Informatics Institute of Technology
In Collaboration With

University of Westminster, UK

University of Westminster, Coat of Arms

DAugtelligent: Automating Data Augmentation Pipeline for Image Classification

A Project Specification Design & Prototype by
Mr. Idirimuni Pubudu Mihiranga De Silva
W1761268 / 2019285

Supervised by
Mr. Guhanathan Poravi

February 2023

This Project Specification Prototype & Design is submitted in partial fulfillment of the requirements for the BEng (Hons) Software Engineering degree at the University of Westminster.

Abstract

In recent years, the computer vision (CV) domain has focused on developing deep learning (DL) based algorithms to perform tasks like image classification. The performance of these DL algorithms heavily depends on the big data. Because the algorithms can learn from more variations of the input data, which can help prevent overfitting. However, collecting proper training datasets in many real-world domains is well-known to be labor-intensive and expensive. Data augmentation (DA) is a widely accepted solution to improve the low diversity datasets. But selecting optimal data augmentation policies (DA policies) based on the given dataset using traditional DA techniques is time-consuming and requires domain expertise.

To mitigate such difficulties and limitations of traditional DA, a novel approach is to use automated data augmentation (AutoDA) techniques to automatically design the best-performing image DA policies based on the dataset and task type. Existing AutoDA solutions need to trade off simplicity, search time cost, and performance to use them in real-world use cases. In this research, the author presents a novel practical AutoDA approach named '**DAugtelligent'**, utilizing Differentiable Programming (DP) into the AutoDA domain.

As per the initial test results found on the CIFAR10 dataset and Wide ResNet 28x10 model, the DAugtelligent was able to archive **86%** accuracy on Wide ResNet 28x10 model. Moreover, due to the new reparameterization of the search space, the system was able to find the best possible DA policy magnitude within **11 CPU hours**. Additionally, unlike existing AutoDA solutions, it is easy to configure user-preferred datasets and image classification models to the DAugtelligent due to its simplicity. After considering all these outcomes, DAugtelligent can be claimed as the most practical AutoDA solution at the moment.

Subject Descriptions:

- Machine Learning → Low Diversity Datasets → Low Accuracy & Generalizability
- Improve Low Accuracy & Generalizability → Data Augmentation → Difficulties in Traditional Data Augmentation → Automation in Data Augmentation

Keywords: Data Augmentation, Automated Data Augmentation, Differentiable Programming, Convolutional Neural Network

Table of Content

List of Figures

List of Tables

Acronyms

AutoDA	Automated Data Augmentation.
DA Policy	Data Augmentation Policy.
ML	Machine Learning.
AutoML	Automated Machine Learning.
DL	Deep Learning.
DP	Differentiable Programming.
MVP	Minimum Viable Product.
UI	User Interface.
SE	Software Engineering.
CV	Computer Vision.
DFD	Data Flow Diagram.
LR	Literature Review.

CHAPTER 1: INTRODUCTION

1.1 Chapter Overview

In this research work, the author aims to address the current limitations in the AutoDA domain and introduce a new practical approach to automate the DA pipeline for the image classification task, which will increase the usage of the AutoDA concept in real-world DA tasks.

This chapter discusses the DA and AutoDA problem domains, the research aim and objectives, the necessary evidence to prove the research gap, and the novelty of the research. Lastly, the author addresses the challenges of this research work that the author seeks to overcome.

1.2 Problem Domain

In recent years, the computer vision (CV) domain has focused on developing deep learning (DL) based algorithms that can learn information from digital images to perform artificial intelligence (AI) tasks like image classification. The performance of these DL algorithms heavily relies on the quality and volume of data samples in the training dataset (Shorten and Khoshgoftaar, 2019). This is because DL algorithms can learn more information from the high quality and large amount of training datasets (Khalifa et al., 2022; Shorten and Khoshgoftaar, 2019). However, collecting proper training datasets in many real-world domains is well-known to be labor-intensive and expensive. For example, medical image analysis. Hence, improving the performance of DL algorithms is one of the key challenges.

Overfitting is a common issue in DL algorithms that occur due to the poor training dataset. Overfitted DL algorithms perform very well in already-seen data (training data) but not in unseen data (testing data) (Shorten and Khoshgoftaar, 2019). The following graphs show what overfitting looks like when visualizing training and testing performance over time.

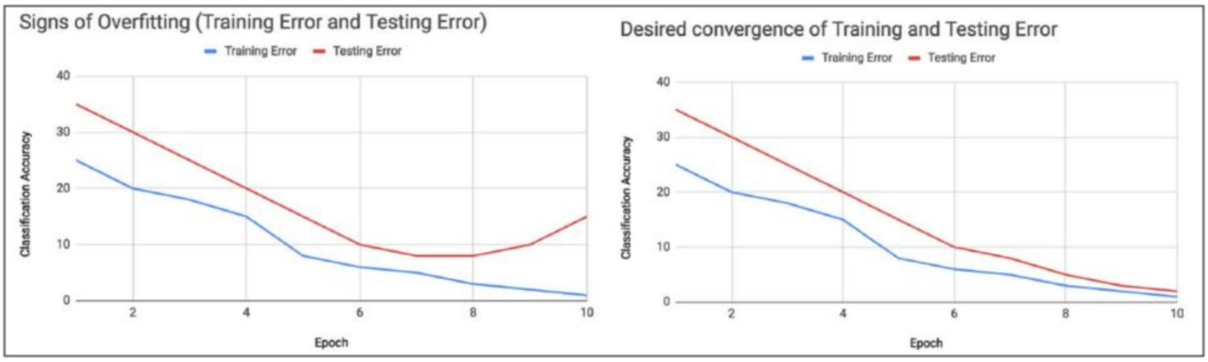


Figure 1: Visualization of model accuracies over training epochs (Shorten and Khoshgoftaar, 2019)

In a better DL algorithm, the training and testing errors must gradually decrease with time (Shorten and Khoshgoftaar, 2019).

1.2.1 Data Augmentation

Data augmentation (DA) is a widely accepted solution to improve low-diversity datasets. The aim of DA is to artificially increase the size of the original dataset by extracting more information from the existing training dataset using oversampling or data-warping DA techniques (Shorten and Khoshgoftaar, 2019). Oversampling DA technique adds synthetic data points to the training dataset, and the data-warping DA technique reshapes existing data points into new shapes while ensuring their original data label is maintained (Nanni et al., 2021; Shorten and Khoshgoftaar, 2019). The augmented data points will provide a wider diversity of possible information, decreasing the testing and training error gaps. Ultimately, DA helps to overcome the overfitting problem and performance improvements of the DL algorithms.

DA can be used to extract more information from many types of data. For example, images, text, and audio. However, in this research work, the author only considers image DA.

According to Shorten and Khoshgoftaar, image DA can divide into two categories. Which are:

1. Basic/classical image manipulations DA techniques
    - Geometrics transformations
    - Kernel filters
    - Random erasing
    - Color space transformations
    - Mixing images
2. Deep learning DA techniques
    - Neural style transfer
    - Adversarial training
    - Generative Adversarial Network-based DA, also known as GAN

1.2.2 Automated Data Augmentation

In the image DA, the DA policy is a collection of image transform operations that are used to extract artificial data points from original data points (Yang et al., 2022). So far, DA policies for CV tasks have been designed manually, and the best-performing DA policies are dataset-specific. For instance, on the MNIST dataset, most top-ranked image classification models use rotation, elastic distortions, translation, and scale. On natural datasets, such as ImageNet and CIFAR 10/100, image mirroring, color shifting, and random cropping are conventional (Yang et al., 2022). Therefore, the manual designing of DA policies based on the given dataset and task is highly subjective and error-prone. The traditional trial-and-error method based on the performance of a DL algorithm might result in unwanted data collection, wasting computational resources and efforts (Yang et al., 2022). To mitigate the difficulties in traditional DA, a novel approach is to use automated data augmentation (AutoDA) techniques to automatically learn and design the best-performing DA policies based on the dataset and task type (Cubuk et al., 2019a). The AutoDA models aim to automatically design the best-performing DA policies that boost given DL algorithm performance. Moreover, recent DA research works have shown that DA policies that are generated by AutoDA models perform better than manually designed DA policies (Cubuk et al., 2019a).

1.3 Problem Definition

DA is a widely accepted fruitful method to avoid overfitting issues and increase the performance of DL algorithms. However, currently, DA policies have been designed manually, and the best-performing DA policies are dataset-specific. As a result, to design best-performing DA policies manually for a given dataset and task requires a considerable amount of expertise in the DA domain, powerful computational resources, and a lot of time. So far, a significant focus of the DL community has been on refining the architectures of the DL algorithms. Less attention has been put into solving difficulties in traditional DA and automatically identifying best-performing DA policies based on the dataset and task type (Cubuk et al., 2019a). Motivated by advancements in automated machine learning (AutoML), the requirement for automatically learned DA has lately been raised as an important problem to solve.

1.3.1 Problem Statement

DA is a widely accepted fruitful technique to overcome poor dataset-related problems like overfitting, but it isn’t easy to design best-performing DA policies based on the given dataset and task type.

1.4 Research Questions

RQ1: What are the newest improvements in Software Engineering (SE) that can be utilized to perform automated data augmentation in a practical manner?

RQ2: How to design an automated data augmentation system that can outperform human-designed data augmentation heuristics?

RQ3: What are the potential challenges that may occur while designing an automated data augmentation system, and how to prevent and overcome them?

1.5 Research Aim & Objectives

1.5.1 Research Aim

The aim of this research is to design, develop and evaluate a system that automates the manual process of designing and fine-tuning image data augmentation policies for any given low-diversity dataset with reduced human intervention.

To further elaborate on the research aim, this research work will develop a system having the capability of performing data augmentation for a given low-diversity dataset. To achieve that, this system aims to select the best-performing data augmentation policies based on the given low-diversity dataset and will tune the magnitude of chosen policies to improve the diversity of the given dataset. Furthermore, the proposed system and its elements will be evaluated and tested against output quality to validate the hypothesis chosen.

1.5.2 Research Objectives

Objective	Description	Learning Outcomes	Research Questions
Literature Review	In order to fulfill the below requirements, a survey of the existing AutoDA literature is conducted. **RO1:** To identify and study the existing systems in the AutoDA domain. **RO2:** To find out the limitations, improvements, and research gaps in the problem. **RO3:** To identify the ways of reducing computational power and complexity of existing AutoDA works without losing accuracy levels. **RO4:** To identify the technologies, algorithms, frameworks, and other required tools for the development phase of the project. **RO5:** To identify the benchmarking measurements and performance evaluation metrics.	LO1, LO6	RQ1, RQ2

Requirement Analysis	The requirement analysis of the proposed system was performed to, **RO6:** To determine the awareness of the risk of using random DA techniques to perform data augmentation. **RO7:** To gather requirements of an AutoDA framework and figure out expectations of end-clients. **RO8:** To gather domain and technical experts' insights and feedback to improve the proposed system.	LO2, LO3, LO4, LO5, LO6	RQ1, RQ2, RQ3
Design	Design the proposed AutoDA system architecture, **RO9:** To design search space where image data augmentation policy consists of. **RO10:** To design DA policy search and evaluation algorithms. **RO11:** To design a method to perform augmentation using identified DA policies.	LO3, LO4, LO6	RQ1, RQ2
Development	Developing the proposed AutoDA framework according to the identified design aspects, software, and hardware requirements, **RO12:** To develop search space, search algorithm, and evaluation algorithm of the proposed framework. **RO13:** To develop the GUI of the proposed framework. **RO15:** To develop core functionalities of the proposed framework using appropriate software and hardware requirements.	LO3, LO4, LO6	RQ1, RQ2
Testing and Evaluation	Testing and evaluating the proposed AutoDA methods' performance, **RO16:** To create a suitable test plan for functional and unit testing.	LO4, LO6	RQ1, RQ2

	RO17: To benchmark the prototype against accuracy and performance aspects. **RO18:** To get feedback from industry and academic experts.		
Publish Findings	To critically evaluate the research, **RO19:** Publish review survey paper about the AutoDA LR. **RO20:** Publish a research paper about the proposed solution and testing results of the proposed AutoDA system.	LO8	RQ1, RQ2, RQ3

Table 1: Research Objectives

1.6 Novelty of the Research

1.6.1 Novelty of the Problem

The performance of modern DL algorithms depends on the availability of big data (Shorten and Khoshgoftaar, 2019). However, it is hard to collect proper datasets in most of the real-world domains (Nanni et al., 2021; Shorten and Khoshgoftaar, 2019). For example, medical imaging. Data augmentation is a widely accepted solution to improve the low diversity datasets. However, selecting optimal DA policies based on the given dataset and task type is time-consuming and requires domain expertise (Shorten and Khoshgoftaar, 2019; Yang et al., 2022). AutoDA is a recently discussed concept to overcome the difficulties and limitations of traditional DA (Cubuk et al., 2019a). Recent research on the AutoDA domain has shown that rather than manually designing the DA policies, directly designing DA policies from the dataset and task type can significantly improve DL algorithm performance (Yang et al., 2022).

Existing implementations in the AutoDA domain have a series of limitations. For example, does not support to configure user-specific datasets and models and requires a large amount of computational resources and time to produce output (Cubuk et al., 2019b; Yang et al., 2022). These concerns about AutoDA limit the broad applicability of the AutoDA concept in real-world scenarios (Yang et al., 2022). Hence, it is essential to research ways of solving difficulties in AutoDA and make it available for public usage. Therefore, the novelty of the AutoDA problem domain remains.

1.6.2 Novelty of the Solution

As mentioned in the previous section, AutoDA is a relatively new concept, and it has a series of limitations, which also causes to limit the applicability of the AutoDA concept. High resource consumption and the requirement for CUDA to perform AutoDA tasks are one of the major disadvantages of existing solutions (Cubuk et al., 2019b; Müller and Hutter, 2021). Hence, eliminating the high resource consumption and CUDA requirement of existing AutoDA solutions will help to increase the usage of the AutoDA techniques in real-world DA scenarios.

Moreover, existing AutoDA models are pre-trained on CIFAR10/100 and ImageNet datasets, and it is complicated to configure those models into user-preferred datasets and image classification models due to the complexity (Müller and Hutter, 2021; Yang et al., 2022). Hence, the development of the plug-and-play AutoDA technique, which provides the ability to configure user-preferred datasets and image classification models easily, is essential.

A well-balanced dataset is an essential requirement of modern neural networks, and imbalanced datasets will cause a reduction in the performance of those models (Shorten and Khoshgoftaar, 2019). None of the existing AutoDA solutions consider the class imbalance issue of the given dataset during the pre-processing layer.

Attempting to develop an AutoDA architecture that provides above mentioned functionalities will be the novelty of the research solution.

1.7 Research Gap

After thoroughly reviewing the existing literature, the author identified several limitations of current research works, summarized below.

The major drawback of existing automated data augmentation techniques is that they are highly resource intensive. For instance, Auto Augment (Cubuk et al., 2019a) requires 15000 NVIDIA Tesla P100 GPU hours on the ImageNet dataset to perform automated image data augmentation tasks. This will limit the widespread applicability of Auto Augment (Cubuk et al., 2019a) in real-world, commercial applications. Hence, it is essential to solving this major drawback of high resource utilization in existing automated data augmentation techniques to use automated data augmentation techniques in practice.

Even though existing AutoDA models improve the performance of DL models on benchmarking datasets, still, these models are hard to use because they don't come with the controller module (Yang et al., 2022). The controller module exists between the user and the AutoDA model, allowing users to perform automated data augmentation for their own datasets. According to the literature, most of the existing AutoDA works are trained on CIFAR-10/100 and ImageNet datasets (Khalifa et al., 2022; Yang et al., 2022), which limits the usage of AutoDA models only to nature-related datasets. Hence, without the controller module, users cannot perform the AutoDA techniques on their own datasets. So, to use AutoDA models in practice, it is essential to develop plug-and-play AutoDA models with the controller.

Furthermore, none of the existing approaches considers the class imbalance issue while generating new data points. A dataset consists of multiple classes, and when the quantity of data samples belongs to one class is higher than the quantity of data samples belongs another class called as a class imbalance issue (Shorten and Khoshgoftaar, 2019). The imbalanced dataset causes the model to overfit. So, it is essential to generate well-balanced datasets (Yang et al., 2022).

Finally, none of the existing approaches gives the ability to toggle a specific type of data augmentation transformation in different application scenarios. For instance, when a user wants to remove some image data augmentation operations from the search space that are specific to their research work, the ability to toggle a particular type of data augmentation from the search space is essential. Moreover, this allows the designed DA policy to be more unique to the provided dataset and task type (Yang et al., 2022).

1.8 Contribution to the Body of Knowledge

As mentioned in previous sections, automated image data augmentation is promising because it allows searching for more powerful compositions of image transformations and parameterizations, increasing the generalizability of modern image classifiers.

The biggest challenge with automating data augmentation is to search over the image DA policy space (compositions of image transformations). However, due to a large number of image data augmentation policies and associated parameters in the search space, this can be prohibitively and computationally expensive by limiting the commercialization of automated data augmentation (Yang et al., 2022).

The main contribution of this project is to come up with a novel computationally affordable system that explores the search space of image data augmentation policies efficiently and effectively using less computational power, which also finds image augmentation strategies that can outperform human-designed image data augmentation policies. Hence, the proposed AutoDA system will be able to expand the broad applicability of automated image DA techniques in real-world applications.

As mentioned in the above section, existing AutoDA models don't come with the controller module, and most of the AutoDA models are pre-trained on nature-related datasets like CIFAR-10/100 and ImageNet. So, the best forming data augmentation policies found on existing AutoDA models are applicable only to nature-related datasets (Khalifa et al., 2022). Both these reasons cause to limit the applicability of AutoDA models on user-preferred datasets. So, another contribution will be attempting to develop a plug-and-play AutoDA model with a controller module, which allows users to perform AutoDA tasks on their datasets easily.

Due to the nature of this research, it must be noted that contributions to both the problem domain and research domain are similar in this project.

1.9 Research Challenges

The primary goal of this project is to address the limitations of the AutoDA literature and enhance the wide usage of automated data augmentation techniques in real-world applications. The following is a list of research challenges based on the proposed methodology.

1. **Improving efficiency and effectiveness** – Automating data augmentation for image classification tasks is a relatively novel concept and has not been thoroughly explored for real-world applications (Yang et al., 2022). Therefore, it is essential to explore theoretical aspects and other ways of defining automated data augmentation architectures more efficiently and effectively is a challenge.
2. **Improving the accuracy of the image classification model** - The usage of automated data augmentation techniques in practice is limited due to efficiency-related drawbacks. Recent works in automated data augmentation have improved the efficiency of automated data augmentation models, but their accuracy remains a bottleneck (Yang et al., 2022). Model accuracy is the key factor of modern image classifiers (Shorten and

Khoshgoftaar, 2019). Thus, improving efficiency while maintaining competitive accuracy will be a challenge.

3. **Data augmentation for sensitive tasks** - With reduced human intervention, generating accurately labeled new data points for sensitive tasks like medical image analysis and bioinformatics can be very difficult to obtain, especially when the dataset is imbalanced and noisy. So, dealing with the sensitive task with an imbalanced and noisy dataset will be another challenge.

1.10 Chapter Summary

This chapter discussed about the problem that is going to solve by this research work. Moreover, the research aim, the identified research novelty and gap, and the research challenges are also discussed by providing necessary evidence and a description of the DA problem domain. Additionally, the learning outcomes from this final project module matched with the research objectives of this project.

CHAPTER 2: SOFTWARE REQUIREMENT SPECIFICATION

2.1 Chapter Overview

The focus of this chapter is on identifying potential requirements and stakeholders of the system, and various requirement elicitation methods were used to collect the relevant data. Initially, a rich picture diagram and stakeholder onion model are used to define the proposed system's stakeholders and their interactions. The selected requirement elicitation methods and their findings are then discussed. After that, use case diagrams and descriptions are provided. Lastly, the chapter concludes with an analysis of the functionalities and non-functionalities of the proposed AutoDA system.

2.2 Rich Picture Diagram

The rich picture is a way of representing the structure, process, and concerns of a system from a bird's-eye view. The below rich picture diagram visualizes the identified structure, process, and concerns of the proposed AutoDA system.

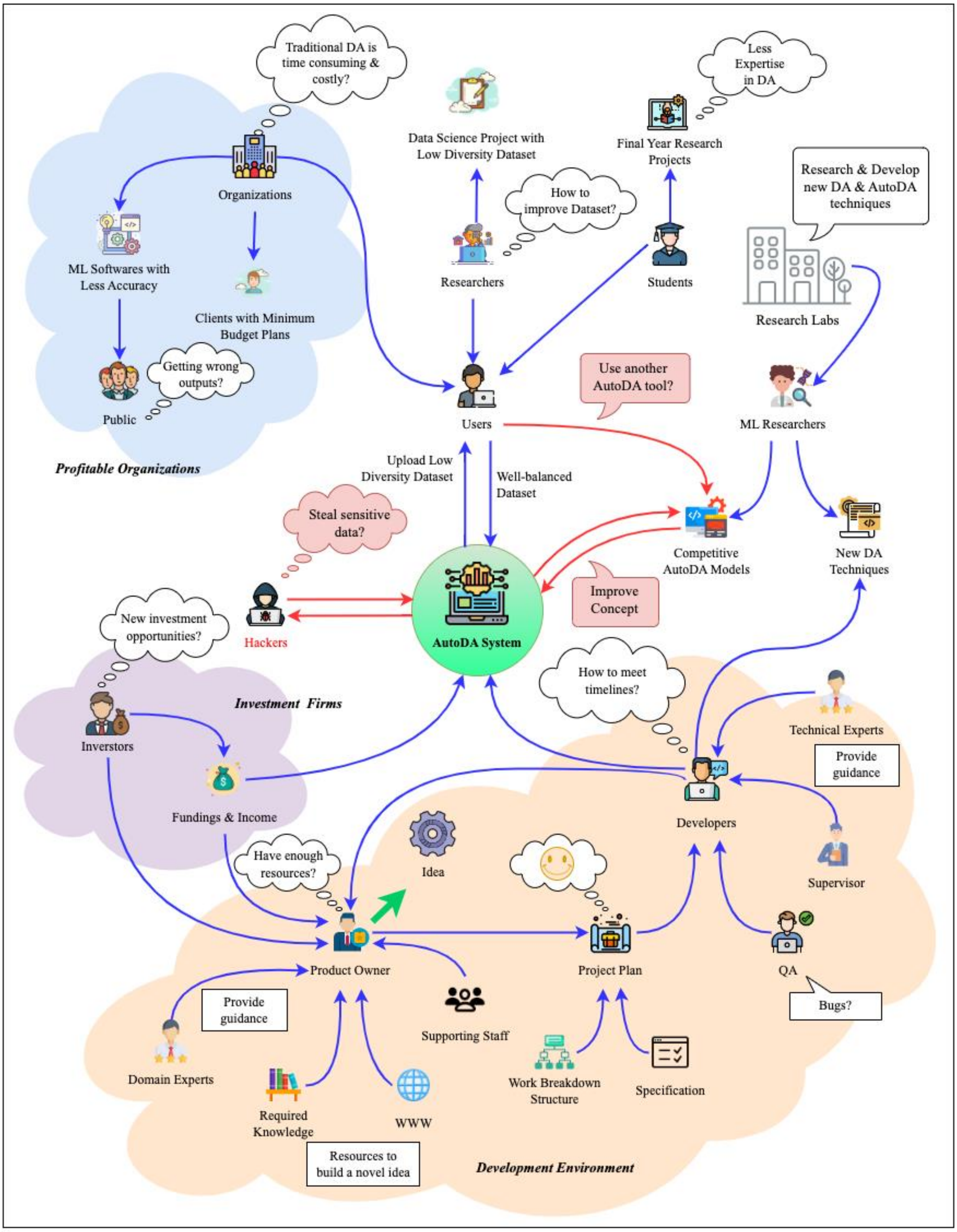


Figure 2: Rich Picture Diagram (self-composed)

2.3 Stakeholder Analysis

The below stakeholder onion model visualizes the identified stakeholders of the proposed AutoDA system. After that, a detailed elaboration of stakeholders of the proposed AutoDA system is also presented.

2.3.1 Stakeholder Onion Model Diagram

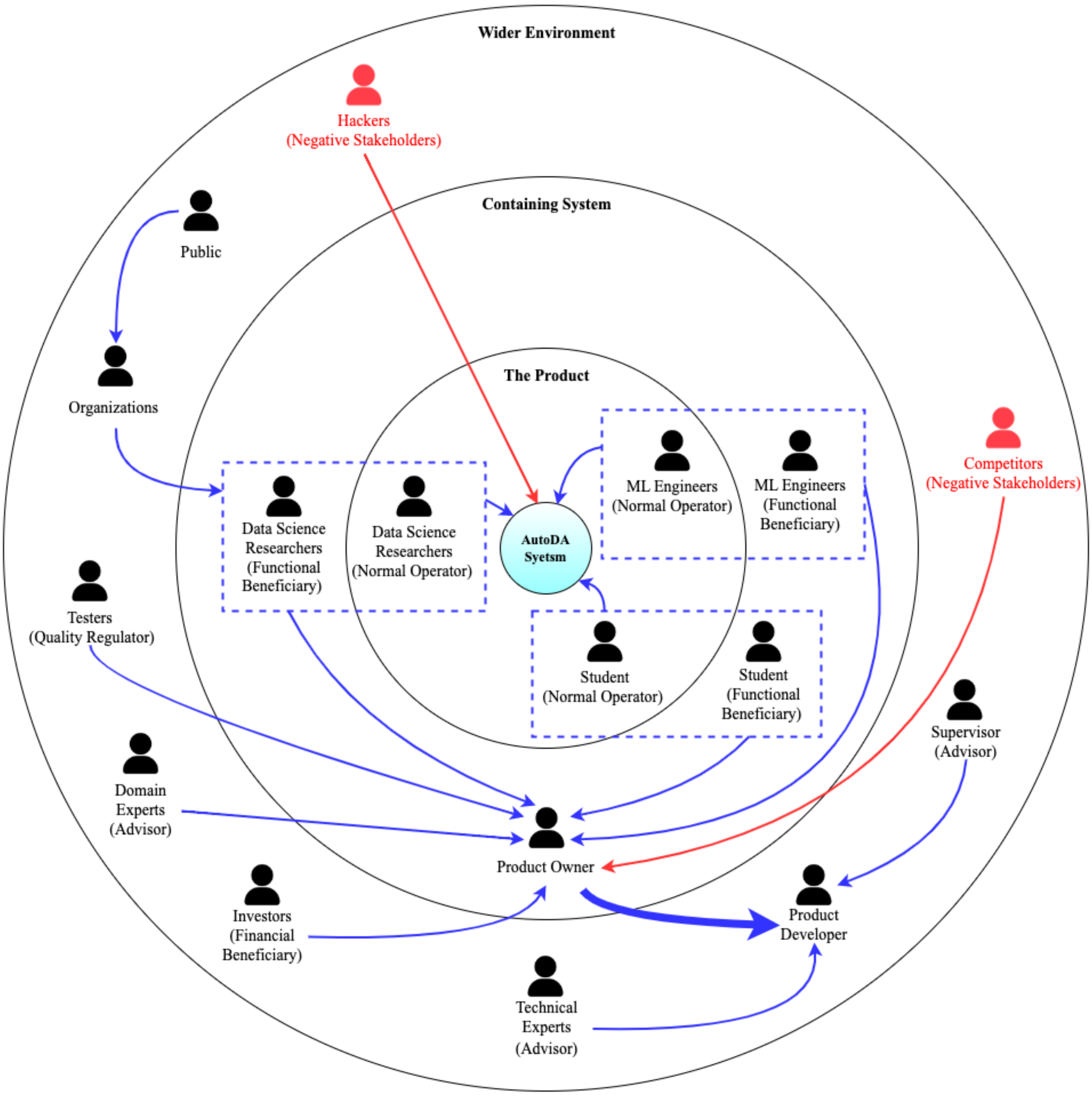


Figure 3: Stakeholder Onion Model (self-composed)

2.3.2 Analysis of the Stakeholders

Stakeholder	Role	Description
The Project Stakeholders		
ML Engineers	Normal operator	Perform AutoDA using the system for their own datasets.
Students		
Data science researchers		

Containing System Stakeholders		
Product owner	Operational beneficiary	Manage and supervise the entire business workflow and product engineers.
Students	Functional beneficiary	Further investigate the research theory presented in the proposed AutoDA system and improve it. Learn about automated data augmentation techniques.
ML Engineers	Functional beneficiary	Get the research hypothesis presented in the proposed system and improve it further. Develop new data augmentation techniques.
Organizations	Functional beneficiary	Use the system to improve their datasets and products.
Wider Environment Stakeholders		
Application developer	Development staff and operational maintainer	Develop and improve the proposed system and fix errors when occurred.
Supervisor	Quality regulator / Advisor	Evaluate the product and provide feedback to improve.
Domain Experts		
Technical Experts		
Testers		
Investors	Financial beneficiary	Invest in the proposed system and gain profit from the audience.
Hackers	Negative stakeholder	Attempt to bring down the system and disable system services.
Competitive AutoDA models	Negative stakeholder	Develop an improved version of AutoDA systems.

Table 2: Analysis of the Stakeholders

2.4 Selection of Requirement Elicitation Methodologies

Requirement elicitation is a way to find out the requirements of a SE project. In order to gather software requirements for this project, among the available requirement elicitation methodologies, LR, interviews, and prototyping methods were chosen. The descriptive reasons for selecting the above requirement elicitation methodologies are outlined below.

Method 1: Literature review
A good LR helps to figure out the research gaps and problems in the existing systems. Hence, at the initial stage of this project, the author did a thorough analysis of the existing AutoDA literature domain, existing AutoDA systems, possible novel approaches, and technologies to solve the current limitation of AutoDA systems.
Method 2: Interviews
Domain and technical experts' insights are very important to validate the identified research idea and research gap. Moreover, with the help of their knowledge, it is easy to identify the best possible way and technologies to solve the identified problem. Additionally, since the AutoDA research domain is still a relatively new research domain. Hence interviews are one of the best approaches to gathering requirements.
Method 3: Prototyping
Since the final aim of this research work is to explore and develop novel architecture to work around the limitations of existing AutoDA systems, prototyping methodology was chosen because prototyping methodology enables the author to perform continuous improvements, testing, and evaluations to find out different ways to make the system better.

Table 3: Requirement Elicitation Methodologies

2.5 Analysis on the Findings

2.5.1 Literature Review

Finding	Citation
Existing AutoDA techniques are highly resource intensive. Hence, it is essential to solving this high resource utilization in existing AutoDA techniques to use AutoDA techniques in real-world commercial applications.	(Lim et al., 2019)
It is challenging to reimplement many published AutoDA methods, limiting the broad applicability of AutoDA techniques. Hence, it is essential to develop plug-and-play AutoDA frameworks which are easy to set up.	(Müller and Hutter, 2021)

Search-free AutoDA methods are more efficient. However, their accuracy is comparatively less than the search-based AutoDA methods.	(Yang et al., 2022)
RandAugment by the Google Brain team consider the most practical AutoDA system, but it can be further enhanced.	(Müller and Hutter, 2021; Yang et al., 2022)
Imbalanced datasets are led to model overfitting and poor generalizability of neural networks.	(Yang et al., 2022)

Table 4: Findings through Literature Review

2.5.2 Interviews

To get the viewpoints of domain and technical experts, the experts in ML, DL, and data science were chosen to be interviewed. A Google expert in ML and AI, an AI & data research graduate, two Ph.D. students in data science, a SE in AI & data, two data engineers, and two senior students were interviewed. Based on the following themes, **thematic analysis** was processed on the outcome of interviews.

Codes	Theme	Analysis
DA Difficulties, Awareness of AutoDA, AutoDA Difficulties, Importance of AutoDA	Research gap and depth of scope	All the participants declared that most DL algorithms do not perform well if they are trained with low-diversity datasets, and data augmentation is a key technique to improve those low-diversity datasets. Also, they stated that selecting specific data augmentation techniques based on the dataset and model is time-consuming. Hence, they thought that resolving the limitations of existing AutoDA methods and making them available for public usage is essential and innovative.
Understating Common DA	Understanding the most used data augmentation techniques	Participants stated that based on their experience working in computer vision and data augmentation, geometric transformations, color space transformations, and kernel filters are the most used data augmentation techniques.

Availability, Tunings, Improvements	Features and suggestions for prototype	Apart from the ability to select the best data augment policies for a given dataset and model, there were a few suggestions for the prototype that were stated by the participants, which are listed below. • Rank the data augmentation techniques for the given dataset and results in comparison with the top 3 suggested techniques. • Automatically identify how to balance out the imbalanced dataset. • Availability as a python package.
The necessity of the AutoDA, Applicable Domains	The necessity of plug-and-play AutoDA system and contributions	All the participants clearly stated that AutoDA is the future of DA because AutoDA techniques will save the time of the developer as it limits the number of experiments that need to be performed to identify the best-performing DA technique for a given dataset. As a result, it will reduce project costs. Moreover, it addresses the knowledge limitations of the developer as implementing DA techniques needs technical and domain knowledge. Ultimately, all the participants thought that, since the AutoDA domain is still a relatively new concept and addressing the limitations of it would be very helpful to the domain.

Table 5: Findings through Interviews

2.5.3 Prototyping

The author utilized prototyping to do the following.

Criteria	Findings
To validate the effective magnitude range of geometric transformations and color space transformations.	The author has identified and validated the 14 different DA techniques (including geometric transformations and color space transformations) with their effective magnitude range for any dataset by conducting a series of porotypes using the knowledge gained from the literature.

To monitor the computational resource consumption and verify the proposed solution is resource friendly.	Since the author has narrowed down the DA policy magnitude search range according to the literature findings and prototype findings, the optimal DA policy search time has been reduced. Moreover, all the experiments conducted on Apple M1 Pro MacBook with 16GB ram smoothly; therefore, it is proven that the proposed solution can also be used by ordinary users.
To evaluate differentiable DA technique approach will give the expected results.	RandAugment by the Google Brain team (Cubuk et al., 2019b) is the widely accepted and most practical approach in the literature; the author has identified that it can be further enhanced by using differentiable data augmentation operations while consuming less computational resources by conducting a series of porotypes. Additionally, differentiable data augmentation operations help to improve the image classification model accuracy since it doesn't use fixed magnitude as RandAugment.

Table 6: Findings through Prototyping

2.5.4 Summary of Findings

Id	Finding	Literature Review	Interviews	Prototyping
1	Identified research gaps in the AutoDA domain need to be filled to use AutoDA techniques in practice.	X	X	
2	Automate the entire data augmentation process, including image classification model training and turning.	X	X	X
3	Should automate the basic image manipulation data augmentation techniques for the initial prototype phase.	X	X	X
4	Provide the ability to add or remove data augmentation techniques from the search space.	X	X	
5	Should consider the class imbalance problem before the data augmentation.	X	X	X
6	Rank the data augmentation techniques for the given dataset and results comparison with the top suggested techniques.		X	

7	Users who don't know much about data augmentation and model training should be able to understand how the system works.	X	X	
8	Should use a standard programing language that utilizes the majority of machine learning models and datasets.	X	X	
9	Availability as a python package.	X	X	
10	Should have a GUI to make it easier to use.		X	

Table 7: Summary of Findings

2.6 Context Diagram

The below context diagram (alias level 0 data flow diagram (DFD)) visualizes the high-level interactions of the DAugtelligent with its high-level environment, including inputs and outputs of the DAugtelligent system.

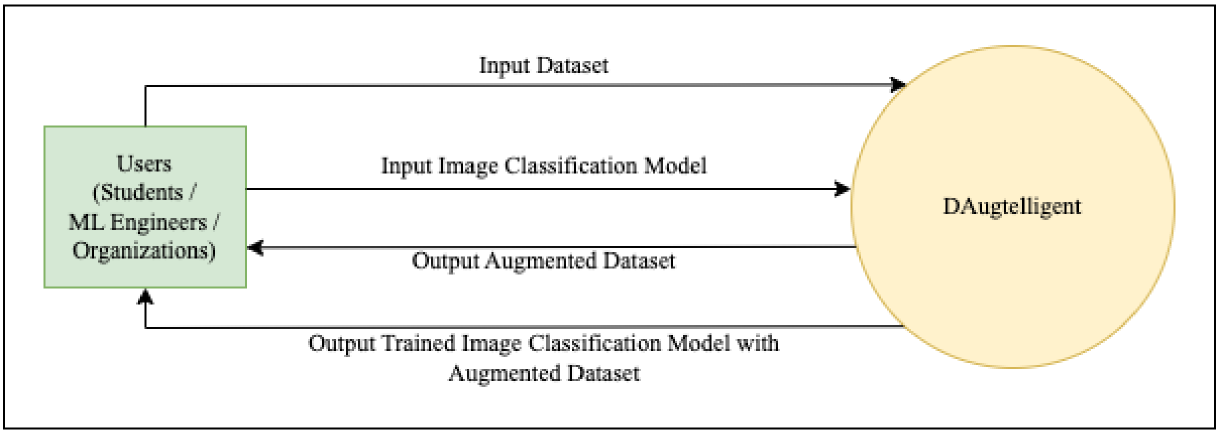


Figure 4: Context Diagram (self-composed)

2.7 Use Case Diagram

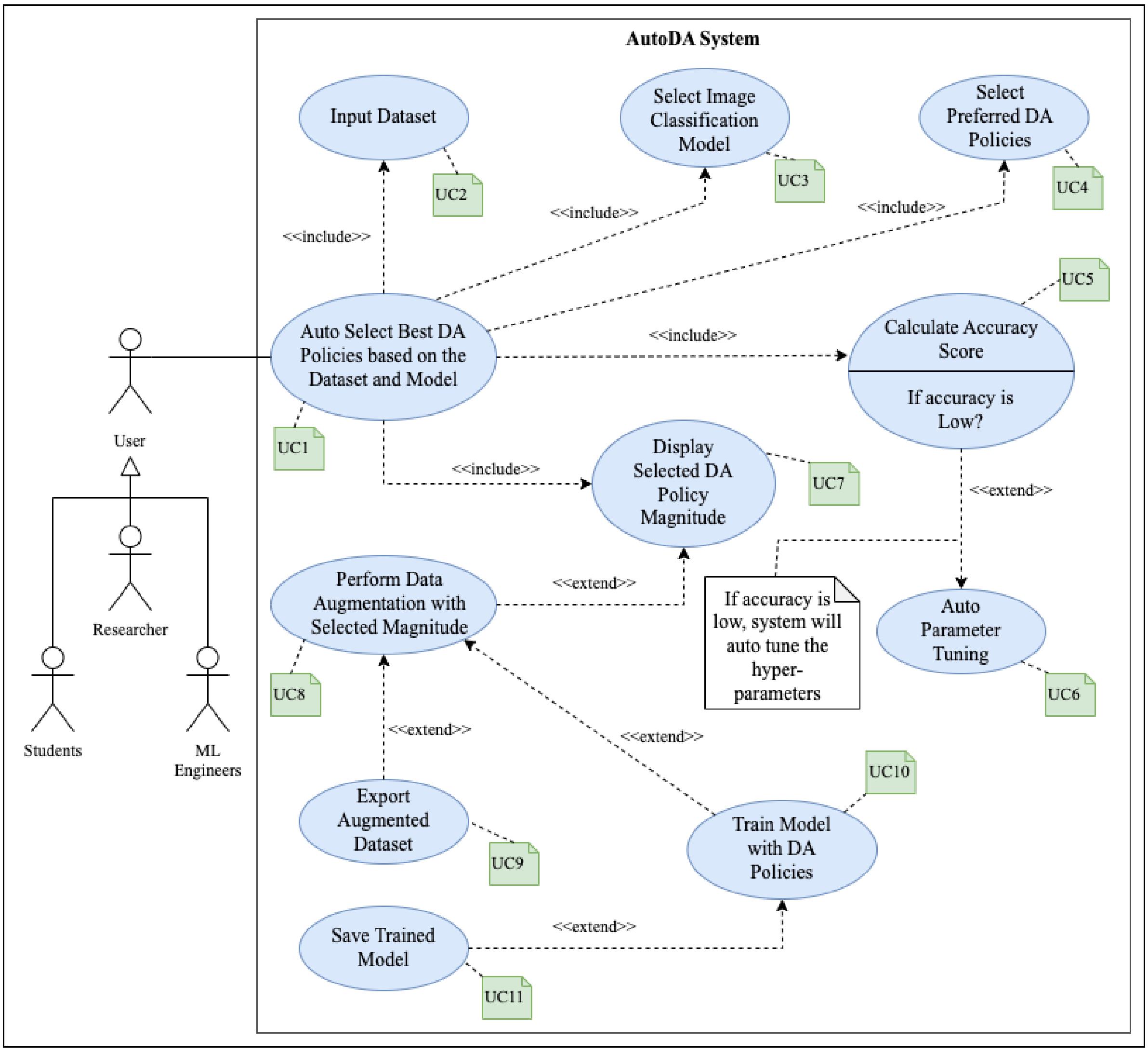


Figure 5: Use Case Diagram (self-composed)

2.8 Use Case Descriptions

Use case name	Auto Select Best DA Policies based on the Dataset and Model
ID	UC1
Description	User is able to find out best-performing DA policies for their dataset that improve the model performance.
Participating actors	User
Preconditions	User must select their dataset, image classification model, and preferred DA policies.
Extended use cases	None

<table>
<tr><td>Included use cases</td><td colspan="2">Input Dataset, Select Image Classification Model, Select Preferred DA Policies, Calculate Accuracy Score, Display Selected DA Policy Magnitude</td></tr>
<tr><td rowspan="2">Main Flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Select the dataset, image classification model, and preferred DA policies.

2. User executes AutoDA task on their dataset and image classification model.

5. Preview results.</td><td>3. Find common DA policy magnitude for user-preferred DA policies based on the user's dataset and image classification model.

4. Display optimal common DA policy magnitude.</td></tr>
<tr><td>Alternative flows</td><td colspan="2">None</td></tr>
<tr><td>Exceptional flows</td><td colspan="2">E1: User did not input the dataset.
2.1 Automated data augmentation will not be executed.
E2: User did not input the image classification model.
2.2 Automated data augmentation will not be executed.
E3: User did not input both the dataset and image classification model.
2.3 Automated data augmentation will not be executed.
E4: User did not select the preferred DA policy set.
2.3 Automated data augmentation will be executed with the default DA policy set.</td></tr>
<tr><td>Post conditions</td><td colspan="2">The system will display the optimal DA policy magnitude.</td></tr>
</table>

Table 8: Use Case Description for Auto Select Best DA Policies based on the Dataset and Model

Use case name	Perform Data Augmentation with Selected Magnitude
ID	UC8
Description	Once the AutoDA module finds the optimal DA policy magnitude for the user's dataset and image classification model, the user is able to perform DA applying that magnitude.
Participating actors	User

<table>
<tr><td>Preconditions</td><td colspan="2">AutoDA module must complete the execution and out the optimal DA policy magnitude user selected DA policies.</td></tr>
<tr><td>Extended use cases</td><td colspan="2">Export Augmented Dataset, Train Model with DA Policies.</td></tr>
<tr><td>Included use cases</td><td colspan="2">None</td></tr>
<tr><td rowspan="2">Main flow</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Click on the ‘Perform DA’ button.</td><td>2. Start DA using the optimal magnitude found by the AutoDA module.

3. Display the ‘Download Augmented Dataset’ and ‘Train Image Classification Model with Augmented Dataset’ buttons.</td></tr>
<tr><td>Alternative flows</td><td colspan="2">None</td></tr>
<tr><td>Exceptional flows</td><td colspan="2">None</td></tr>
<tr><td>Post conditions</td><td colspan="2">Display the ‘Download Augmented Dataset’ and ‘Train Image Classification Model with Augmented Dataset’ buttons.</td></tr>
</table>

Table 9: Use Case Description for Perform Data Augmentation with Selected Magnitude

2.9 Requirement Specification

The ‘MoSCoW’ technique is utilized to determine the priority levels of the identified functionalities and non-functionalities of the proposed AutoDA system.

Priority Level	Description
Must have (M)	The requirements that are important and essential to develop a successful Minimum Viable Product (MVP).
Should have (S)	The requirements that are important but not essential to develop a successful MVP. But the system will have some limitations.
Could have (C)	The requirements that are not important and essential to develop a successful MVP.
Will not have (W)	The requirements that are not developed during the initial stage of MVP.

Table 10: Summarization of "MoSCoW" prioritization levels

2.9.1 Functional Requirements

FR ID	Requirement	Priority Level	Use Case
FR01	Users must be allowed to choose their preferred dataset.	M	UC2
FR02	Users must be allowed to choose the preferred image classification model.	M	UC3
FR03	Users must be allowed to choose their preferred DA techniques among the available DA techniques.	M	UC4
FR04	System should be able to identify and solve the class imbalance issue of the selected dataset if present.	S	UC6
FR05	Based on the given dataset and image classification model, the system must be able to find the best-performing DA techniques with their magnitudes among the selected DA techniques.	M	UC1
FR06	System should be able to show the ranking of auto selected DA techniques with their magnitude.	S	UC7
FR07	System must be able to train the given image classification model with auto selected DA techniques.	M	UC10
FR08	Users should be able to download the augmented dataset.	S	UC9
FR09	Users must be able to download the trained image classification model with augmented data.	M	UC11
FR10	Users could have the ability to tune the given image classification model hyper-parameters.	C	UC6

Table 11: Functional Requirements

2.9.2 Non-Functional Requirements

NFR ID	Requirement	Description	Priority Level
1	Usability	The goal of the proposed AutoDA system is to tackle the difficulties in traditional DA. Hence users must be able to perform AutoDA tasks even without expertise in the DA domain.	M

2	Output Quality	Since the quality of the augmented images makes a considerable amount of impact on a given image classification mode, it is important to maintain augmented image quality.	M
3	Performance	The system should not take too much time to produce a response. It should be able to manage available computational resources and perform AutoDA tasks.	S
4	Scalability	The system should be able to perform AutoDA techniques in larger datasets and handle the workload smoothly during the upcoming stages.	C
5	Security	The system should have the proper defense mechanisms to prevent any attacks.	W

Table 12: Non-Functional Requirements

2.10 Chapter Summary

This chapter intends to figure out the functionalities and non-functionalities of the proposed project. To begin with, the rich picture diagram and Saunder's Onion model were used to identify and represent possible stakeholders and their interactions with the proposed system. Then, the author discussed about the requirement-gathering methodologies which were utilized to gather required data and opinions from the identified stakeholders. Then, the use case diagram of the proposed AutoDA system and functional as well as non-functional requirements were presented based on the gathered data using the relevant requirement elicitation methodologies. Lastly, the priority levels of the identified functionalities and non-functionalities were determined using the "MoSCoW" requirements prioritization technique.

CHAPTER 3: SYSTEM ARCHITECTURE AND DESIGN

3.1 Chapter Overview

This is chapter discusses the design conclusions that were taken to come up with an architecture for prototype development of the proposed AutoDA system. During the requirement-gathering phase of this project, the author conducted a series of interviews and reviews on AutoDA LR and prototyping findings. All the design decisions mentioned in this chapter were based on those requirement-gathering phase findings. Moreover, in order to express prototype system architecture, the proposed AutoDA system architecture decisions, user interfaces, and other required design diagrams have been presented.

3.2 Design Goals

Design Goal	Description
Output Quality	The correction of selected optimal DA policies and their magnitude should be as good as possible. Describing why a user is getting the suggested DA policies using the given image classification model performance metrics.
Usability	One of the main objectives of this project is to overcome the optimal DA policy selection phase difficulties and reduce the complexity of the implementation. Therefore, it is important to develop the system straightforwardly to the dwindling number of clicks required to complete AutoDA tasks. Additionally, from the developer's point of view, the proposed AutoDA concept should easy-to-learn even without prior knowledge in DA.
Performance	Most of the existing AutoDA systems are not able to use by the ordinary audience due to the high computational resource consumption. Hence, it is essential that the system will perform even with limited computational resources.
Scalability	The system should be able to work with large datasets properly while optimizing the available computational resources.
Extendibility	Initially proposed system supports 14 DA technologies. However, there are a lot of DA technologies out there. Hence, the system should build with the best software practices (SOLID principles) in mind to make sure that while

	extending the system with more DA technologies does not cause any problems.

Table 13: Design Goals of the proposed system

3.3 System Architecture Design

3.3.1 System Architecture Diagram

The below illustration visualizes the three-tier architecture of the proposed AutoDA system.

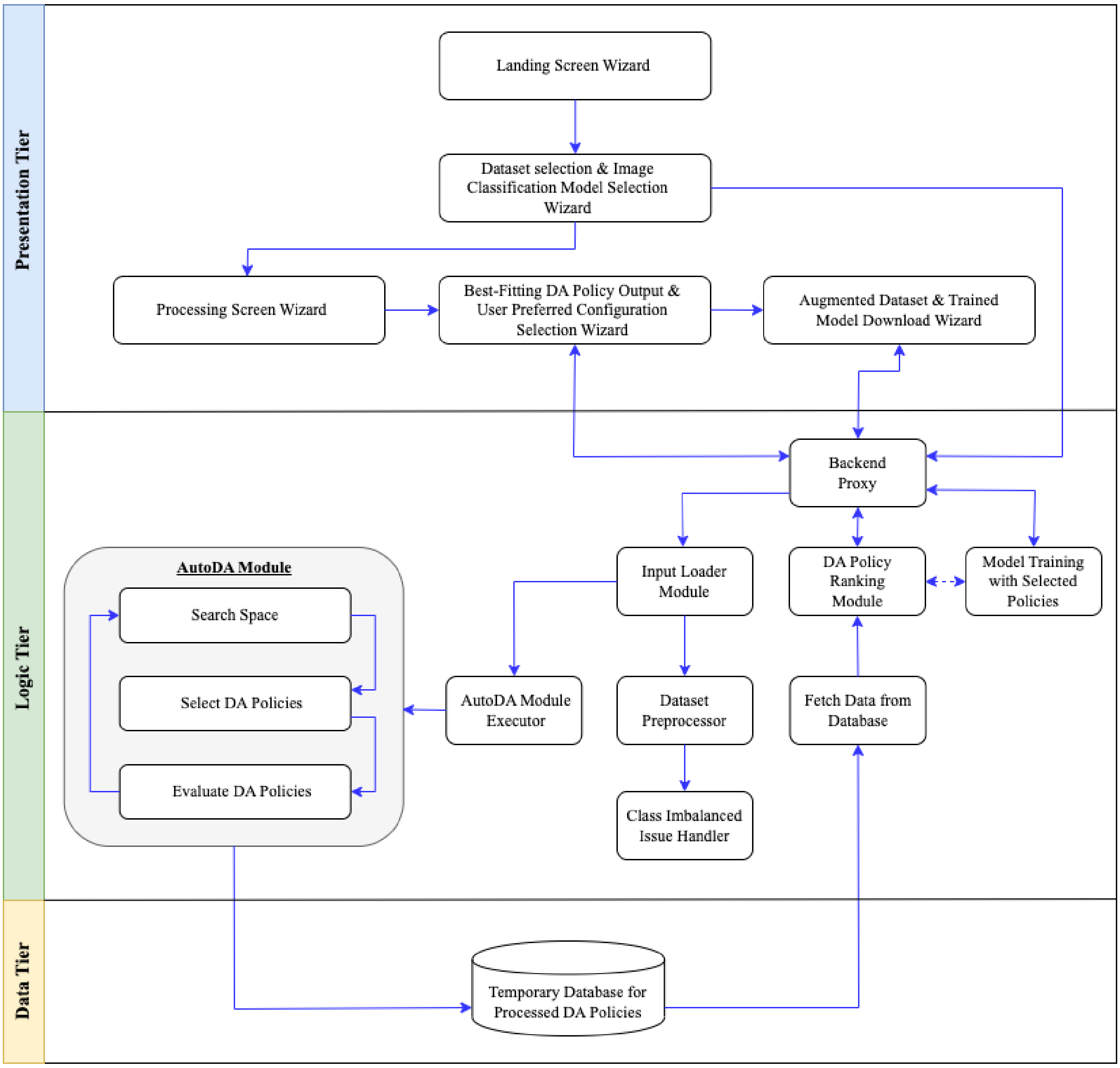


Figure 6: Three Tiered Architecture (self-composed)

3.3.2 Discussion of System Architecture Tiers

As illustrated in the above diagram, the architecture of the proposed AutoDA system was designed based on the three-tier architecture concept (presentation (client), logic, and data tiers) to ensure the reliability of the system. The client tier is responsible for capturing all the user interactions, while the logic tier is responsible for executing all the logical operations based on the user interactions. And the data tier stores all necessary data for the proposed AutoDA system.

Data Tier

1. Temporary Database for Processed DA Policies – This data storage allows storing already processed DA policies by AutoDA module with their magnitude and the given image classification model accuracy.

Logic Tier

1. Input Loader Module – This module is to fetch and pre-process the user-selected dataset and image classification model into the system. Additionally, it will validate the given inputs.
2. Class Imbalanced Issue Identifier – This module will identify if a given dataset is unbalanced and take necessary actions to make it balanced.
3. AutoDA Module – The core of the system which selects the best-performing DA policies based on the given dataset and image classification model. This component consists of two subcomponents.
    a. Search Space – All the available DA policies. At the initial stage, this module consists of 14 DA policies which are Identity, Shear, Translate, Rotate, Cutout, Contrast, Auto Contrast, Equalize, Solarize, Solarize Add, Posterize, Color, Brightness, and Sharpness.
    b. Evaluation Function – This module is responsible for validating the performance of selected DA policies from the search space.
4. DA Policy Ranking Module – This module is to rank the selected DA policies from the AutoDA module based on the given image classification model performance.
5. Model Training with Selected Policies – This module is to retrain the given image classification model with the user preferred DA policies.

Client Tier

1. Landing Screen Wizard – This page will consist with the system introduction and 'Get Started' functionalities.
2. Dataset selection & Image Classification Model Selection Wizard – This page will allow users to select their dataset and image classification model.
3. Processing Screen Wizard – Since it takes time for the AutoDA module to process and find out the best-performing DA policies, this page will display the progress of that task.

4. Best Fitting DA Policy Output & User Preferred Configuration Selection Wizard – This page will show the selected DA policies with their magnitude founded by the AutoDA module. Additionally, this page will allow users to select the user-preferred DA policies to retrain the given image classification model.
5. Augmented Dataset & Trained Model Download Wizard – This page will allow users to download the well-balanced & optimized dataset and trained image classification model.

3.4 System Design

3.4.1 Selection of the Design Paradigm

There are two different design paradigms that are normally used for the software development process that is:

1. Object Oriented Analysis and Design (OOAD)
2. Structured Systems Analysis and Design Method (SSADM)

The goal of this project is to come up with a novel approach to solve difficulties in existing AutoDA implementations. To archive this goal, it is essential to conduct a set of experiments and rapid changes to the code. Among the OOAD and SSADM, SSADM provides a more precise and easy process to improve existing software systems. Hence, the **SSADM** is ideal for this research project.

3.5 Design Diagrams

3.5.1 Component Diagram

The below illustration depicts the modules, main components, and sub-components of the proposed AutoDA system with their interactions and relationships.

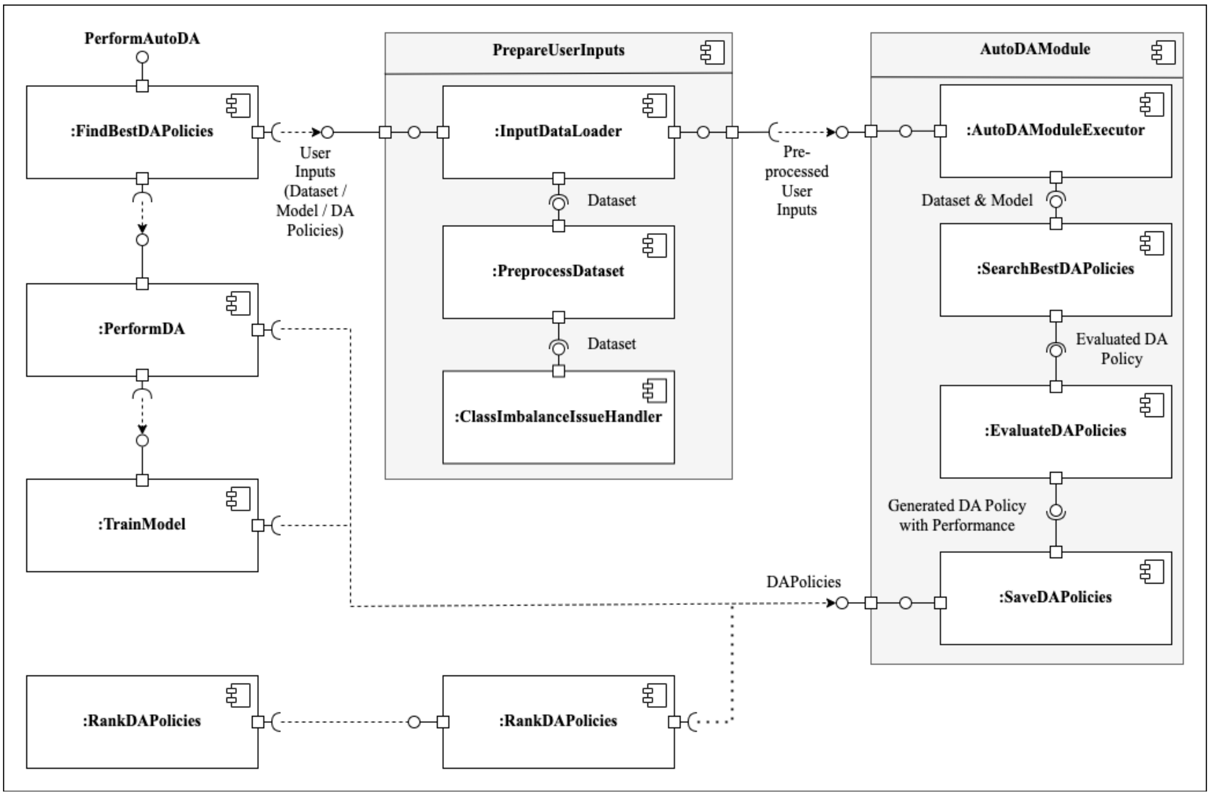


Figure 7: Component Diagram (self-composed)

3.5.2 Data Flow Diagram

The below illustration visualizes the level 1 DFD of the proposed AutoDA system, which provides additional details of the level 0 DFD. The level 0 DFD of the proposed AutoDA system was described during the SRS chapter.

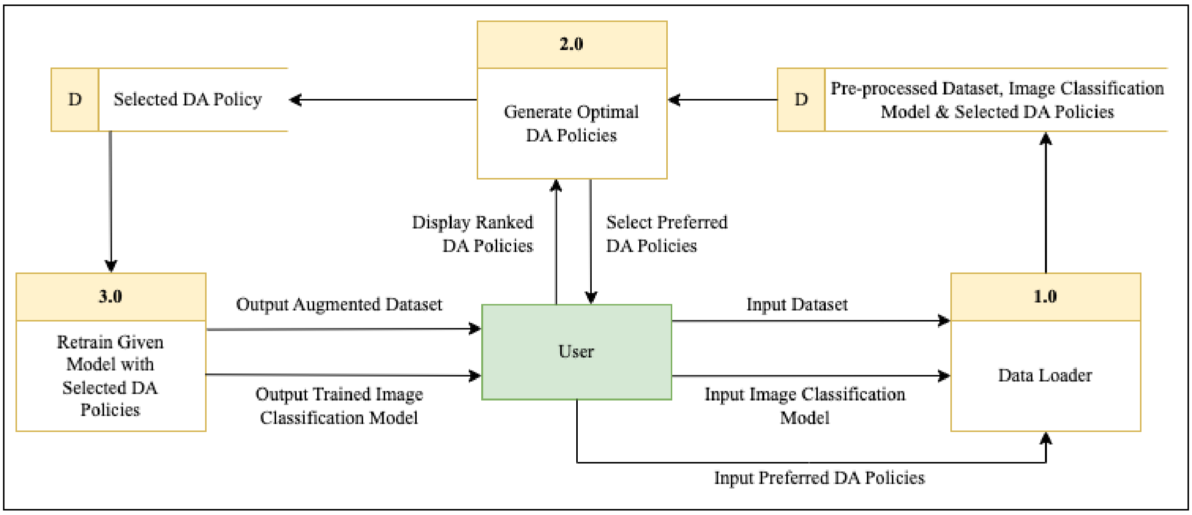


Figure 8: Data Flow Diagram - Level 1 (self-composed)

The level 1 DFD consists of 3 major data processes. Which are

1. Data loader
2. Generate optimal DA policies

3. Train given image classification model with user-preferred DA policies among the generated DA policies

The below illustrations depict level 2 data flow diagrams of the above-mentioned major processes, which provides a more detailed breakdown of the level 1 DFD major processes.

The below illustration visualizes level 2 DFD of the data loader module.

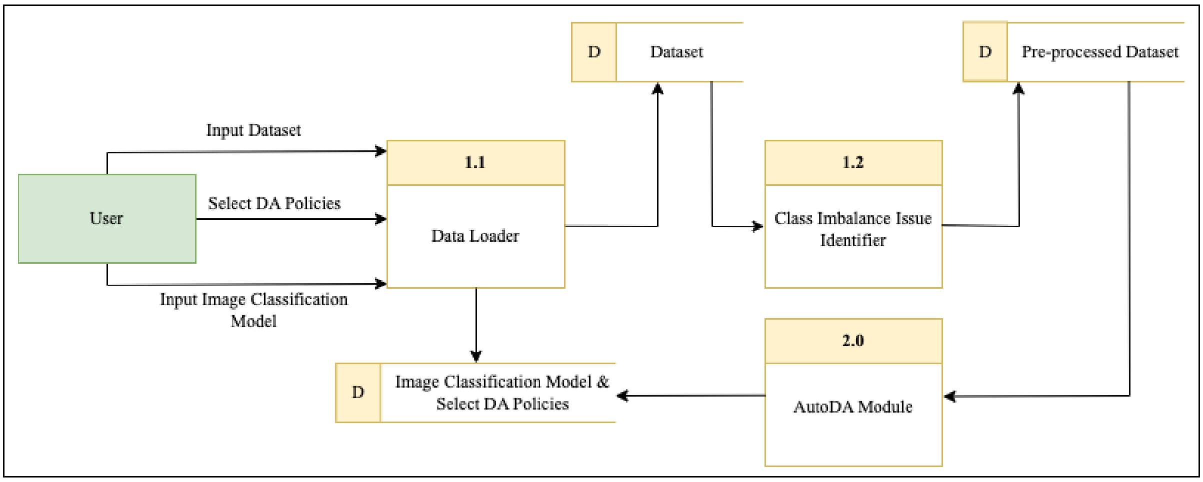


Figure 9: Data Loader - Data Flow Diagram - Level 2 (self-composed)

The below illustration depicts level 2 DFD of the AutoDA (core) module.

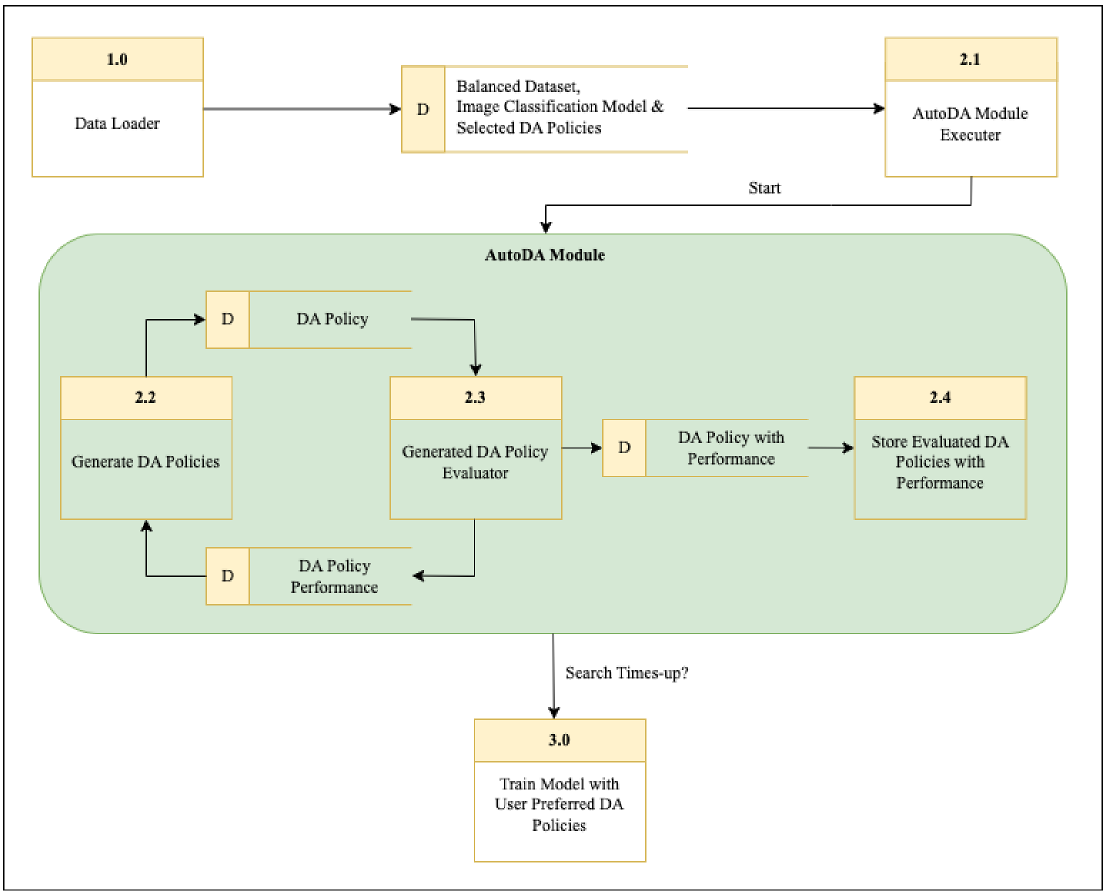


Figure 10: AutoDA Module - Data Flow Diagram - Level 2 (self-composed)

The below illustration depicts level 2 DFD of the given image classification model train module.

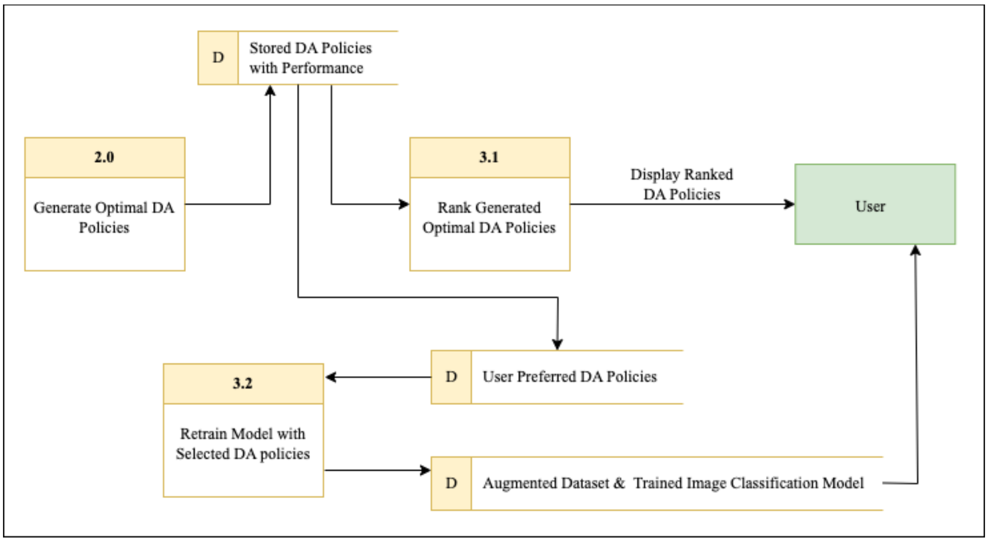


Figure 11: Train Image Classification Module - Data Flow Diagram - Level 2 (self-composed)

3.5.3 Utilization of Differentiable Programming into AutoDA

The author utilized Differentiable Programming (DP) in the AutoDA domain by examining the following question:

Given a dataset and an image classification task, how can we find and adjust the common magnitude for all DA policies to maximize the given image classification model performance?

To find a solution to this question using Differentiable Programming, the author intends to design and evaluate a neural network called **Differentiable Policy Augment Module**. This neural network contains trainable common magnitude parameter for all DA policies. Further, this neural network can learn the best-performing common magnitude for all DA policies using differentiable DA operations.

The below illustration depicts the integration of the Differentiable Policy Augment Module neural network with the given image classification model. Moreover, the implementation of the Differentiable Policy Augment Module neural network and differentiable DA operations have been discussed in the Implementation chapter.

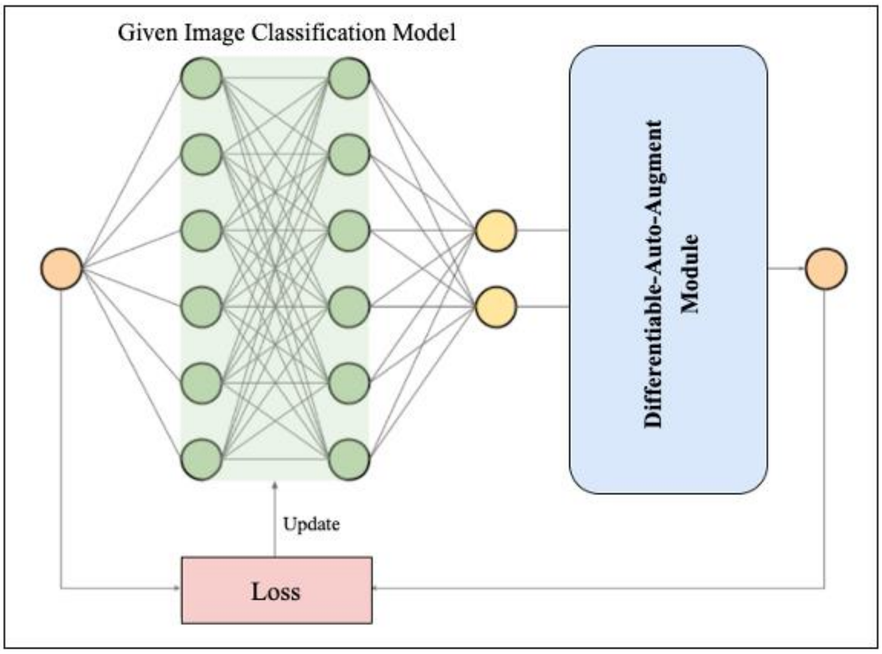


Figure 12: Integration of the Differentiable Policy Augment Module (self-composed)

According to the author's best knowledge and the AutoDA domain literature findings, none of the existing approaches consider the utilization of Differentiable Programming to find a common magnitude for all DA policies. Hence, attempting to solve the current limitations of AutoDA by utilizing Differentiable Programming can be identified as the core contribution of the research.

3.5.4 System Activity Diagram

The below illustration depicts the activity diagram of the proof-of-concept system. It stimulates the general workflow of the prototype system, including user inputs, processes, and outputs based on the developer's point of view.

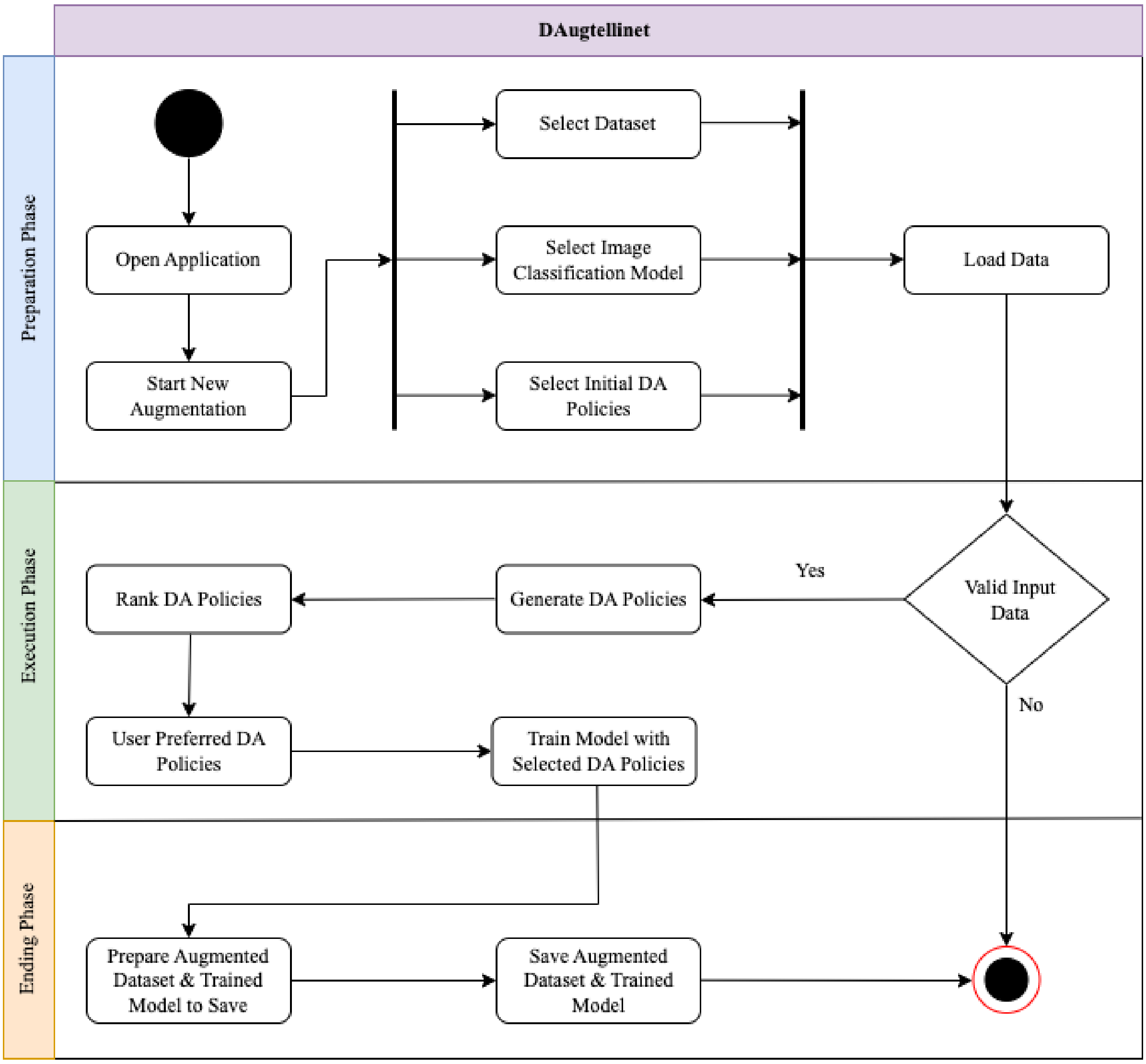


Figure 13: System Activity Diagram (self-composed)

3.5.5 User Interface Design

Based on the requirements gathered during interviews, the best way to simulate this research is to build a Python library. However, to fill the academic requirements, the author comes up with a straightforward UI in which users can easily perform AutoDA tasks and train their image classification model.

The below illustration depicts the planned high-level fidelity designs of the main input and output configuration screens of the AutoDA system. The high-fidelity designs for other screens have been placed in **Appendix C**, and the low-level fidelity wireframes have been placed in **Appendix D**.

Figure 14: User-preferred Dataset, Model & DA Policy Selection Wizard (self-composed)

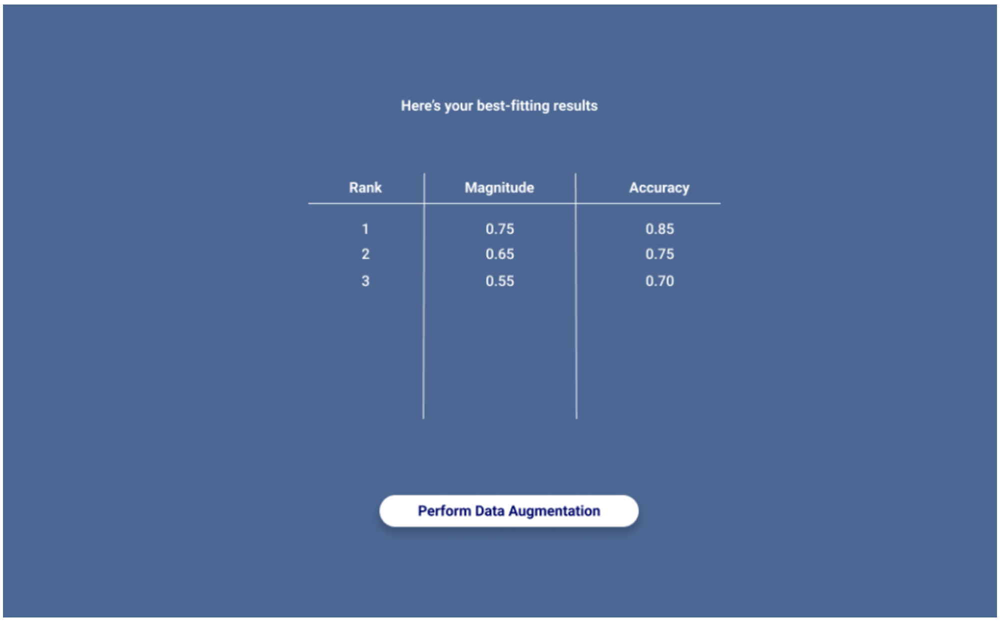


Figure 15: AutoDA Module Output Display Wizard (self-composed)

3.6 Chapter Summary

This chapter intends to discuss the design goals and system architecture aspects of the proposed AutoDA system. To begin with, the author has demonstrated the design goals of the proposed AutoDA system with the proper justification. Then three-tier system architecture diagram and other required design diagrams, along with the data flow diagrams, were presented. Later, the core contribution, which is the utilization of Differentiable Programming into the AutoDA concept, was discussed. Lastly, initial UI designs for MVP have been presented.

CHAPTER 4: IMPLEMENTATION

4.1 Chapter Overview

This chapter discusses the development of core functionalities of the proposed AutoDA system, proving the research hypothesis. Moreover, the author discusses the selected technology stack, programming languages, and additional tools that are utilized to develop the prototype, along with the respective reasons for each selection. Lastly, this chapter discusses how the design decisions were translated into an MVP by demonstrating the required code snippets of the initial implementation.

4.2 Technology Selection

4.2.1 Technology Stack

The technology stack that was utilized to develop the three-tier architecture of the proposed AutoDA system is described in this illustration.

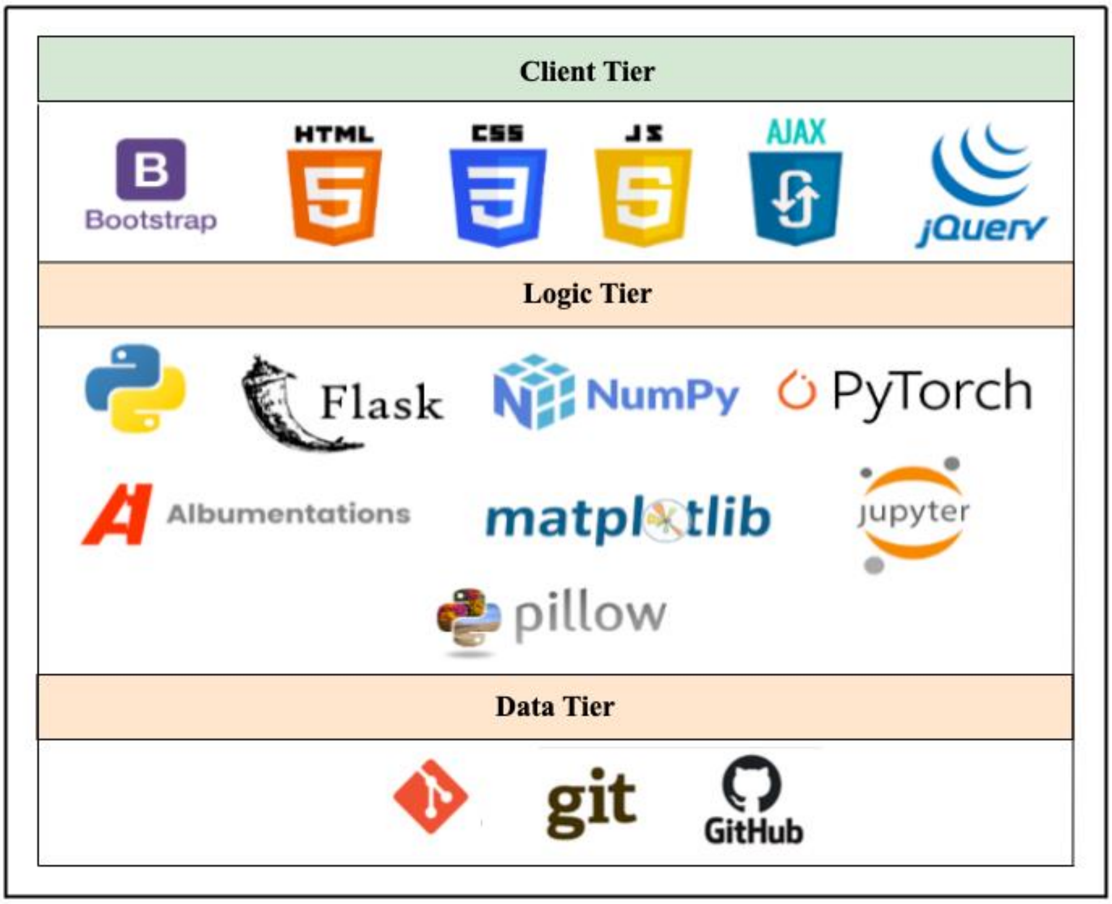


Figure 16: Technology Stack

4.2.2 Programming Language Selection

Programming Language	Justification for Selection
Python	Among the available programming languages, Python has been utilized in most of the ML projects due to the vast collection of community and supporting libraries.
JavaScript	For the front end, it is easy to make a highly interactive and inviting user experience using JavaScript.

Table 14: Justifications for Programming Language Selection

4.2.3 Development Framework Selection

Development Framework	Justification for Selection
PyTorch	Most of the existing AutoDA works used PyTorch for their implementation. Hence to do a fair comparison, PyTorch was selected. Moreover, the benchmark proves that the performance of PyTorch is better than the TensorFlow.
Flask	Easy to build Application Programming Interfaces (APIs) for Python.

Table 15: Justifications for Development Framework Selection

4.2.4 Libraries

Libraries	Justification for Selection
Numpy	Supports an extensive variety of mathematical and algebraic development.
Matplotlib	Supports an extensive variety of visualization methods for use in data analysis.
Albumentations	Supports an extensive variety of DA augmentation methods.
Pillow	Supports an extensive variety of image-related operations like opening and saving images.

Table 16: Justifications for Development Libraries Selection

4.2.5 IDE Selection

IDE	Justification for Selection
Google Colab	Since the author doesn’t have access to GPU resources and Google Colab is free to use, it was utilized to run the experiments on GPUs.

VSCode	Powerful IDE to build both front-end and back-end while being very simple and flexible to use.

Table 17: Justifications for IDE Selection

4.2.6 Summary of Technologies and Tools Selection

Component	Tools
Programming Languages	Python, JavaScript
Development Frameworks	PyTorch, Flask
Libraries	Albumentations, Numpy, Matplotlib, Pillow
IDEs	VSCode
Version Control	Git, GitHub

Table 18: Summary of Technologies and Tools Selection

4.3 Implementation of Core Features

As mentioned, the goal of this project is the find optimal DA policies based on the given dataset and image classification model. In this section, the author discusses the generation and evaluation of the best possible DA policies for a given dataset and task.

According to the Differentiable Programming concept, it requires three things. Which are:

1. A parameterized method to optimized
2. A loss value to measure the current performance
3. A differentiable model to optimized

Based on these requirements, the author implemented the following:

- Parameterized DA functions named differentiable data augmentation policies
- A neural network that can learn the best performance magnitude called a differentiable policy augment module
- Utilized automatic gradient descent algorithm to measure the current loss

Differentiable Data Augmentation Policies

After doing a thorough LR and conducting a series of interviews with domain and technical experts, the author concluded that the majority of users use 14 DA techniques. Which are:

- Identity
- Shear-X & Shear-X
- Sharpness
- Auto Contrast
- Equalize
- Solarize

- Translate-X & Y
- Rotate
- Cutout
- Posterize
- Color
- Brightness

So, the proposed system will only consider these DA operations during the initial stage.

```python
class BaseOperation(ABC):
    """Abstract class of image augmentations.
    Args:
        min_val, max_val: The minimum and maximum magnitude for each operation.
        quantize_magnitude: If `True`, quantize the magnitude by converting the scaled
            value to `int` type. Default is `False`.
    """
    def __init__(self, *args: float, quantize_magnitude: bool = False):
        if len(args) != 0 and len(args) != 2:
            raise ValueError(
                f"expected 0 or 2 arguments, but given {len(args)} arguments"
            )

        self.magnitude_range = args
        self.quantize_magnitude = quantize_magnitude

    def __call__(
        self, x: Union[np.ndarray, torch.Tensor], magnitude: Union[float, torch.Tensor]
    ) -> Union[np.ndarray, torch.Tensor]:
        """Perform an image augmentation.
        Args:
            x: The input image. It should be numpy array or torch tensor.
            magnitude: The master magnitude.
        Returns:
            The augmented image. It will have same type of the input.
        """
        if self.magnitude_range:
            min_val, max_val = self.magnitude_range
            magnitude = magnitude * (max_val - min_val) + min_val

        if self.quantize_magnitude:
            if isinstance(magnitude, torch.Tensor):
                magnitude = magnitude.long()
            else:
                magnitude = int(magnitude)

        if isinstance(x, np.ndarray):
            return self.apply_numpy(x, magnitude)
        elif isinstance(x, torch.Tensor):
            return self.apply_tensor(x, magnitude)
        else:
            raise TypeError(f"type {type(x)} is not allowed")

    @abstractmethod
    def apply_numpy(self, x: np.ndarray, value: float) -> np.ndarray:
        ...

    @abstractmethod
    def apply_tensor(self, x: torch.Tensor, value: torch.Tensor) -> torch.Tensor:
        ...
```

Figure 17: Implementation of Abstract Class for DA Operations

This is an abstract class for DA operations, and every DA operation must inherit from this class and implement apply_tensor and apply_numpy methods. These methods will apply a corresponding transformation based on the input image type (NumPy or Tensor). For example, if the user enters the NumPy image array, the apply_numpy method will be executed.

Ultimately, all DA operations support the differentiation of the input image and tensor magnitude. Therefore, it can calculate the gradient difference between the input image and tensor magnitude. Below code snippets depict the implementation of Auto Contrast and Color data augmentation techniques.

```
class AutoContrast(BaseOperation):
    """Normalize the image contrast.
    This class maximize the image contrast by remapping the lowest value to `0` and the
    highest value to `0xFF`. It makes the darkest pixels to be black and the lightest
    ones to be white.
    Note:
        This class does not use the magnitude value. So you do not need to specify the
        range of the magnitude.
    """

    def apply_numpy(self, x: np.ndarray, value: float) -> np.ndarray:
        print('dev test AutoContrast apply_numpy')
        min_val = x.min(axis=(0, 1), keepdims=True)
        max_val = x.max(axis=(0, 1), keepdims=True)

        x = (x - min_val) / (max_val - min_val + 1e-6)
        return np.clip(0xFF * x, 0, 0xFF).astype(np.uint8)

    def apply_tensor(self, x: torch.Tensor, value: torch.Tensor) -> torch.Tensor:
        print('dev test AutoContrast apply_tensor')
        min_val = x.amin(dim=(2, 3), keepdim=True)
        max_val = x.amax(dim=(2, 3), keepdim=True)
        return (x - min_val) / (max_val - min_val + 1e-6)

    def get_transform_name():
        return 'AutoContrast'
```

Figure 18: Implementation of Auto Contrast DA Policy

```
class Color(BaseOperation):
    """Adjust the color balance of the image.
    This class blends the original image and its grayscale image. The magnitude controls
    the blending ratio of them.
    """

    def apply_numpy(self, x: np.ndarray, value: float) -> np.ndarray:
        print('dev test Color apply_numpy')
        return np.clip(
            value * x + (1 - value) * x.mean(2, keepdims=True), 0, 0xFF
        ).astype(np.uint8)

    def apply_tensor(self, x: torch.Tensor, value: torch.Tensor) -> torch.Tensor:
        print('dev test Color apply_tensor')
        return torch.clamp(value * x + (1 - value) * x.mean(1, keepdim=True), 0, 1)

    def get_transform_name():
        return 'Color'
```

Figure 19: Implementation of Color DA Policy

As mentioned earlier, the proposed system will only consider 14 DA operations during the initial stage. Based on the LR and prototype findings, the author conculcated that each DA policy has an effective magnitude range for any given dataset. By doing so, the author was able to reduce the time for searching for the best-performing common magnitude for all DA policies. Moreover, the system requires user-preferred DA policies as input. If users do not specify their preferred DA policies, the system will take the default DA policy set. Below code, snippets depict the default DA policy set with their effective magnitude range.

```
DefaultOpSet = [
    Identity(),
    ShearX(0, 0.3),
    ShearY(0, 0.3),
    TranslateX(0, 0.45),
    TranslateY(0, 0.45),
    Rotate(0, 30),
    Cutout(0, 0.2),
    AutoContrast(),
    Equalize(),
    Solarize(0, 0xFF),
    SolarizeAdd(0, 0x6E),
    Posterize(0, 4),
    Contrast(0.1, 1.9),
    Color(0.1, 1.9),
    Brightness(0.1, 1.9),
    Sharpness(0.1, 1.9),
]
```

Figure 20: Implementation of Default DA Policy Set

Differentiable Policy Augment Module

```
class DifferentiablePolicyAugmentModule(nn.Module):
    """Trainable `DifferentiablePolicyAugmentModule` module.
    Args:
        num_ops: The number of operations.
        normalized: If `True`, the input image is normalized to `[-1, 1]`. In this case,
            this class automatically rescale to `[0, 1]` during the operations and then
            restores to original scale. Default is `True`.
        opset: The operation set to use in `RandAugment`. Default is `DefaultOpSet`.
    """

    def __init__(
        self,
        num_ops: int,
        normalized: bool = True,
        opset: List[BaseOperation] = DefaultOpSet,
    ):
        super().__init__()
        self.num_ops = num_ops
        self.normalized = normalized
        self.opset = opset

        self.magnitude_logits = nn.Parameter(torch.empty(()).normal_(0, 1))

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        # Normalize the image to [0, 1].
        if self.normalized:
            x = (x + 1) / 2

        magnitude = self.magnitude_logits.sigmoid()
        for _ in range(self.num_ops):
            op = random.choice(self.opset)
            print("dev test: ", op)
            x = op(x, magnitude)


        # Normalize the image to [-1, 1].
        if self.normalized:
            x = 2 * x - 1
        return x

    def get_magnitude(self) -> float:
        """Return the trained magnitude value."""
        print('dev test: ', self.magnitude_logits.detach().cpu().sigmoid().item())
        return self.magnitude_logits.detach().cpu().sigmoid().item()
```

Figure 21: Implementation of Differentiable Policy Augment Module

This neural network contains trainable common magnitude parameter for all DA policies. As per the Differentiable Programming concept, this neural network can learn the best-performing common magnitude parameter for all DA policies using the automatic gradient descent algorithm and differentiable DA policies discussed above. Initially, this class will start with a random magnitude, and after every training batch, it will automatically adjust the common magnitude parameter value while decreasing the loss.

Auto Data Augment Module

```
class AutoDataAugment(A.ImageOnlyTransform):
    """Image augmentor class`.
    Args:
        num_ops: The number of operations.
        normalized: If `True`, the input image is normalized to `[-1, 1]`. In this case,
            this class automatically rescale to `[0, 1]` during the operations and then
            restores to original scale. Default is `True`.
        opset: The operation set to use in `RandAugment`. Default is `DefaultOpSet`.
    """

    def __init__(
        self, num_ops: int, magnitude: float, opset: List[BaseOperation] = DefaultOpSet
    ):
        super().__init__(always_apply=False, p=1.0)
        self.num_ops = num_ops
        self.magnitude = magnitude
        self.opset = opset

    def apply(self, x: np.ndarray, **params: Any) -> np.ndarray:
        for _ in range(self.num_ops):
            op = random.choice(self.opset)
            x = op(x, self.magnitude)
        return x
```

Figure 22: Implementation of Auto Data Augment Module

Lastly, the common magnitude parameter found by the Differentiable Policy Augment Module will be passed to the Auto Data Augment Module. This module will perform the DA for a given dataset using the common magnitude and train the given image classification model with augmented data.

4.4 Chapter Summary

This chapter intends to discuss the technology stack, programming languages, and additional tools that are utilized to develop the proposed research prototype. Moreover, the implementation of the core functionalities of the proposed research project was discussed with relevant code snippets which prove the research hypothesis and contributions.

CHAPTER 5: CONCLUSION

5.1 Chapter Overview

This chapter discusses the deviations from the initial project proposal, the initial test results of the ongoing project development, and the required improvements that the author wishes to archive before the final prototype demonstration. Lastly, the YouTube link to the demo video, which presents the current progress of the project, and also the GitHub link to the initial implementation code is attached.

5.2 Deviations

5.2.1 Scope Related Deviations

During the initial project proposal, the author proposed support for any given image classification model. However, due to the time constraint that the author has to deal with, this requirement is limited to a few commonly used pre-defined image classification models. Other than this, there are no significant scope-related deviations, and all other proposed in-scope functionalities will be demonstrated during the MVP.

5.2.2 Schedule Related Deviations

As of now, February 2023, there are no schedule-related deviations. The below table describes all major states of this project with their deadlines.

Task	Deadline	Status
Project initiation	November 2022	Completed
Initial LR	November 2022	Completed
Requirement gathering	January 2023	Completed
Designing of the System	January 2023	Completed
Selection of tools and technologies	January 2023	Completed
Prototype implementation	March 2023	In progress
Testing and evaluation	March 2023	In progress
Dissertation and documentation	April 2023	In progress

Table 19: Schedule Related Deviations

As planned in the Gantt chart, the extended LR analysis, implementation, and testing of MVP will continue till March 2023.

5.3 Initial Test Results

For the initial test runs, the author utilized the CIFAR10 dataset and Wide ResNet 28x10 neural architecture.

The CIFAR10 dataset contains 600,000 32x32 color image samples (50,000 training and 10,000 testing image samples) organized into ten categories (automobile, airplane, cat, bird, dog, deer, horse, frog, truck, and ship), and each data category consists of 6,000 images. Moreover, the CIFAR10 dataset is widely used as a benchmarking dataset to determine the performance of modern image classification algorithms.

Wide ResNet 28x10 is a DL algorithm designed for image classification tasks. These Wide ResNet neural networks are designed are designed to increase the network's capacity and improve its performance. To archive this, the depth of the network is reduced, and the width of the network is increased. The selected Wide ResNet model consists of 28 network layers and 10 widening factors. The widening factor determines how many filters are used in each layer. This architecture has been used in many image classification competitions and benchmarks.

The author conducted the initial testing utilizing the below criteria. However, since this is an initial test accuracy of the Wide ResNet 28x10 model will be considered.

Criteria	**Test Results**
Train and test Wide ResNet 28x10 network with CIFAR10 without data augmentation.	`loss: 2.4165 - accuracy: 0.6699` *Figure 23: Accuracy of the Model without DA*
Train and test Wide ResNet 28x10 network with CIFAR10 with random data augmentation policies and magnitude. In this test, Rotate and Flip is used as data augmentation methods.	`loss: 0.8419 - accuracy: 0.7375` *Figure 24: Accuracy of the Model with Random DA Policies*
Train and test Wide ResNet 28x10 network with CIFAR10 with data augmentation policies and magnitude generated by AutoDA module.	As per the initial test results, the proposed system was able to produce competitive results. After the ten epochs, the AutoDA module found that **0.3** was the best-performing magnitude for

<table>
<tr>
<td></td>
<td>all DA policies, along with the 86% accuracy of the Wide ResNet 28x10 network. Below is the screenshot of the initial test results for the AutoDA module.

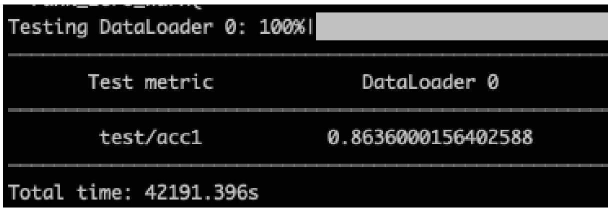

Figure 25: Founded Common Magnitude


Figure 26: Initial Test Accuracy</td>
</tr>
</table>

Table 20: Initial Test Results

Based on the test results, we can conclude that the DA policies generated by the AutoDA module improve the performance of the Wide ResNet 28x10 neural network.

5.4 Required Improvements

The core research component of this project is DA policy generation and evaluation based on the given dataset and image classification model (AutoDA module). Moreover, since this Project Specification & Prototype Design submission required a core research component of the proposed system, the author completed the development of the AutoDA module. However, before the final project submission, the author wishes to improve below task items:

- Pre-process layer, which handles the class imbalance issue of a given dataset. This layer will help to increase the performance of the given image classification model further.
- Developments of user interfaces to improve the user experience.
- Cloud deployment of AutoDA module to improve the performance.

5.5 Demo of the Prototype

Link to the initial implementation demonstration video: **https://youtu.be/ntzS2O5wED4**
Link to the code: **https://github.com/pubudu-m/DAugtelligent**

5.6 Chapter Summary

This chapter intends to discuss the deviations from the initial project proposal and initial test results of the current implementation. The scope-related deviations were addressed using the in-scope requirement of the initial project proposal, and schedule-related deviations were addressed using the proposed Gantt Chart. Later, the author discussed about the initial test results and the testing criteria. Based on the initial test results, the author concluded that the proposed system archived competitive results. However, it can further improve based on expert feedback. Lastly, the link to the demo video, which presents the current progress, and the link to the code have been presented.

REFERENCES

Alves, J.H., Oliveira, L.F., 2020. Optimizing Neural Architecture Search using Limited GPU Time in a Dynamic Search Space: A Gene Expression Programming Approach. undefined. https://doi.org/10.1109/CEC48606.2020.9185856

Cai, H., Zhu, L., Han, S., 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware.

Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V., 2019a. AutoAugment: Learning Augmentation Strategies From Data, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 113–123. https://doi.org/10.1109/CVPR.2019.00020

Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V., 2019b. RandAugment: Practical automated data augmentation with a reduced search space.

Fawzi, A., Samulowitz, H., Turaga, D., Frossard, P., 2016. Adaptive data augmentation for image classification, in: 2016 IEEE International Conference on Image Processing (ICIP). Presented at the 2016 IEEE International Conference on Image Processing (ICIP), pp. 3688–3692. https://doi.org/10.1109/ICIP.2016.7533048

Hataya, R., Zdenek, J., Yoshizoe, K., Nakayama, H., 2020. Meta Approach to Data Augmentation Optimization.

Ho, D., Liang, E., Stoica, I., Abbeel, P., Chen, X., 2019. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules.

Jeong, J.J., Patel, B., Banerjee, I., 2022. GAN augmentation for multiclass image classification using hemorrhage detection as a case-study. J. Med. Imaging 9, 035504. https://doi.org/10.1117/1.JMI.9.3.035504

Khalifa, N.E., Loey, M., Mirjalili, S., 2022. A comprehensive survey of recent trends in deep learning for digital images augmentation. Artif. Intell. Rev. 55, 2351–2377. https://doi.org/10.1007/s10462-021-10066-4

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 84–90. https://doi.org/10.1145/3065386

Lee, D., Park, H., Pham, T., Yoo, C.D., 2020. Learning Augmentation Network via Influence Functions, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern

Recognition (CVPR). Presented at the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10958–10967. https://doi.org/10.1109/CVPR42600.2020.01097

Leevy, J.L., Khoshgoftaar, T.M., Bauder, R.A., Seliya, N., 2018. A survey on addressing high-class imbalance in big data. J. Big Data 5, 42. https://doi.org/10.1186/s40537-018-0151-6

Li, C., Zhang, J., Hu, L., Zhao, H., Zhu, H., Shan, M., 2022. In-and-Out: a data augmentation technique for computer vision tasks. J. Electron. Imaging 31, 013023. https://doi.org/10.1117/1.JEI.31.1.013023

Li, P., Liu, X., Xie, X., 2021. Learning Sample-Specific Policies for Sequential Image Augmentation, in: Proceedings of the 29th ACM International Conference on Multimedia. Presented at the MM ’21: ACM Multimedia Conference, ACM, Virtual Event China, pp. 4491–4500. https://doi.org/10.1145/3474085.3475602

Li, Y., Hu, G., Wang, Y., Hospedales, T., Robertson, N.M., Yang, Y., 2020. DADA: Differentiable Automatic Data Augmentation.

Lim, S., Kim, I., Kim, T., Kim, C., Kim, S., 2019. Fast AutoAugment.

Lin, C., Guo, M., Li, C., Xin, Y., Wu, W., Lin, D., Ouyang, W., Yan, J., 2019. Online Hyper-parameter Learning for Auto-Augmentation Strategy.

Liu, C., Zoph, B., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K., 2018. Progressive Neural Architecture Search. undefined. https://doi.org/10.1007/978-3-030-01246-5_2

Müller, S.G., Hutter, F., 2021. TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation.

Nanni, L., Paci, M., Brahnam, S., Lumini, A., 2022. Feature transforms for image data augmentation. Neural Comput. Appl. https://doi.org/10.1007/s00521-022-07645-z

Nanni, L., Paci, M., Brahnam, S., Lumini, A., 2021. Comparison of Different Image Data Augmentation Approaches. J. Imaging 7, 254. https://doi.org/10.3390/jimaging7120254

[PDF] A Comprehensive Survey of Image Augmentation Techniques for Deep Learning | Semantic Scholar, n.d. https://doi.org/10.48550/arXiv.2205.01491

[PDF] A comprehensive survey of recent trends in deep learning for digital images augmentation | Semantic Scholar, n.d.

[PDF] Circumventing Outliers of AutoAugment with Knowledge Distillation | Semantic Scholar, n.d.

[PDF] DADA: Differentiable Automatic Data Augmentation | Semantic Scholar [WWW Document], n.d. URL https://www.semanticscholar.org/reader/197c376d433fc693c9ea197cc3fb50b1c6ea23f8 (accessed 10.17.22).

Pham, H., Guan, M., Zoph, B., Le, Q.V., Dean, J., 2018. Efficient Neural Architecture Search via Parameter Sharing. undefined.

Ratner, A.J., Ehrenberg, H.R., Hussain, Z., Dunnmon, J., Ré, C., 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal Policy Optimization Algorithms.

Shorten, C., Khoshgoftaar, T.M., 2019. A survey on Image Data Augmentation for Deep Learning. J. Big Data 6, 60. https://doi.org/10.1186/s40537-019-0197-0

Takase, T., Karakida, R., Asoh, H., 2020. Self-paced Data Augmentation for Training Neural Networks.

Tan, M., Le, Q.V., 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. undefined.

Tang, Z., Peng, X., Li, T., Zhu, Y., Metaxas, D., 2019. AdaTransform: Adaptive Data Transformation, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). Presented at the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2998–3006. https://doi.org/10.1109/ICCV.2019.00309

Terauchi, A., Mori, N., 2021. Evolutionary Approach for AutoAugment Using the Thermodynamical Genetic Algorithm. Proc. AAAI Conf. Artif. Intell. 35, 9851–9858. https://doi.org/10.1609/aaai.v35i11.17184

Yang, Z., Sinnott, R.O., Bailey, J., Ke, Q., 2022. A Survey of Automated Data Augmentation Algorithms for Deep Learning-based Image Classication Tasks.

Zhang, R., Liang, Y., Somayajula, S.A., Xie, P., 2021. Improving Differentiable Architecture Search with a Generative Model. https://doi.org/10.48550/arXiv.2112.00171

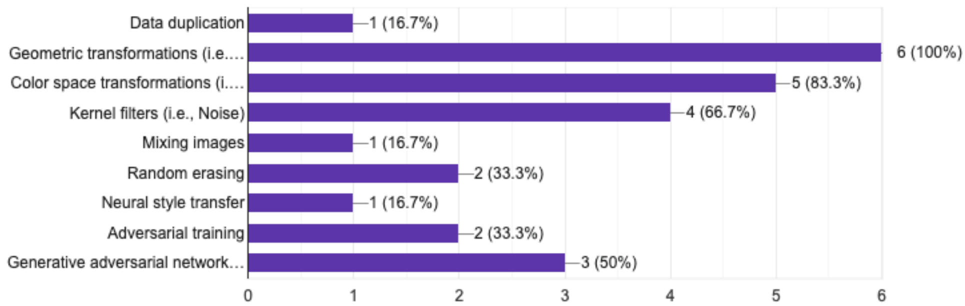
Zhang, X., Wang, Q., Zhang, J., Zhong, Z., 2019. Adversarial AutoAugment.

Zoph, B., Le, Q.V., 2017. Neural Architecture Search with Reinforcement Learning. https://doi.org/10.48550/arXiv.1611.01578

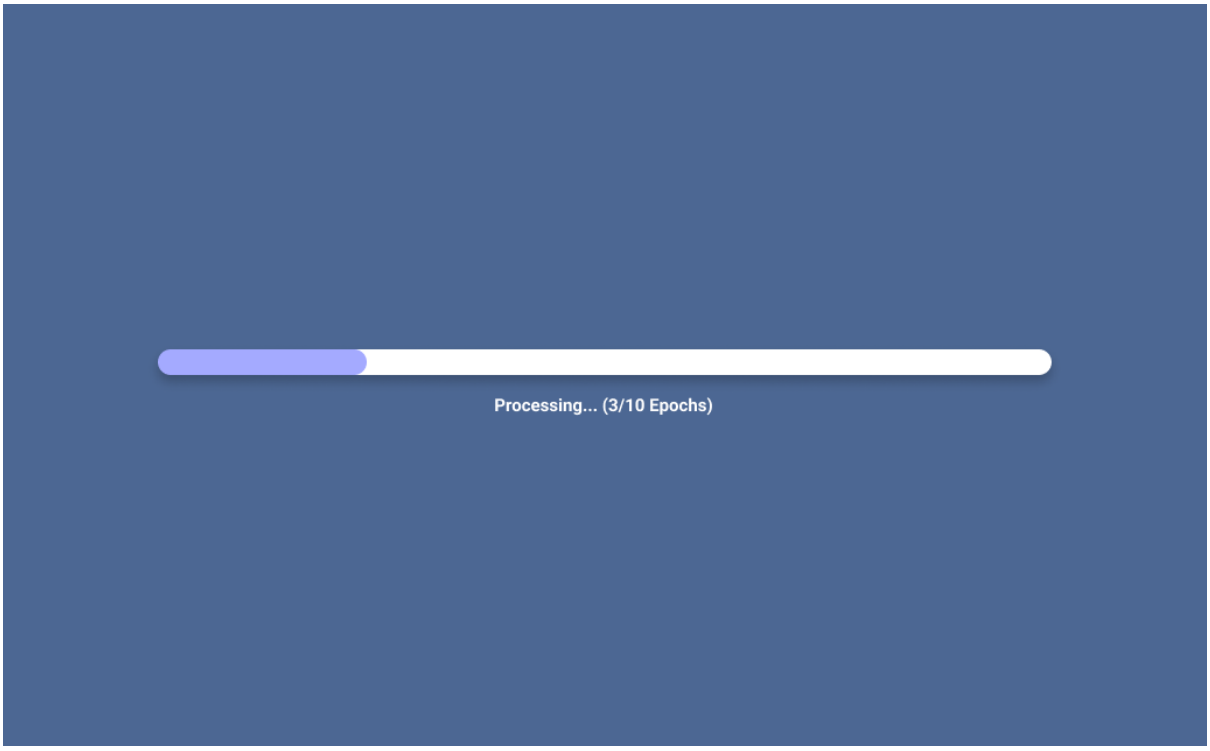
APPENDIX A – Interview Questions

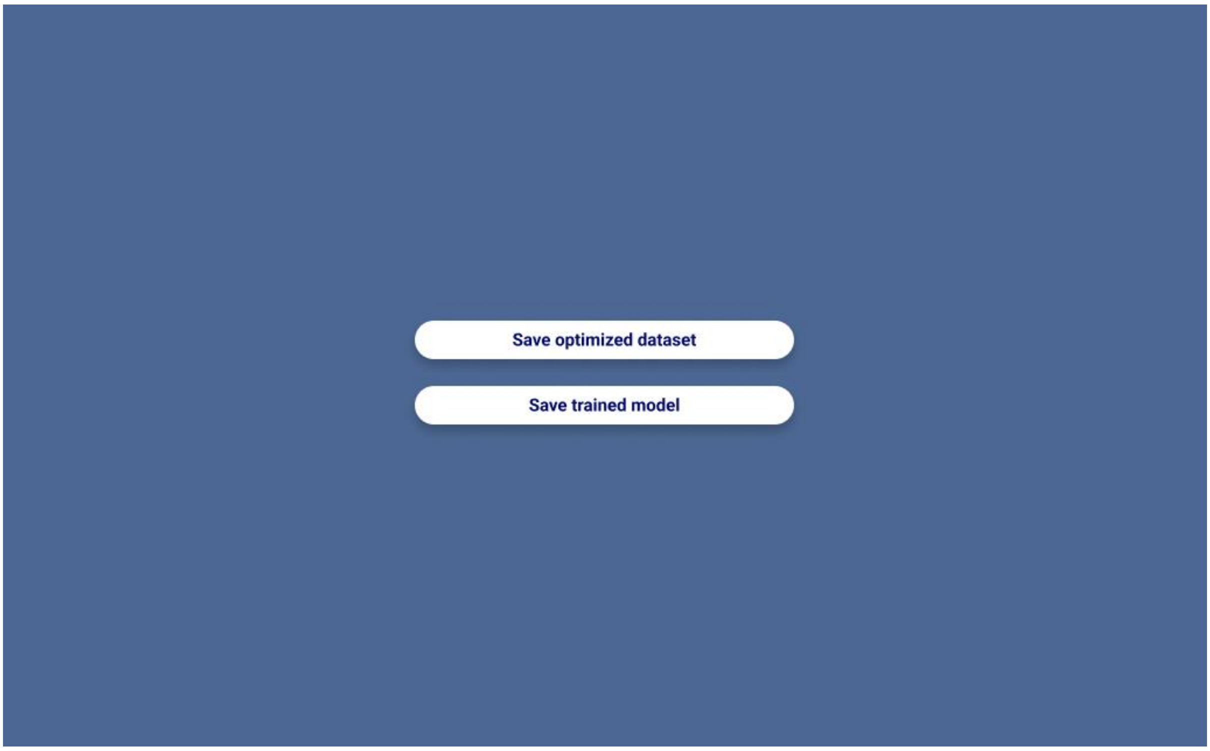
1. Have you pushed back from implementing artificial intelligence solutions due to the dataset limitations?
2. What are the most used image data augmentation techniques in your projects?
3. What challenges have you faced while selecting the image data augmentation methods for your dataset and task?
4. What potential do you see in automated data augmentation techniques?
5. Have you used automated data augmentation techniques in your projects? If yes, what challenges have you faced while using automated data augmentation techniques?
6. What would you expect from such a system apart from the ability to select best-performing data augmentation techniques? (i.e., add or remove specific data augmentation techniques from the system)
7. Most existing works performed reinforcement learning-based searches to find the best-performing data augmentation techniques. However, this reinforcement learning-based search is time-consuming. Any suggestions on other methods to solve the search efficiency issues?
8. Are you aware of any efficient methods to compare the original and the augmented dataset other than comparing density differences?
9. Any other suggestions to improve the system?
10. Would you be interested in being one of my evaluators?

APPENDIX B – Interview Findings

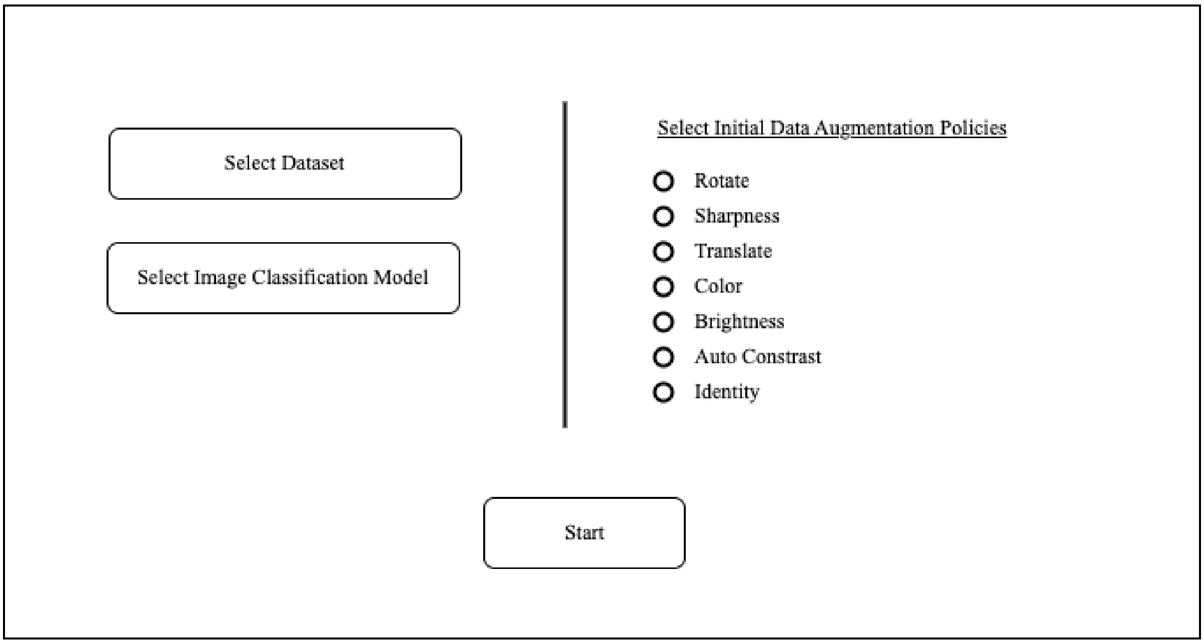
Interview Question	Evidence
2	
3	Obstructing relevant information that could be necessary for generating desired output. Rotation may result in unrealistic information. What specific augmentations would work the best for my model to perform better, Lots of time being spent in applying different augmentations, Performance issues when model being combined with augmentations. Mostly deciding on which technique could eventually give me the best results in the quickest possible time. This also has been the reason to not explore Neural Network based approaches to Data Augmentation. Selecting data augmentation techniques is an trial and error process which needs lot of experiments. Hence, project cost can be high since it requires lot of dev hours. When doing data augmentation need to consider the balance between model overfitting and underfitting. (No proper balance between that can occur model underfitting). Have to validate for data bias after DA, if necessary will have to implement some algorithms to correct data bias.
4	It is an interesting concept and certainly it would help researchers to focus more on the domain they are working and worry less on experimenting on augmenting data for their model's use case, to improve its performance. This should allow me to focus more on the final training aspect rather than spending too much time on carrying out augmentation manually. I am assuming the tool would perform multiple runs for different possible combinations and provide me with the best combination that I can use. It is a good concept, if it can provide a ranking of the concepts used and a score, it would give insight on how to develop models.

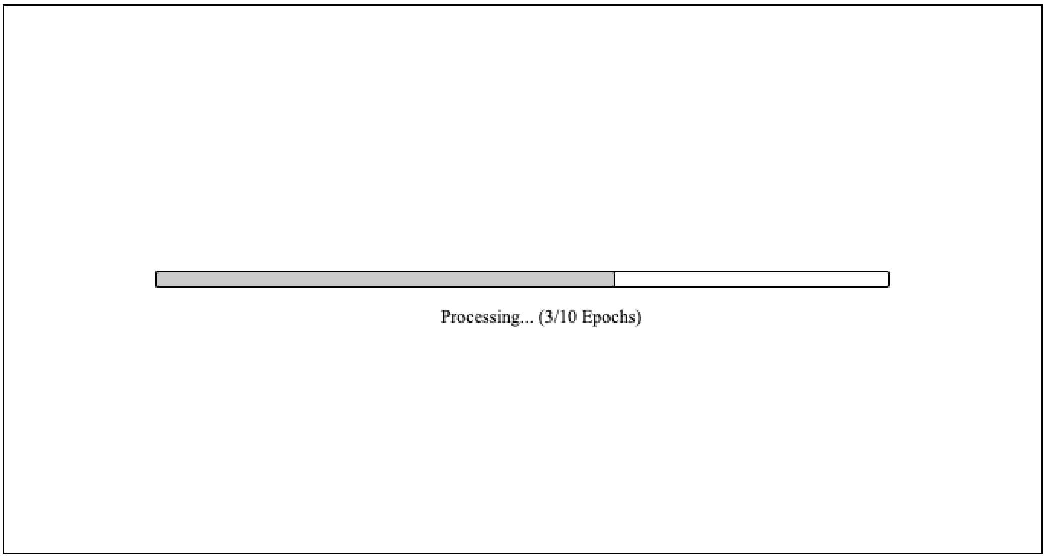
APPENDIX C – High Fidelity UI Designs

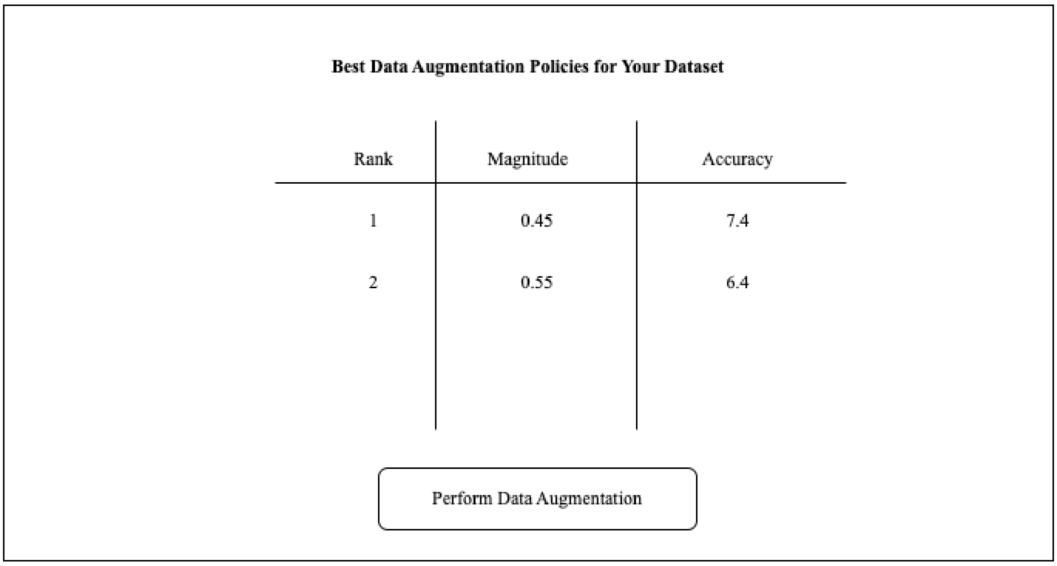


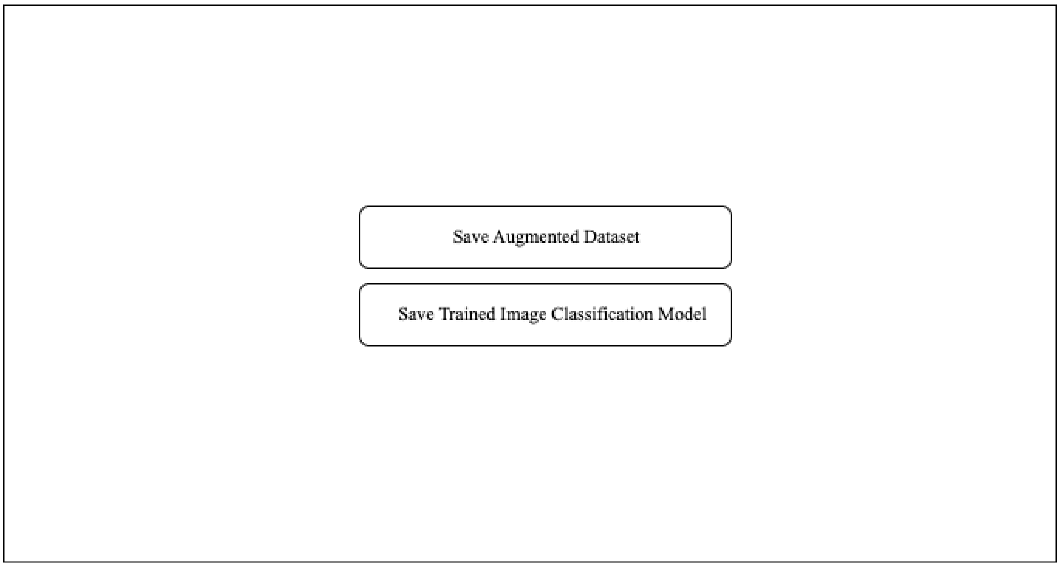

Save optimized dataset
Save trained model

APPENDIX D – Low Fidelity UI Wireframes




Processing... (3/10 Epochs)


Best Data Augmentation Policies for Your Dataset
Rank
Magnitude
Accuracy
1
0.45
7.4
2
0.55
6.4
Perform Data Augmentation


Save Augmented Dataset
Save Trained Image Classification Model