of this project.

IDE/Code Editor	Justification
Google Colab	Colab was selected for initial implementation and testing since it offers cloud resources and a pre-built environment allowing for rapid setup and execution of ML models.
Jupyter Notebook	Jupyter Notebook was chosen since it allows to run components as code-blocks and its overall versatility. Since a local GPU-machine was used for a bulk of the development process, Notebooks proved to be an ideal overall fit.
PyCharm	The extensive tooling and debugging features of PyCharm would allow to build and test components of the system with greater ease.
VS-Code	VS-Code was selected since it functions as a versatile development platform capable of working with a multitude of different languages and frameworks. Furthermore, the extensive 3rd part library support offered is useful when debugging and testing various components of the project.

Table 4-5: IDE Selection

4.2.7 Summary of Technology Selection

All the above technological selections could be summarized as follows:

Component	Tool(s)/Technology(ies)
Programming Language	Python, JavaScript
Development Framework	Flask

Deep Learning Library	Pytorch
UI Framework/Library	React, AntDesign
Other Libraries	TensorFlow, NumPy, Matplotlib, OpenCV, Scikit-Learn, Scikit-image, Pillow, Albumentations
IDEs	Google Colab, Jupyter Notebook, PyCharm, VS-Code
Version Control	Git, GitHub

Table 4-6: Summary of Technological Selections

4.3 Implementation of Core Functionality

The first stage of implementation was to prepare the datasets to suit our task and model architecture. As such, the following image data manipulation scripts were written to add degradations to high-res images (from the Div2k dataset) and to create a dataset with pairs of horizontally concatenated high res and degraded images.

add_noise_2.py > ...

```python
import numpy as np
import cv2
import glob
import os

mean = 5
var = 2
sigma = var ** 0.9

images = glob.glob('C:\Work\Deep Learning\Lab\Datasets\div2k\\512_val\*.png')
output_dir = ('C:\Work\Deep Learning\Lab\Datasets\div2k\\512_val_noise')

for id, image in enumerate(images):
    img = cv2.imread(image)

    # Define degradation parameters (image noise)
    gaussian = np.random.normal(mean, sigma, (img.shape[0],img.shape[1],img.shape[2]))
    noisy_image = np.zeros(img.shape, np.float32)

    # Add noise to image
    noisy_image = img + img*gaussian

    cv2.normalize(noisy_image, noisy_image, 0, 255, cv2.NORM_MINMAX, dtype=-1)
    noisy_image = noisy_image.astype(np.uint8)

    output_message = 'Noised index' + str(id)
    print(output_message)

    # Save to output directory
    output_filename = os.path.split(image)[1]
    cv2.imwrite(os.path.join(output_dir, output_filename), noisy_image)
```

Figure 4-2: Add noise to image data

combine_data2.py > ...

```python
import cv2
import glob
import os

output_dir = 'C:\Work\Deep Learning\Lab\Datasets\div2k\\512_val_combo'

for i in range(0, 800):

    # Define image data input and output directories
    input_dir_1 = 'C:\Work\Deep Learning\Lab\Datasets\div2k\\512_val_noise'
    input_dir_2 = 'C:\Work\Deep Learning\Lab\Datasets\div2k\\512_val'

    input_filename = str(i) + '.png'

    # Concatenate hr and degraded images horizontally
    img_right = cv2.imread(os.path.join(input_dir_1, input_filename))
    img_left = cv2.imread(os.path.join(input_dir_2, input_filename))
    img_combined = cv2.hconcat([img_right, img_left])

    # Save concatenated image files
    cv2.imwrite(os.path.join(output_dir, input_filename), img_combined)
    output_message = 'Combined index:' + str(i)
    print(output_message)
```

Figure 4-3: Concatenate high-res and degraded images

Next, the generator and discriminator of the image restoration GAN model were implemented. A standard PatchGAN discriminator architecture was selected for our model.

```python
class Discriminator(nn.Module):
    def __init__(self, in_channels=3, features=[64, 128, 256, 512]):
        super().__init__()
        # Initialize as sequential model.
        self.initial = nn.Sequential(
            nn.Conv2d(in_channels*2, features[0], kernel_size=4, stride=2, padding=1, padding_mode="reflect"),
            nn.LeakyReLU(0.2),
        )

        # Define layers in the descrimintor
        layers = []
        in_channels = features[0]
        for feature in features[1:]:
            layers.append(
                CNNBlock(in_channels, feature, stride=1 if feature == features[-1] else 2),
            )
            in_channels = feature

        layers.append(
            nn.Conv2d(
                in_channels, 1, kernel_size=4, stride=1, padding=1, padding_mode="reflect"
            ),
        )

        self.model = nn.Sequential(*layers)
```

Figure 4-4: Model Discriminator

The selected base generator architecture is based on a standard U-Net like design with down-sampling and up-sampling blocks. Shown below is the implemented optimised generator architecture.

```python
class Generator(nn.Module):
    def __init__(self, in_channels=3, features=64):
        super().__init__()
        self.initial_down = nn.Sequential(
            nn.Conv2d(in_channels, features, 4, 2, 1, padding_mode="reflect"),
            nn.LeakyReLU(0.2),
        )
        # Downsampling blocks
        self.down1 = Block(features, features * 2, down=True, act="leaky", use_dropout=False)
        self.down2 = Block(…
        self.down3 = Block(…
        self.down4 = Block(…
        self.down5 = Block(…
        self.down6 = Block(
            features * 8, features * 8, down=True, act="leaky", use_dropout=False
        )
        self.bottleneck = nn.Sequential(
            nn.Conv2d(features * 8, features * 8, 4, 2, 1), nn.ReLU()
        )
        # Upsampling blocks
        self.up1 = Block(features * 8, features * 8, down=False, act="relu", use_dropout=True)
        self.up2 = Block(…
        self.up3 = Block(…
        self.up4 = Block(…
        self.up5 = Block(…
        self.up6 = Block(…
        self.up7 = Block(features * 2 * 2, features, down=False, act="relu", use_dropout=False)
        self.final_up = nn.Sequential(
            nn.ConvTranspose2d(features * 2, in_channels, kernel_size=4, stride=2, padding=1),
            nn.Tanh(),
        )
```

Figure 4-5: Model Generator

During preliminary experiments, it was observed that image artifacts appeared at random in restored image samples. To avoid this, Batch normalisation was replaced with Instance normalisation in both the generator and discriminator.

```python
class CNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=2):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 4, stride, bias=False, padding=1, padding_mode="reflect"),
            nn.InstanceNorm2d(out_channels, affine=True),
            nn.LeakyReLU(0.2)
        )

    def forward(self, x):
        return self.conv(x)
```

Figure 4-6: CNNBlock definition with InstanceNorm in Discriminator

Finally, training is carried out for 500 epochs while saving checkpoints every 5 epochs.

```python
    for epoch in range(config.NUM_EPOCHS):
        train_fn(
            disc, gen, train_loader, opt_disc, opt_gen, L1_LOSS, BCE, g_scaler, d_scaler,
        )

        if config.SAVE_MODEL and epoch % 5 == 0:
            save_checkpoint(gen, opt_gen, filename=config.CHECKPOINT_GEN)
            save_checkpoint(disc, opt_disc, filename=config.CHECKPOINT_DISC)

        save_some_examples(gen, val_loader, epoch, folder="evaluation")
```

Figure 4-7: Training for defined number of epochs

```python
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
TRAIN_DIR = "data/train"
VAL_DIR = "data/val"
LEARNING_RATE = 2e-4
BATCH_SIZE = 16
NUM_WORKERS = 2
IMAGE_SIZE = 256
CHANNELS_IMG = 3
L1_LAMBDA = 100
LAMBDA_GP = 10
NUM_EPOCHS = 500
LOAD_MODEL = False
SAVE_MODEL = True
CHECKPOINT_DISC = "disc.pth.tar"
CHECKPOINT_GEN = "gen.pth.tar"
```

Figure 4-8: Model configs

4.4 Chapter Summary

This chapter provided a detailed discussion on the specifics involved in the implementation of the proposed prototype system based on the formulated design goals discussed in the preceding chapter. The selection of datasets, development frameworks, programming languages and libraries were justified with supporting reasoning. Code snippets related to the implementation of the core components of the prototype were presented for further clarity and understanding of the reader. The subsequent chapter will wrap up this document with details on the progress of the project thus far and final concluding remarks.

5 CONCLUSION

5.1 Chapter Overview

As the final and concluding chapter of this document, this section will focus on the progress made thus far and the deviations taken/planned based on the research findings, implementation complexity and time constraints. Improvements to be made to the prototype system prior to the final submission in the form of a minimum viable product will also be discussed. Finally, a brief demo of the prototype system in its current state will be included for review.

5.2 Deviations

5.2.1 Scope Related Deviations

Based on the knowledge gathered during the course of this research, the following scope related deviations were made from the initial proposal.

1. It was decided to focus on attempting to architect the generator network in an image restoration GAN using NAS instead of the entire restoration model. Limiting the search space to a single network is helpful in reducing the complexity of implementation since both NAS and GANs are rather specialised areas of deep learning which require a great deal of expertise and knowledge to fully comprehend and manipulate.
2. Based on feedback from industry experts, it was decided to experiment with existing NAS implementations and algorithms instead of custom search strategies. This would allow for more experimenting and improvements since implementing NAS from scratch for ill-posed computer vision problems such as these are extremely time and resource consuming processes. Matching NAS and GAN implementations were explored to aid in this process.

5.2.2 Schedule Related Deviations

Due to the complexity of the selected research topic, the actual progress of the project has taken deviations from the initially proposed timeline as follows:

Item	Projected Roadmap	Actual Status
System design and documentation	Start → 21/11/2022 End → 21/01/2023	Start → 0512/2022 End → 25/01/2023
Preparation and amending the PSPD document	Start → 12/12/2022 End → 02/02/2023	Start → 23/12/2022 End → 08/02/2023

Experimentation, implementation, amendment, and documentation of the core research component.	Start → 31/10/2022 End → 01/02/2023	Start → 31/11/2022 End → 08/02/2023

Table 5-1: Schedule Related Deviations

5.3 Initial Test Results

Initial tests were carried out during training and validation of the GAN model. Whilst the generated outputs during the beginning of the training phase were severely lacking in quality, as the model converged, decent results were produced.

The most relevant metrics for image restoration systems are the PSNR score and SSIM score. The developed prototype model can achieve the following results on validation images after training for 500 epochs.

PSNR	`(base) PS C:\Work\Deep Learning\Lab\Results\metrics> python .\psnr.py`
	`PSNR value is 29.17658539899691 dB`
	Figure 5-1: PSNR Score (29.17 dB)

SSIM	`(base) PS C:\Work\Deep Learning\Lab\Results\metrics> python .\ssim1.py -o input_f.png -c result.png`
	`MSE: 237.0743408203125`
	`SSIM: 0.8497957633732129`
	Figure 5-2: SSIM Score (0.849)

Table 5-2: Evaluation Metrics

Figure 5-3: Degraded Input

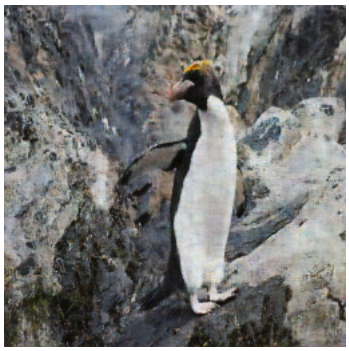


Figure 5-4: Restored Output

Figure 5-5: Degraded validation inputs

Figure 5-6: Restored validation outputs

As seen in the above examples, a significant visual improvement has been achieved using the trained restoration model.

5.4 Required Improvements

Based on the initial test results and the progress made in implementation of the core research component, the following improvements have been identified as essential or highly desirable for the minimum viable product to be presented at the final submission at the end of April.

1. Improve the integration of NAS with the image restoration model so that it could be optimised further.
2. Build a GUI to serve as the client-application of the MVP.
3. Add functionalities to retrain the image restoration model on a custom dataset, if possible, with hardware resources and time constraints.
4. Improve performance of the image restoration process to produce better results.
5. Experiment with other areas of image restoration such as deblurring, super-resolution to allow the user to restore images with different degradation types.

5.5 Demo of the Prototype

Please click the below link to view a brief demonstration of the implemented prototype along with an explanation of the problem being solved and planned future improvements.

- https://youtu.be/JkLOzKsweH4

The link to the prototype's codebase is given below. Since the repository is private, access will be provided if requested.

- https://github.com/Ruchira2k/pix2pix_denoise

5.6 Chapter Summary

This chapter consisted of details of the progress made on the project to-date, including deviations taken and planned, initial test results, and planned improvements for the future. Finally, a brief demo of the prototype was also included for better understanding and clarity of the reader, marking the conclusion to this document.

End of document.

REFERENCES

Arican, M.E. et al. (2022). ISNAS-DIP: Image-Specific Neural Architecture Search for Deep Image Prior. Available from http://arxiv.org/abs/2111.15362 [Accessed 25 September 2022].

Chen, Y.-C. et al. (2020). NAS-DIP: Learning Deep Image Prior with Neural Architecture Search. In: Vedaldi, A. Bischof, H. Brox, T. et al. (eds.). *Computer Vision – ECCV 2020*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 442–459. Available from https://doi.org/10.1007/978-3-030-58523-5_26 [Accessed 28 September 2022].

Elsken, T., Metzen, J.H. and Hutter, F. (2019). Neural Architecture Search: A Survey. Available from http://arxiv.org/abs/1808.05377 [Accessed 28 September 2022].

Fu, K. et al. (2020). Image Super-Resolution Based on Generative Adversarial Networks: A Brief Review. *Computers, Materials & Continua*, 64 (3), 1977–1997. Available from https://doi.org/10.32604/cmc.2020.09882.

Ganepola, V.V.V. and Wirasingha, T. (2021a). A Novel Framework for Semantic Image Segmentation using Generative Adversarial Networks and Neural Architecture Search. *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*. 11 August 2021. Negambo, Sri Lanka: IEEE, 252–257. Available from https://doi.org/10.1109/ICIAfS52090.2021.9605889 [Accessed 12 October 2022].

Ganepola, V.V.V. and Wirasingha, T. (2021b). Automating Generative Adversarial Networks using Neural Architecture Search: A Review. *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. 5 March 2021. Pune, India: IEEE, 577–582. Available from https://doi.org/10.1109/ESCI50559.2021.9396991 [Accessed 28 September 2022].

Gao, C. et al. (2020). AdversarialNAS: Adversarial Neural Architecture Search for GANs. Available from http://arxiv.org/abs/1912.02037 [Accessed 28 September 2022].

Gilbert, A., Messingher, S. and Patel, A. (2018). Non-Blind Image Deblurring using Neural Networks. 7.

Gong, X. et al. (2019). AutoGAN: Neural Architecture Search for Generative Adversarial Networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. October 2019. Seoul, Korea (South): IEEE, 3223–3233. Available from https://doi.org/10.1109/ICCV.2019.00332 [Accessed 28 September 2022].

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. Available from http://arxiv.org/abs/1701.00160 [Accessed 18 October 2022].

Goodfellow, I.J. et al. (2014). Generative Adversarial Networks. Available from http://arxiv.org/abs/1406.2661 [Accessed 28 September 2022].

Hu, B. et al. (2020). Subjective and objective quality assessment for image restoration: A critical survey. *Signal Processing: Image Communication*, 85, 115839. Available from https://doi.org/10.1016/j.image.2020.115839.

Kupyn, O. et al. (2018). DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. Available from http://arxiv.org/abs/1711.07064 [Accessed 19 July 2022].

Pan, X. et al. (2020). Exploiting Deep Generative Prior for Versatile Image Restoration and Manipulation. Available from http://arxiv.org/abs/2003.13659 [Accessed 28 September 2022].

Su, J., Xu, B. and Yin, H. (2022). A survey of deep learning approaches to image restoration. *Neurocomputing*, 487, 46–65. Available from https://doi.org/10.1016/j.neucom.2022.02.046.

Suganuma, M., Ozay, M. and Okatani, T. (2018). Exploiting the Potential of Standard Convolutional Autoencoders for Image Restoration by Evolutionary Search. 10.

Ulyanov, D., Vedaldi, A. and Lempitsky, V. (2020). Deep Image Prior. *International Journal of Computer Vision*, 128 (7), 1867–1888. Available from https://doi.org/10.1007/s11263-020-01303-4.

Wang, Z., She, Q. and Ward, T.E. (2022). Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy. *ACM Computing Surveys*, 54 (2), 1–38. Available from https://doi.org/10.1145/3439723.

Xu, X. et al. (2017). Learning to Super-Resolve Blurry Face and Text Images. 10.

Zhang, H. et al. (2021). Memory-Efficient Hierarchical Neural Architecture Search for Image Restoration. Available from http://arxiv.org/abs/2012.13212 [Accessed 25 September 2022].

Zhang, K. et al. (2022). Deep Image Deblurring: A Survey. *International Journal of Computer Vision*, 130 (9), 2103–2130. Available from https://doi.org/10.1007/s11263-022-01633-5.

APPENDIX A – Thematic Analysis (Requirement Elicitation)

Interviewee ID	Name	Designation
P1	Anonymous	Senior Lecturer, Department of Electronic and Telecommunication Engineering, University of Moratuwa
P2	Ms. Vishmi Ganepola	Software Engineer/Researcher, Pearson Lanka
P3	Ms. Senuri De Silva	PhD Student, National University of Singapore
P4	Mr. Thepul Ginige	Senior Lecturer/Programme Coordinator, University College Lanka
P5	Mr. Dulaj Weerakoon	Research Engineer, Singapore Management University
P6	Anonymous	Data Scientist, King's College London

Table 0-1: Appendix A - Interviewee Details

Question	**How useful do you think a system to automate the creation of image restoration neural network architectures and integrating them with GANs would be on a scale of 1-5?**
Responses	
P1	"Concept sounds pretty advanced, especially for an undergraduate. Surely a 4 or 5 out of 5 for me. Working with degraded images is one of the more difficult areas in computer vision"
P2	"I definitely think it would be a 4 or 5 out of 5. It's important to try out new approaches to solve problems because that is the point of having technological developments."
P3	"This idea sounds interesting. I'm not too familiar with image restoration, but the concept sounds novel and unique. I think it's a 4 out of 5"

P4	"Automating this is a bit like AutoML I think. Should be very useful if you can build it. It's not easy to work with visual corruptions – but it's very useful for a lot of fields nowadays. I'd say 4 out of 5"
P5	"GANs is not very easy to work with. Automating the architecture will help a lot of research in this field. 4 out of 5"
P6	"I'm not very familiar with computer vision, but I know GANs is very powerful and complex. Automating part of the architecture of something like that is definitely a 4 out of 5. This will be useful since you're working on image degradations and corruptions"
Question	**What are your thoughts on using Neural Architecture Search to automatically architect such image restoration neural networks?**
Response	
P1	"NAS is very resource intensive. If you can find a way to use it with low resources, then that is great!"
P2	"Using NAS for this is very impressive. Experimenting with applying NAS towards image restoration problems is a commendable research effort."
P3	"Very useful trying out state of the art methods"
P4	"This would be good for Memory-Efficient Hierarchical Neural Architecture Search for Image Restoration"
P5	"Using NAS with GANs is very complex. This has high academic research value"
P6	"Impressive to use technologies like that."
Question	**What are the main technical aspects (if any) you would advise that I must prioritise and consider in relation to the NAS component of this project?**
Response	
P1	"Try to use existing NAS algorithm and integrate it with the restoration model. If you try to build it from scratch it might be too resource intensive."
P2	"Consider existing NAS based systems as a starting point since it would be easier to obtain a better understanding on how such a system functions. Using an existing algorithm would also make it easier to do more experiments and make improvements faster within the available time period."
P3	"I'm not familiar enough with NAS to give a definitive answer"

P4	"Evolutionary neural architecture search, Deep Image Prior to be applied fully automatically"
P5	"Try out with existing algorithms to get more understanding. Mathematically based NAS algorithms might be a lot more efficient than ones using reinforcement learning or similar approaches."
P6	"I'm not too familiar with NAS to answer this question properly"
Question	**Are there any specific network architectures/GAN models or NAS algorithms that you could recommend for the purpose of this project?**
Response	
P1	"Look into existing NAS algorithms. You could also start with GAN architectures like Pix2Pix or CycleGAN. You must first verify if the selected NAS and GANs components are compatible."
P2	"Pix2Pix GAN might be very good for this. CycleGAN is good too but could be more complex since there are 4 networks instead of 2"
P3	"Try out some popular and proven GAN architectures first"
P4	"CycleGAN can be used"
P5	"I'm not sure about specific GAN architectures but you can start with some commonly used ones. Pix2Pix might be a good fit"
P6	"Try to select a GAN with lesser complexity because otherwise the research might become too heavy"
Question	**What AI/ML libraries would you recommend experimenting with for the progression of this project?**
Response	
P1	"Standard ones like PyTorch and Tensorflow should do."
P2	"You can try with PyTorch and also Tensorflow. Tensorboard is useful to view model metrics. Other image processing libraries will also be useful."
P3	"Both Tensorflow and Pytorch can be tested."
P4	"Tensorflow, Pytorch, SciKit Learn. There are other libraries as well."
P5	"Tensorflow or PyTorch. You might have to experiment a little to decide properly."
P6	"Pytorch and Tensorflow mainly."

Question	**What web-application tools/framework(s) would you recommend for building a prototype to demonstrate the functionality of the proposed system as an end product?**
Response	
P1	"That is up to you. The most important part here is the core research component because it has a very good novelty and potential. Try to focus on that"
P2	"You can use React for the front end and something like Flask to build the API. Use whatever you're comfortable with"
P3	"Flask or Django depending on your preference. Use any front-end library you like"
P4	"React is good for front end if you're familiar with it. Django or Flask for backend"
P5	"You can use React or Vue if you're comfortable with that, or even Flutter. Backend could be built with Flask. This project has very high academic research value so it's best to focus on that"
P6	"Use any libraries you're comfortable with. Focus on the research component more. This project seems to have very high potential"
Question	**What datasets would you recommend for training the NAS and GAN components of this project?**
Response	
P1	"Use datasets that have been commonly used for other research in this domain. Make sure the data is high quality because it will affect your results"
P2	"NAS for DIP dataset since it has been used for another NAS project. If you're using a Pix2pix GAN, make sure you have 1:1 annotated dataset for training."
P3	"RealBlur, GoPro-V7 datasets"
P4	"RealBlur and GoPro-V7 datasets. Try out other popular datasets as well."
P5	"You could try with tried and tested datasets to make sure the data is of high quality. For initial testing you could use something like the MNIST dataset. Later you could use datasets with complex high-resolution images."
P6	"You can try with Kaggle datasets or other datasets used for similar projects."
Question	**What evaluation metrics would you recommend for evaluating the final image restoration model and resulting outputs?**

Response	
P1	"Standard metrics like PSNR and SSIM"
P2	"PSNR, SSIM scores are useful."
P3	"FID scores could be used since you're working with a GAN model."
P4	"PSNR Score to verify the quality of the output."
P5	"PSNR and SSIM scores. FID score could be used to verify GAN performance."
P6	"PSNR score can be used"
Question	**What features would you recommend to include in the final prototype of this project?**
Response	
P1	"You can show the results by restoring an image. Spend more time on improving the core component of the research. Use a simple UI if you need one"
P2	"Start with the main component, that is restoring degraded images. If you have time you could also add features to allow retraining the model on custom datasets. Also you could integrate Tensorboard into the GUI to show the model performance."
P3	"Nothing in specific. Showcase the performance of the core research component"
P4	"Mainly the image restoration feature. Use a simple UI"
P5	"Include the core feature, that is restoring degraded images. You could also show the model performance metrics on the GUI. Build a simple UI so that you have more time to spend on the core research component."
P6	"Focus on the image restoration component mostly"

Table 0-2: Appendix A - Interview Summary

APPENDIX B – UI Wireframes

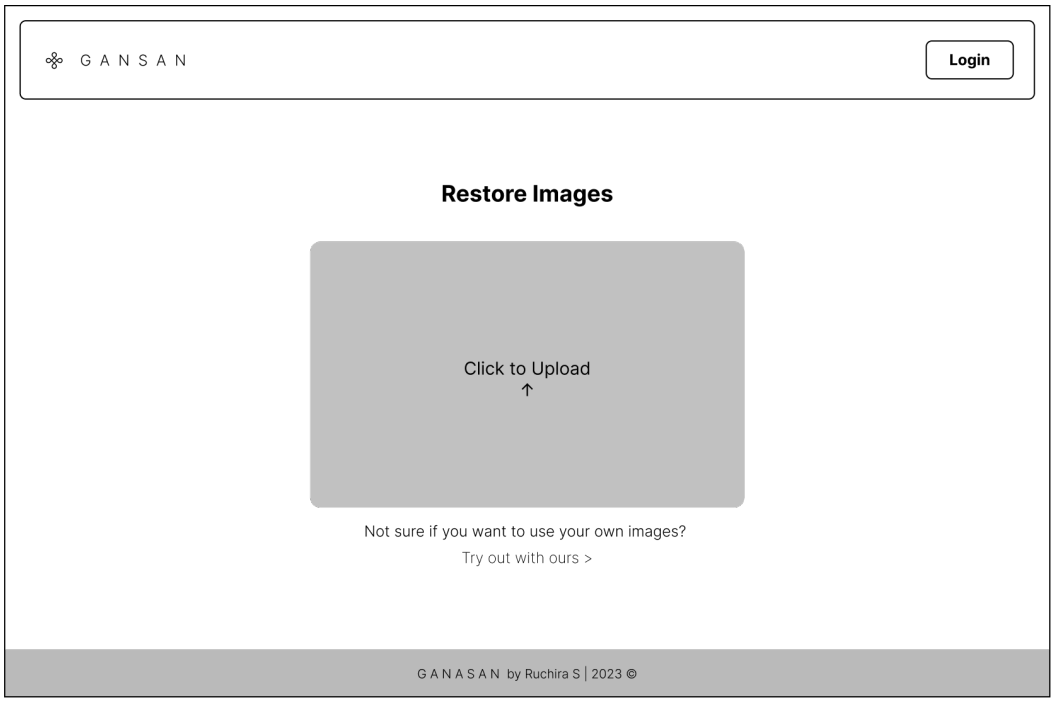


Figure 0-1: Image Upload page

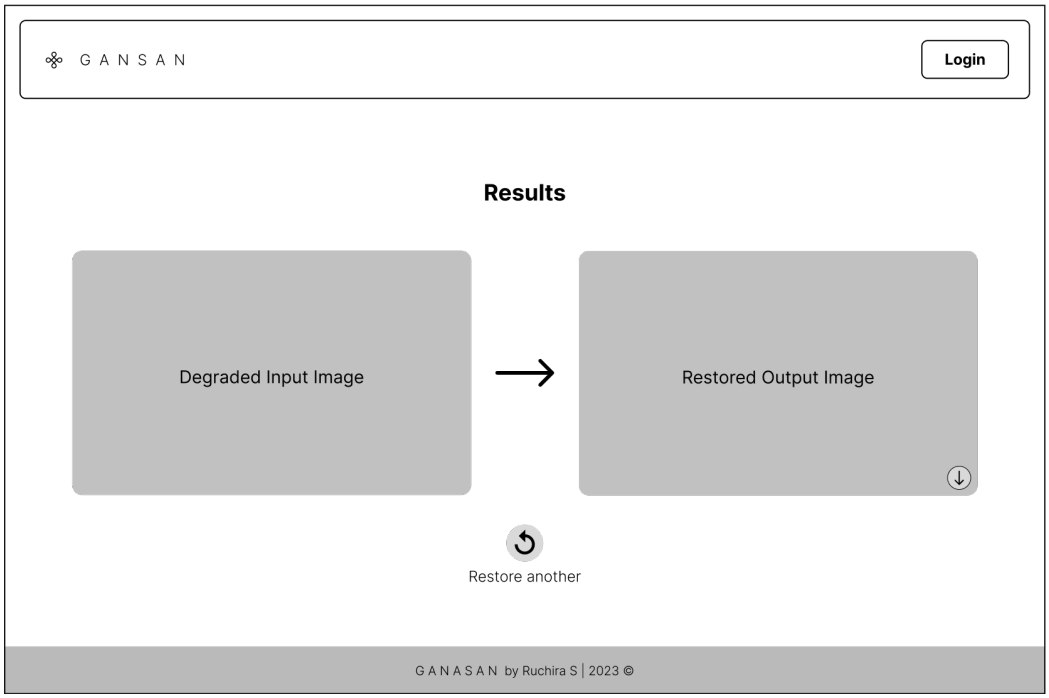


Figure 0-2: Results Output page

APPENDIX C – Component Diagram

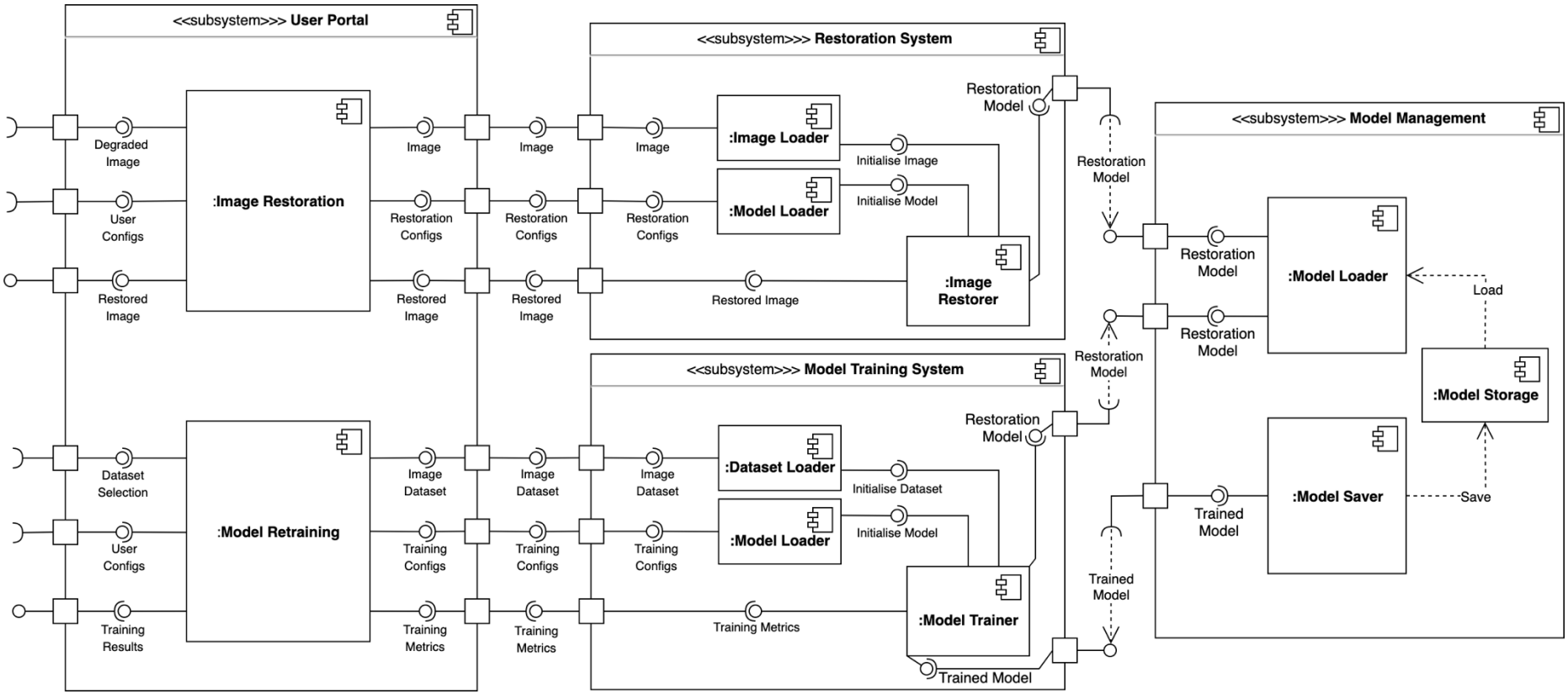


Figure 0-3: Component Diagram (Self-Composed | Horizontal)