

# ECE496: Final Report

## Image Data Regeneration using Machine Learning

Gokul Kaushik | Rifdhan Nazeer | Zi Chien Chew | Henry Chen

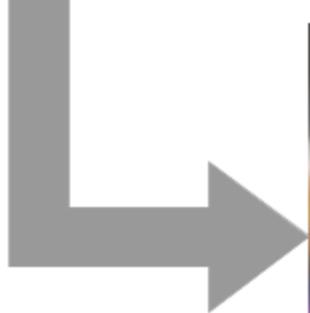
---

Project #: 672

Supervisor: Professor Jason Anderson

Administrator: Nick Burgwin

Submitted: March 22, 2018



## Group Final Report Attribution Table

This table should be filled out to accurately reflect who contributed to each section of the report and what they contributed. Provide a **column** for each student, a **row** for each major section of the report, and the appropriate codes (e.g. 'RD, MR') in each of the necessary **cells** in the table. You may expand the table, inserting rows as needed, but you should not require more than two pages. The original completed and signed form must be included in the hardcopies of the final report. Please make a copy of it for your own reference.

Section	Student Names			
	Gokul	Rifdhan	Zi Chien	Henry
Executive Summary, Group Highlights + Cover	RD, MR	RD, MR	ET	
1.0 Introduction	MR	MR	ET	RD
2.0 Technical Design	MR	RD, MR		ET
3.0 Work Plan	MR	MR	RD, ET	ET
4.0 Testing and Verification	RD, MR	MR	ET	
5.0 Conclusion	RD, MR	MR		ET
Appendices A-C	MR, ET	RD, MR	ET	
Appendices D-E	RD, MR	MR	ET	ET
Appendices F-K	MR	RD, MR	RD, ET	RD, ET
All	CM, FP	CM, FP	FP	FP

### Abbreviation Codes:

Fill in abbreviations for roles for each of the required content elements. You do not have to fill in every cell. The "All" row refers to the complete report and should indicate who was responsible for the final compilation and final read through of the completed document.

RS – responsible for research of information

RD – wrote the first draft

MR – responsible for major revision

ET – edited for grammar, spelling, and expression

OR – other

"All" row abbreviations:

FP – final read through of complete document for flow and consistency

CM – responsible for compiling the elements into the complete document

OR - other

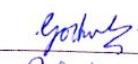
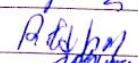
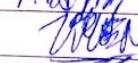
If you put OR (other) in a cell please put it in as OR1, OR2, etc. Explain briefly below the role referred to:

OR1: enter brief description here

OR2: enter brief description here

### Signatures

By signing below, you verify that you have read the attribution table and agree that it accurately reflects your contribution to this document.

Name	Gokul Kumar Kaushik	Signature		Date: Mar 19 2018
Name	Rifdhan Nazar	Signature		Date: Mar 19 2018
Name	Zi Chien Chew	Signature		Date: Mar 19 2018
Name	Hengyue Chen	Signature		Date: Mar 19 2018

## Voluntary Document Release Consent Form<sup>1</sup>

To all ECE496 students:

To better help future students, we would like to provide examples that are drawn from excerpts of past student reports. The examples will be used to illustrate general communication principles as well as how the document guidelines can be applied to a variety of categories of design projects (e.g. electronics, computer, software, networking, research).

Any material chosen for the examples will be altered so that all names are removed. In addition, where possible, much of the technical details will also be removed so that the structure or presentation style are highlighted rather than the original technical content. These examples will be made available to students on the course website, and in general may be accessible by the public. The original reports will not be released but will be accessible only to the course instructors and administrative staff.

Participation is completely voluntary and students may refuse to participate or may withdraw their permission at any time. Reports will only be used with the signed consent of all team members. Participating will have no influence on the grading of your work and there is no penalty for not taking part.

If your group agrees to take part, please have all members sign the bottom of this form. The original completed and signed form should be included in the hardcopies of the final report.

Sincerely,  
 Khoman Phang  
 Phil Anderson  
 ECE496Y Course Coordinators

### Consent Statement

We verify that we have read the above letter and are giving permission for the ECE496 course coordinator to use our reports as outlined above.

Team #: 672

Project Title: Image Data Registration Using Machine Learning

Supervisor: Jason Anderson

Administrator: Nick Burgwin

Name	<u>Gokul Kumar Kaushik</u>	Signature	<u>Gokul</u>	Date: Mar 19 2018
Name	<u>Rifshan Nazeer</u>	Signature	<u>Rifshan</u>	Date: Mar 19 2018
Name	<u>Zi Chien Chew</u>	Signature	<u>Zi Chien Chew</u>	Date: Mar 19 2018
Name	<u>Hengyue Chen</u>	Signature	<u>Hengyue Chen</u>	Date: Mar 19 2018

<sup>1</sup> This form will be detached from the hardcopy of the final report. Please make sure you have nothing printed on the back page.  
*last updated: March 2018*

# Executive Summary

Author: Rifdhan

Our project aims to reverse the effects of JPEG image compression using Machine Learning. It explores the effectiveness of Neural Network models for regenerating data lost through JPEG compression. This procedure is referred to as Image Regeneration.

We were able to evaluate two Machine Learning models, a Generative Adversarial Network (GAN) and a Convolutional Neural Network (CNN). Of the two, the GAN exhibited better performance, and was selected as the final model, upon which additional tests were performed. Overall, the model is able to regenerate image data lost through JPEG compression, as outlined in the Project Goal. The GAN is also able to satisfactorily meet all the Functions, Objectives, and Constraints in the Project Requirements, except for one Objective. Furthermore, when viewing the Output Images, the subjective performance of the model is much more impressive than the numbers would suggest.

While we have learned that Machine Learning can indeed regenerate image data lost through JPEG compression, some of the performance characteristics we have observed surprised us. For example, when evaluating performance with Flower images, a model trained on Faces outperforms a model trained on Flowers. Another example: a model trained on 1% JPEG compressed images (very low quality) consistently degrades the image quality of 10% JPEG compressed images (higher quality). The nature of SSIM as a metric has been explored as well, and its limitations have been identified. Furthermore, the effects of JPEG compression on other image properties have been studied.

The project is two weeks behind the original schedule outlined in the Project Proposal, but some tasks have been modified since then. Initial delays were mainly due to underestimating the impact of conflicting deadlines from other courses. Additionally, we have spent more time than anticipated on generating and analyzing results from our models; there is an endless stream of experiments to try. We also encountered difficulties with the ECF compute environment which we use for our work.

The next steps in the project mainly relate to the Design Fair on April 4th. We aim to set up a compelling interactive demonstration, where users take photos, view the effects of low-quality JPEG compression on them, and then feed the compressed Input Images into our Machine Learning model and view the resulting Output Images. We are actively working on building a system to enable this experience. Despite the wide variety of results we have so far, we have only scratched the surface of what this technology is capable of. The GAN model will continue to be tested under new conditions, in order to learn as much as possible prior to the conclusion of the term.

# Group Highlights and Individual Contributions

Author: Gokul

This section summarizes the final accomplishments of the group, and notes the key contributions from each team member to the project as a whole.

## Group Highlights

Our team is pleased to state that we were able to achieve the project goal: recover some of the data lost in image compression using Machine Learning techniques. Furthermore, we were able to discover interesting relationships between image compression, image quality, and file size. Highlighted below are some of our key accomplishments and results. Please refer to the glossary in Appendix K for definitions of technical terms.

1. **Accomplishment:** Met all the project's Functions and Constraints, and all but one Objective (please refer to Section 4.2).
2. **Accomplishment:** We were able to stay close to schedule and achieve all major milestones set out in the Gantt Chart (refer to Section 3.2). The project is approximately 2 weeks behind schedule, which is a reasonable degree of planning error for a project of this length. We explored a number of Machine Learning models, selected two models (the Generative Adversarial Network (GAN) and the Convolutional Neural Network (CNN)), and tested these them extensively. Finally, the GAN was chosen as the final model, and detailed experiments were performed with it.
3. **Accomplishment:** We were able to sufficiently explore the capabilities of the model by modifying its constituent parameters, varying the training and testing dataset categories, and the varying the quality levels of the images in the dataset. For the complete list of variations explored, please refer to Appendices F and G.
4. **Result:** We discovered many interesting and in some cases, counter-intuitive relationships from the training and testing of the various datasets and parameter tunings. For example, a model trained using the Faces dataset exclusively, was able to regenerate Flower images better than it was able to regenerate Face images. There are many more results and conclusions available in Appendices F and G.
5. **Result:** We explored the effects of JPEG Compression on file size and SSIM values. While higher SSIM values, on average, mean higher image quality, we found that it is a noisy metric. Moreover, the relationship between compression level and file size was observed to be non-linear. For more details, please refer to Appendix D.

# Individual Contributions

There are 4 members in our team. Key contributions for each member over the course of the project are listed below.

## Gokul Kaushik

- CNN 1 Model: Setup and configured the first model to generate an output. However, the model did not perform its intended task correctly. Due to the seeming early success, invested more time and effort attempting to correct the behaviour. Eventually abandoned the model to work on other tasks when other team members' models emerged as more viable solutions.
- Dataset: Obtained and organized the MIRFLICKR-1M dataset (the primary dataset used in the project). Also obtained the Manga 109 dataset (See Appendix E).
- Team Leader: Organized meetings with the supervisor and internal team meetings. Assigned tasks and monitored in reference to the Gantt Chart. Initiated discussions on challenges and finalizing proposed solutions. Communicated with the relevant authorities for project resources.
- Data Analysis: Did initial work on evaluating the performance of the two main models (the GAN and the CNN), leading the way to more detailed experiments on them. Later, gathered results from the two individual models, and did a comparative analysis on their respective performance.
- Metric Evaluation: Examined the effectiveness of SSIM as a metric in the context of our project. Explored the relationship between JPEG compression, SSIM Score, and other attributes. See Appendix D for more details. Also explored the nature of JPEG compression itself, with regards to image quality and file size.

## Rifdhan Nazeer

- GAN Model: One of the two working models, and the model eventually selected at the end of the project. Responsible for setting up and modifying the existing code to perform its intended task of superresolution. Later, transformed the model to perform Image Regeneration. Responsible for training and testing the model on various datasets and parameters.
- Data Analysis: Generated data on the performance of the GAN model in a variety of conditions. Explored modifying the model, training/testing on various image categories and JPEG compression levels, and cross-testing on different datasets than what the model was trained on. Produced extensive results using this accumulated data (see Appendix F).

- Technical Expertise: Being the most experienced programmer in the team, provided assistance to other members with any technical difficulties. Wrote the key ReadMes and guides pertaining to understanding, setting up, and utilizing tools and software. Worked with Tim Trant to ensure the technical needs of the team were being met, and to troubleshoot various issues we encountered over the course of the project (described in Section 3.3).

### **Zi Chien Chew**

- CNN 2 Model: One of the two working models. Responsible for setting up the model to perform its intended task of image denoising. Later, transformed the model to perform Image Regeneration. Learned the libraries and tools necessary to achieve this.
- Data Generation: Responsible training and testing the CNN 2 model on various datasets and conditions. Generated data on the performance of the model under these circumstances, and stored the data in an appropriate format for further evaluation (see Appendix G).
- Modification of Model: Explored modifying the ML model, and the related performance effects this had. Initial results indicated the changes in performance were not worth the tradeoffs encountered with these modifications, so the investigation was concluded early.

### **Henry Chen**

- Colorization Model: One of the three models that performed its intended function. Setup and configured the model to perform image colorization. Investigated modifying the model to perform Image Regeneration, but eventually abandoned the project and moved to work on CNN 2 model (with Zi Chien), as it showed greater promise.
- Pixiv Dataset: Wrote web crawler script to download cartoon images from pixiv.net [9] to assist with training and testing the CNN 2 model. This model was designed for use with cartoon images (which were not available in other existing datasets), so a specialized dataset was required. However, this dataset did not end up being used much, as we required the use of a standardized data sets between the two models in order to be able to compare their performance fairly. The Manga 109 dataset eventually replaced it.
- Demo for Design Fair: Working on an application to demonstrate a practical use case for the research. The application will take photos of subjects, compress the images into a low JPEG compression level, and regenerate the some of the data lost using our ML model. The images at these three stages will be displayed side-by-side.

## Acknowledgements

We would like to express our gratitude to our supervisor, Professor Jason Anderson. His support and insights during our weekly meetings helped us through the most difficult parts of the project.

We would also like to thank Mr. Tim Trant for the assistance he provided with regards to the ECF compute resources. The project would not have been possible without these machines.

Lastly, we would like to acknowledge the prior work of GitHub users Hi-king and Nagadomi. Their Machine Learning models (designed to perform image super-resolution and denoising respectively) were modified for use in our project. Without their initial work to serve as a starting point, it would have been significantly more difficult to achieve the same quantity of results within the given timeframe.

# Table of Contents

<b>1.0 Introduction</b>	<b>1</b>
1.1 Background and Motivation	1
1.2 Project Goal	1
1.3 Project Requirements	2
<b>2.0 Technical Design</b>	<b>3</b>
2.1 Measuring Image Similarity	3
2.2 Machine Learning Methodology	3
2.3 System Level Overview	5
2.4 Module Level Descriptions	6
2.4.1 Preprocessing Stage	6
2.4.2 Training Stage	6
2.4.3 Testing Stage	6
2.5 Overview of GAN Model	7
2.6 Assessment of Final Design	9
<b>3.0 Work Plan</b>	<b>10</b>
3.1 Work Breakdown Structure	10
3.2 Final Gantt Chart	13
3.3 Challenges Faced	15
<b>4.0 Testing and Verification</b>	<b>17</b>
4.1 Verification Matrix	17
4.1.1 Test: Image Data Regeneration	18
4.1.2 Test: Timing Model Performance	18
4.1.3 Test: Maximizing Regeneration	19
4.1.4 Test: Variance Testing	19
4.1.5 Test: Testing for Overfit	19
4.2 Module-Level Test Results	19
4.2.1 Result: Output Image Generation	19
4.2.2 Result: Image Data Regeneration	19
4.2.3 Result: JPEG Image Format	21
4.2.4 Result: Timing Model Performance	21
4.2.5 Result: Maximizing Regeneration	21
4.2.6 Result: Variance Testing	22
4.2.7 Result: Uses a Machine Learning Model	23
4.2.8 Result: ECF Workstation Environment	23
4.2.9 Result: Public Datasets	23
4.2.10 Result: Existing Machine Learning Libraries	23
4.2.11 Result: Testing for Overfit	24
<b>5.0 Conclusion</b>	<b>25</b>
<b>6.0 References</b>	<b>26</b>
<b>Appendix A: Gantt Charts History</b>	<b>28</b>
A.1 Original Gantt Chart - Project Proposal	28
A.2 Updated Gantt Chart - Progress Report	29

<b>Appendix B: Original Validation Tests</b>	31
B.1 SSIM Measurement Methodology	31
B.2 Testing for Overfit	31
<b>Appendix C: Existing Image Quality Metrics</b>	32
C.1 Mean Square Error (MSE)	32
C.2 Structural Similarity (SSIM)	32
<b>Appendix D: Metric Evaluation</b>	34
D.1 The Ideal Image Storage Format	34
D.1.1 Image Quality	35
D.1.2 File Size	35
D.1.3 Viewing Speed	35
D.2 The Effects of JPEG Compression on Images	36
D.2.1 Effects on Manga Images	36
D.2.2 File Size Effects	36
D.2.3 Image Quality Effects	37
D.2.4 Non-Linearity of SSIM Value with JPEG Compression Level	38
D.2.5 Samples of JPEG Compressed Images	39
D.3 Shortcomings of SSIM	41
<b>Appendix E: Image Categories and Datasets</b>	43
E.1 Image Categories	43
E.2 Dataset Pollution	44
<b>Appendix F: Results from GAN Model</b>	46
F.1 Performance on All Datasets	46
F.2 Performance when Crossing Datasets, by Training	48
F.3 Performance when Crossing Datasets, by Testing	50
F.4 Performance when Modifying the Model	53
F.5 Performance at Various JPEG Compression Levels	54
F.6 Performance on Training versus Testing Data	56
F.7 Samples with Good Performance	57
F.8 Samples with Poor Performance	62
<b>Appendix G: Results from CNN Model</b>	67
<b>Appendix H: Comparing Results (GAN vs CNN)</b>	68
<b>Appendix J: Existing Regeneration Technology</b>	70
<b>Appendix K: Glossary</b>	73

# 1.0 Introduction

Author: Henry

This report describes our capstone project for the ECE496 final year design course. It discusses the motivation behind the research project, and documents the final design implementation and performance with respect to the updated Project Requirements from the Progress Report. This section briefly introduces the project, and provides explanations of key concepts.

## 1.1 Background and Motivation

Data compression is the process of representing data using fewer bits [1]. It has become increasingly popular on the internet. In the past 5 years alone, the fraction of websites using some sort of data compression rose by 30% [2].

Compression can be either lossless or lossy. Existing lossy compression technologies achieve low file sizes at the expense of image quality. However, the compression techniques used cannot recover the lost data [3]. The most popular lossy compression image format used on the internet is JPEG [4]. Figure 1 below shows the drawbacks of lossy compression. An analysis of the performance of existing popular applications for improving image quality is presented in Appendix J.

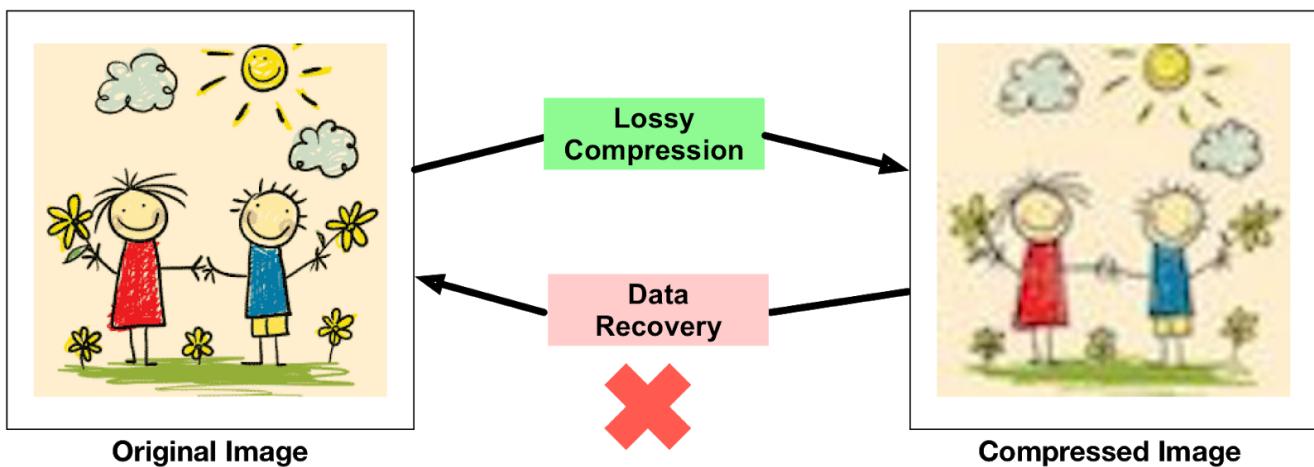


Figure 1 - The problem: Lossy compression incapable of data recovery (adapted from [5])

## 1.2 Project Goal

The goal of this project is to explore how well Machine Learning techniques can regenerate data lost through JPEG image compression. This process will henceforth be referred to as Image Regeneration.

## 1.3 Project Requirements

Presented in Table 1 below are the project's final functions, objectives, and constraints.

**Table 1 - Functions, Objectives, and Constraints**

ID	Description
1	<b>Primary Functional Requirement:</b> The system must generate an output image.
2	<b>Functional Requirement:</b> The SSIM Score must be strictly positive (i.e. SSIM 2 - SSIM 1 > 0). This means that the Output Image must have a greater SSIM score than the Input Image with reference to the Target Image for any JPEG compression level (between 0% and 100%).
3	<b>Sub Functional Requirement:</b> The system should be able to accept input images in the JPEG format (as it is the most popular lossy image format on the internet [4]).
4	<b>Objective:</b> Minimize computation time. The design should produce the Output Image in under 20 seconds on the ECF workstations.
5	<b>Objective:</b> Minimize the JPEG compression level. The design should achieve an SSIM Score of at least 0.05 with Input Images of 10% JPEG compression or lower. This implies that SSIM 2 is greater than SSIM 1 by at least 0.05.
7	<b>Objective:</b> Performance should generalize across varied image content. Using a test set of 50 random Input Images (from a dataset with varied image content), the difference between the highest and lowest SSIM Scores among all the test images should be less than 0.2.
8	<b>Constraint:</b> The Image Regeneration should be done using an ML model.
9	<b>Constraint:</b> The model should run on standard computing hardware (x86 processor architecture and NVIDIA GPUs, as present in ECF workstations).
10	<b>Constraint:</b> The system must use publicly-available datasets.
11	<b>Constraint:</b> System should use an existing ML library.
12	<b>Constraint:</b> ML model should not overfit on training data (should be able to regenerate data in previously-unseen images). The difference between training and testing SSIM Scores should be less than 0.3.

## 2.0 Technical Design

Author: Rifdhan

This section introduces some technical concepts, then details the final design. For most of the project, we worked with two different ML models: a Generative Adversarial Network (GAN), and a Convolutional Neural Network (CNN). Towards the end of the project, we selected the GAN as the final model, due to its lower resource requirements, and ease to work with. It also exhibited better performance in most of the comparative tests performed between the two models (see Appendix H).

### 2.1 Measuring Image Similarity

Our project aims to take a compressed image and make it “better”. However, measuring “better” is a challenge, and how we define it greatly impacts how our ML Models are evaluated. For our project, “better” means more similar to the Original Image. Structural Similarity (abbreviated SSIM) is the metric chosen to measure similarity between two images. Refer to Appendix C for a discussion on SSIM and other competing metrics. SSIM was designed to approximate how humans perceive similarity in images [8]. Its values range from 0 to 1, with a higher value indicating greater similarity. An SSIM value of 1 indicates that the two compared images are identical, and a value of 0 indicates no similarity. During the project, we discovered a few shortcomings of SSIM (see Appendix D).

### 2.2 Machine Learning Methodology

To provide a foundation for the technical design, we will first introduce some basic terminology. There are 3 main stages in the ML process (as depicted in the Figure 2 below).

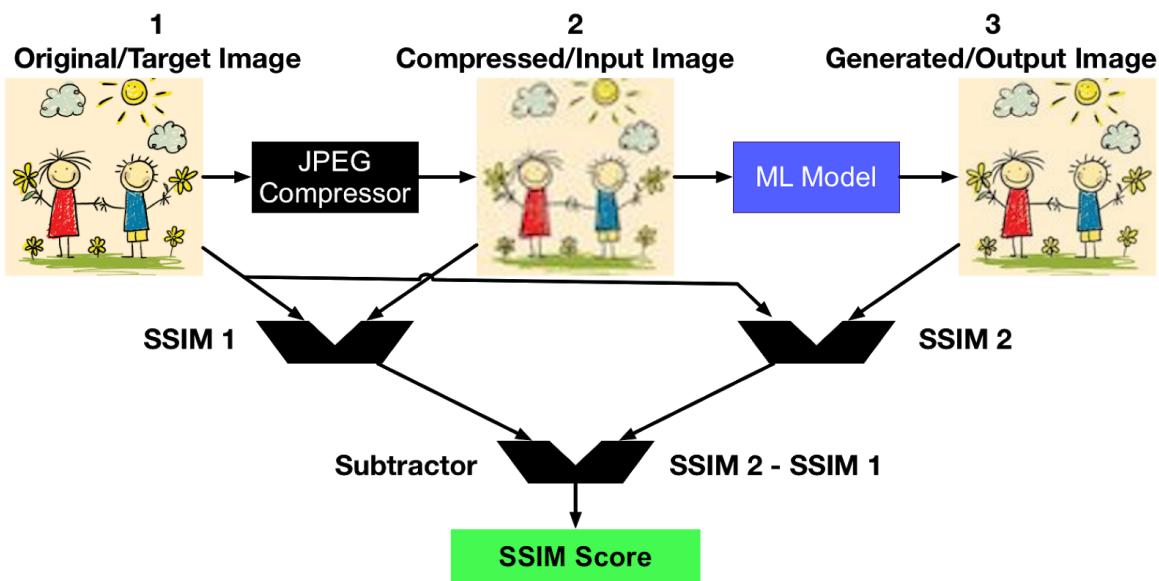


Figure 2 - A block diagram of the regeneration process and scoring methodology

Steps in evaluation process:

1. **Original or Target Image:** This is the original image at full quality without any pre-processing.
2. **Compressed or Input Image:** JPEG compressed version of the Original Image, at some quality level measured in percent.
3. **Generated or Output Image:** The image generated by the machine learning model.

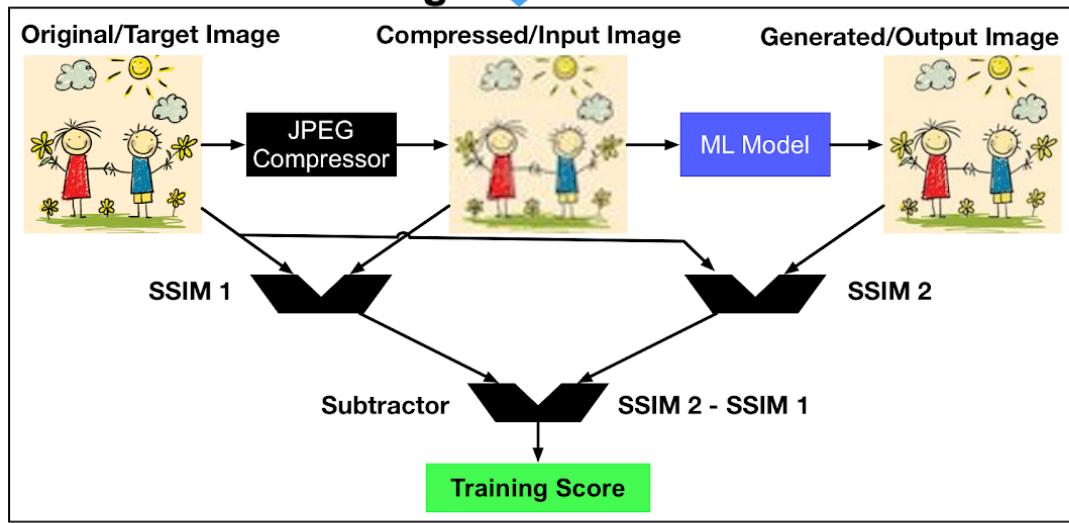
In order to measure a model's performance, we need to compare both the Input and Output Images with the Target Image respectively. An SSIM score is generated for the two pairs (SSIM 1 = Input and Target Image, SSIM 2 = Output and Target Image). SSIM 1 indicates how similar the Compressed Image is to the Original Image. It acts as a baseline upon which the model must improve. SSIM 2 measures how similar the Generated Image is to the Original Image. If SSIM 2 is greater than SSIM 1, then the model has successfully regenerated some of the data lost in JPEG compression. Note that if SSIM 2 is somehow less than SSIM 1, then the model has degraded the image further, and is obviously not successful. Refer to Section 1.3 for specifics on SSIM scores that we aim to achieve.

## 2.3 System-Level Overview

### 1. Image Pre-Processing



### 2. ML Model Training



### 3. ML Model Testing

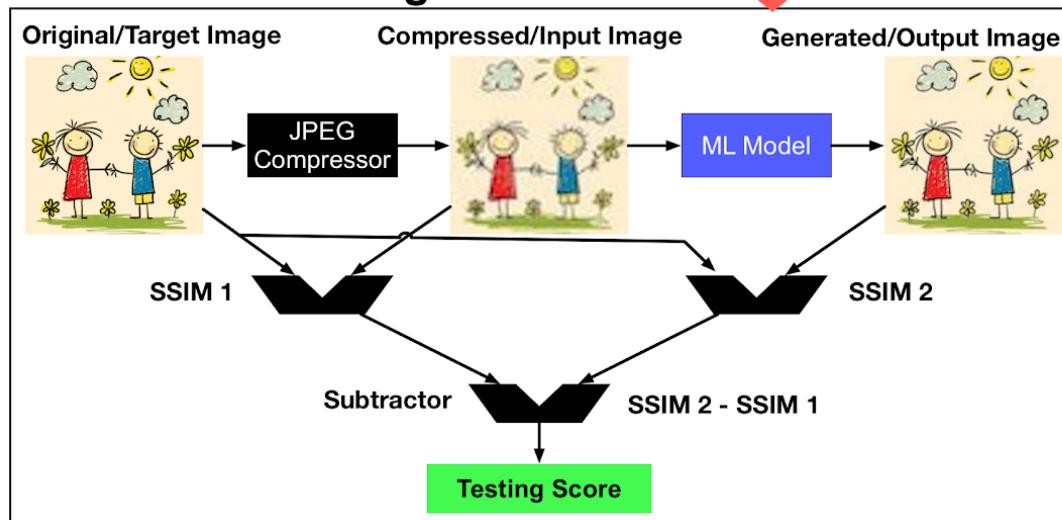


Figure 3 - System-Level Overview, consisting of 3 Stages, discussed in Section 2.4

## **2.4 Module-Level Descriptions**

The Machine Learning model consists of 3 main stages: Preprocessing, Training, and Testing, as seen in Figure 3 above. These stages are described in more detail in this section.

### **2.4.1 Preprocessing Stage**

The first stage in the process is to prepare a dataset. The main dataset used in our project was the MIRFLICKR-1M dataset [7], consisting of one million images of various categories. This dataset was then sorted by image category, to permit training/testing on specific categories, such as Faces or Clouds. Afterwards, the images were scaled to a smaller resolution if necessary (some images were too large and caused memory problems on the ECF machines - see Section 3.3), and then JPEG-compressed to a number of preset compression levels (such as 10%, 5%, and 1%). The images in each category were then split into a training and testing set, consisting of 80% and 20% of the images in the category respectively. In some cases, where image categories contained a large number of images, the test set was reduced to less than 20%, as testing is time-consuming. This entire procedure was performed for all other datasets used in the project as well.

### **2.4.2 Training Stage**

The second stage in the process is the most time-consuming: training the ML model. In this stage, the training dataset is provided to the model, and it randomly selects images from this set to be Input Images. These images are used to generate Output Images, and the performance of the model is evaluated for each image, using a fast metric, such as Peak Signal-to-Noise Ratio (used in the CNN) or Softmax Cross-Entropy (used in the GAN). Based on these performance characteristics, the model's internal parameters are adjusted, and the process repeats with the next training image. At regular intervals during the training process, the model state is saved to persistent storage, marking its progress as it trains. By the end of the stage, we have a set of trained models, at incremental levels of training.

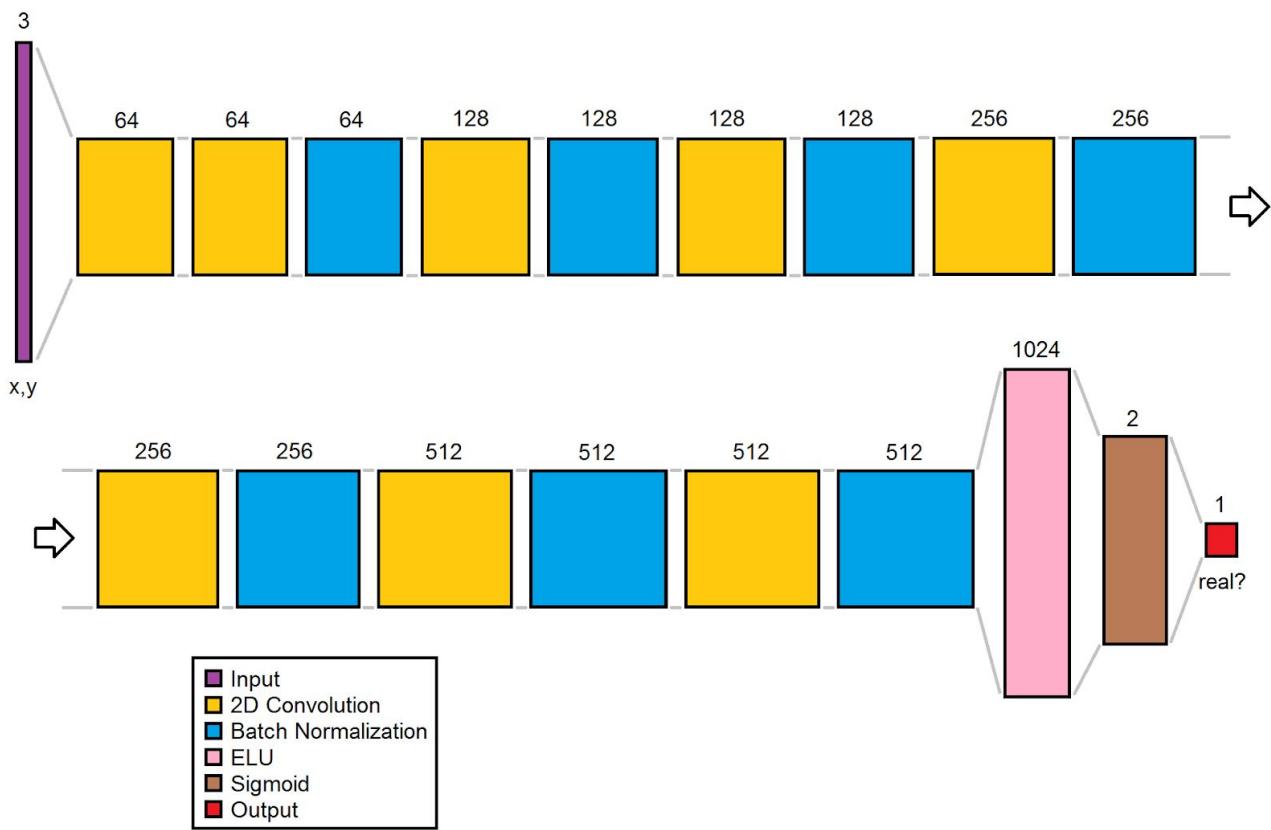
### **2.4.3 Testing Stage**

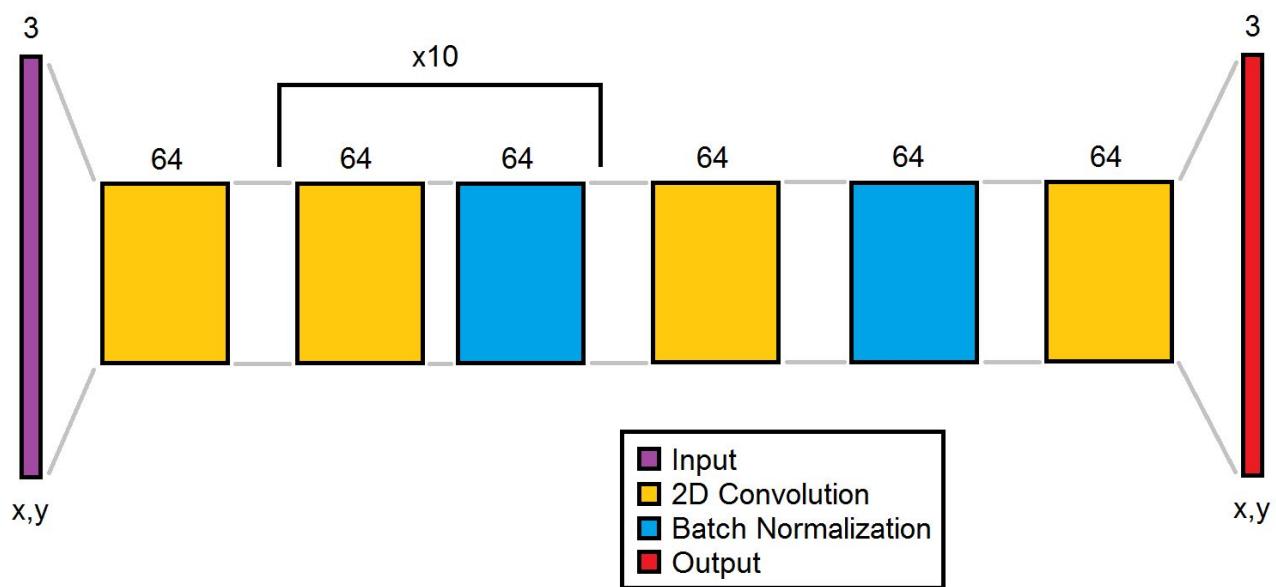
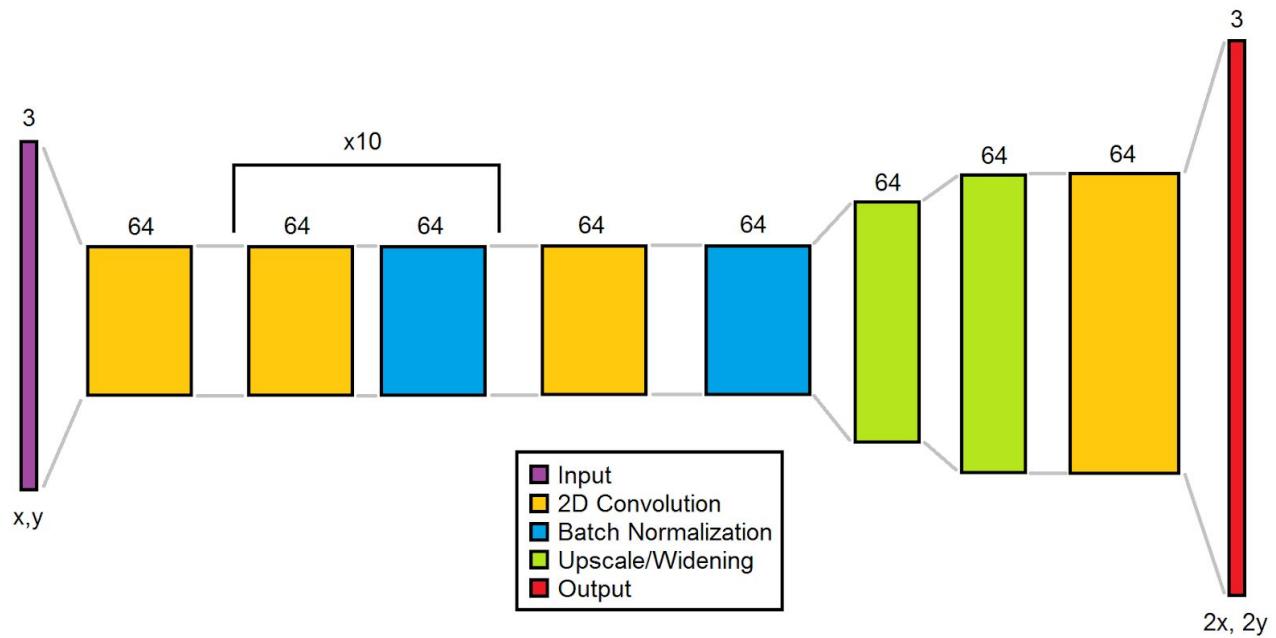
In the final stage of the design, we take the set of trained models, and evaluate the performance of the system with them. The set of test images are fed into the models as the Input Images, and the models generate Output Images for each. These images are then compared with the Original Images, and a variety of statistics are calculated, including SSIM Score, Mean-Square Error, and file size. Since we have a separate model for each incremental level of training, we can then graph these statistics versus the amount of training, to observe how the statistics change as the model trains on more

examples. This allows us to gain an understanding of how much training is appropriate, and at what levels of training the model underfits or overfits the data. Additionally, cross-testing between datasets was performed to evaluate how the model is able to adapt to the dataset it trained on. This involves testing the model on a different dataset than it was trained on.

## 2.5 Overview of GAN Model

The GAN model was the final model used in the project. It contains two Neural Network models within it: the Generator and Discriminator models. The Generator model was modified to convert its functionality from superresolution to Image Regeneration. A block-level diagram of both models, including the unmodified and modified versions of the Generator, are presented below. Note that these diagrams are simplified - blocks contain more parameters then shown here. The number indicated above each block indicates the number of output layers it produces.





## 2.6 Assessment of Final Design

We are pleased with the performance of our final design, the GAN model. We successfully evaluated two different Machine Learning models (a GAN and a CNN) on a multitude of training and testing datasets, and gained a good understanding of the strengths and limitations of using Machine Learning to solve the of Image Regeneration, and found the GAN was the better performer of the two (see Appendix H). The individual components of the design perform their intended functions without issue, and we were able to use the GAN to generate a wealth of results and statistics. The model was also able to meet all of our Functions and Constraints, and most of our Objectives, demonstrating its successfulness. Furthermore, as seen in the sample images in Figure 7 below, the regeneration ability of the GAN is quite visible just through observation of the results. Detailed performance statistics for the GAN model can be found in Appendix F.

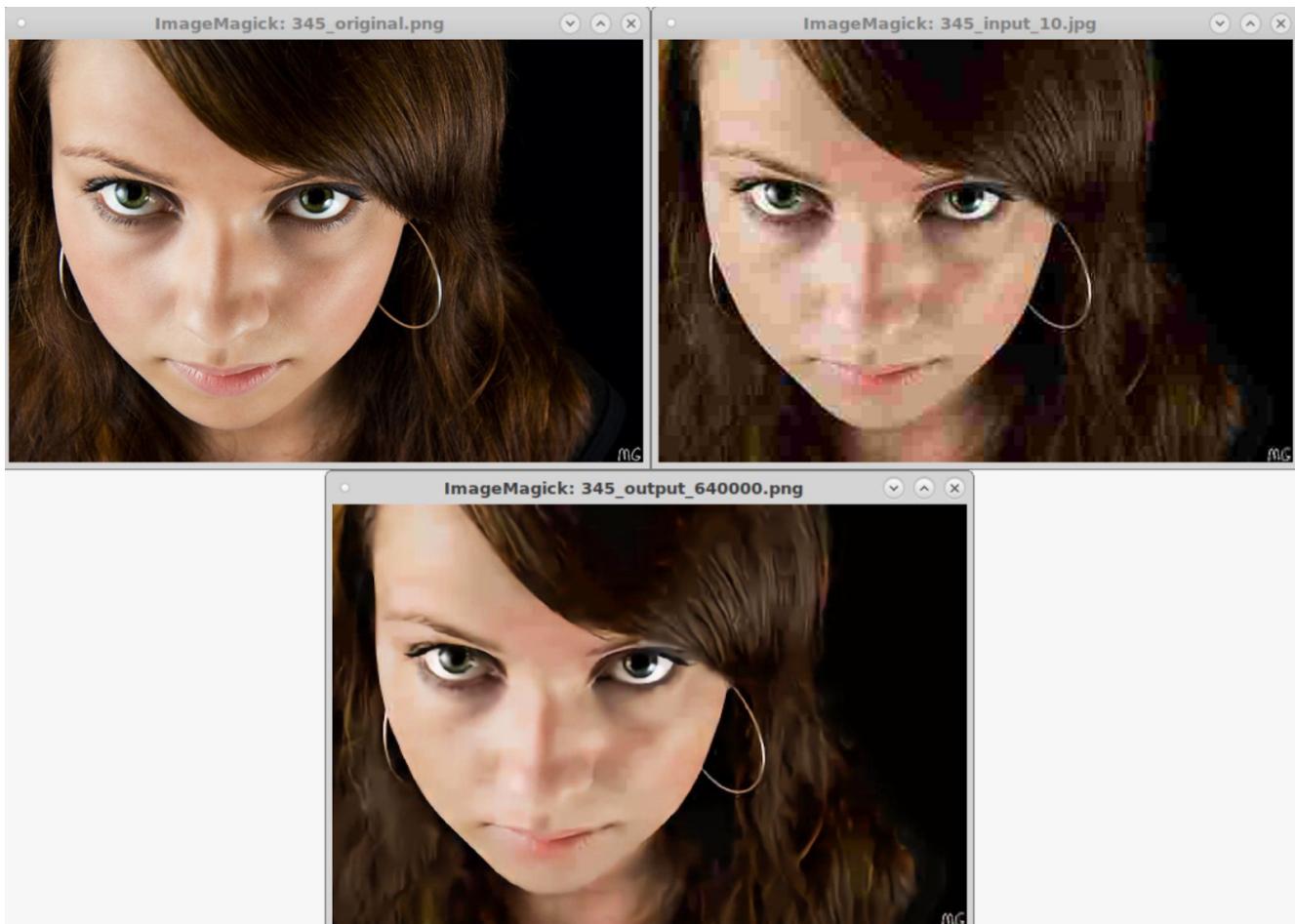


Figure 7 - Original (left), Input (right), and Output (bottom) Images for a 10% JPEG test image from the Faces dataset. Regeneration performed by the GAN model, trained on Faces at 10% JPEG.

## 3.0 Work Plan

Author: Zi Chien

This section outlines the progress of the project, measured through individual tasks and milestones. A discussion of problems encountered is presented as well. Overall, the project is approximately 2 weeks behind schedule, but we have adjusted the remaining tasks to allow us to complete the remaining work prior to the Design Fair.

### 3.1 Work Breakdown Structure

Presented in Table 2 below is an updated Work Breakdown Structure (WBS), corresponding to the updated Gantt Chart shown in the following section, Section 3.2. Note that this WBS represents the final set of tasks as completed during the project. We tried to avoid changing the number of any existing task, resulting in discontinuous numbering when tasks were added/removed/reordered later.

**Table 2 - Work Breakdown Structure**

Note: R = Responsible, A = Assisting

#	Task Description	Rifdhan	Gokul	Zi Chien	Henry
1	<b>Obtain ECF resources</b>	R			
2	<b>Select metric for measuring image similarity</b>	R			
3	<b>Explore image datasets</b>				
3.1	Explore cat dataset				R
3.2	Explore IMAGENet dataset		R		
3.3	Explore cartoon image dataset			R	
3.4	Explore MIRFLICKR-1M dataset		R		
5	<b>Find existing ML model that solve similar problems, learn libraries and tools used, set up locally and get working as-is</b>				
5.1	CNN 1 (Superresolution)		R		
5.2	Colorization				R
5.3	GAN (Superresolution)	R			
5.4	CNN 2 (Waifu2x)			R	
6	<b>Modify existing implementations to solve our problem</b>				
6.1	CNN 1		R		

6.2	Colorization				R
6.3	GAN	R			
6.4	CNN 2 (Waifu2x)			R	
7	<b>Select two most successful models and evaluate further with new data</b>				
7.1	GAN	R			
7.2	CNN 2 (Waifu2x)			R	
*7	<b>Organize additional datasets</b>				
*7.1	Generate additional training datasets from Flickr		R		
*7.2	Organize MIRFLICKR-1M dataset		R		
*7.3	Generate Pixiv image dataset for CNN 2				R
*7.4	Standardize dimensions and resolutions of Pixiv dataset				R
10	<b>Optimize implementation against objectives</b>				
10.1	Obtain GPU support	R			
10.3	Modify code to utilize GPUs and train models using GPUs	R		A	
10.2	Explore generalizing to wider selection of input images		R		A
11	<b>Train and test final models across a variety of datasets and compare performance</b>				
11.1	GAN	R			
11.2	CNN 2 (Waifu2x)			R	
8	<b>Select the most successful model and refine implementation: GAN</b>				
8.1	Improve model	R		A	
8.2	Automate testing on wide variety of training datasets	R		A	
9	<b>Verify design against test criteria</b>				

9.1	Make tool to measure SSIM and MSE errors		R		
9.2	Document statistics on trained models		R	A	
12	<b>Develop interactive system to demonstrate project at design fair</b>				
12.1	Frontend mobile app to take photos and display images				R
12.2	Backend server to interface with ML model				R

### 3.2 Final Gantt Chart

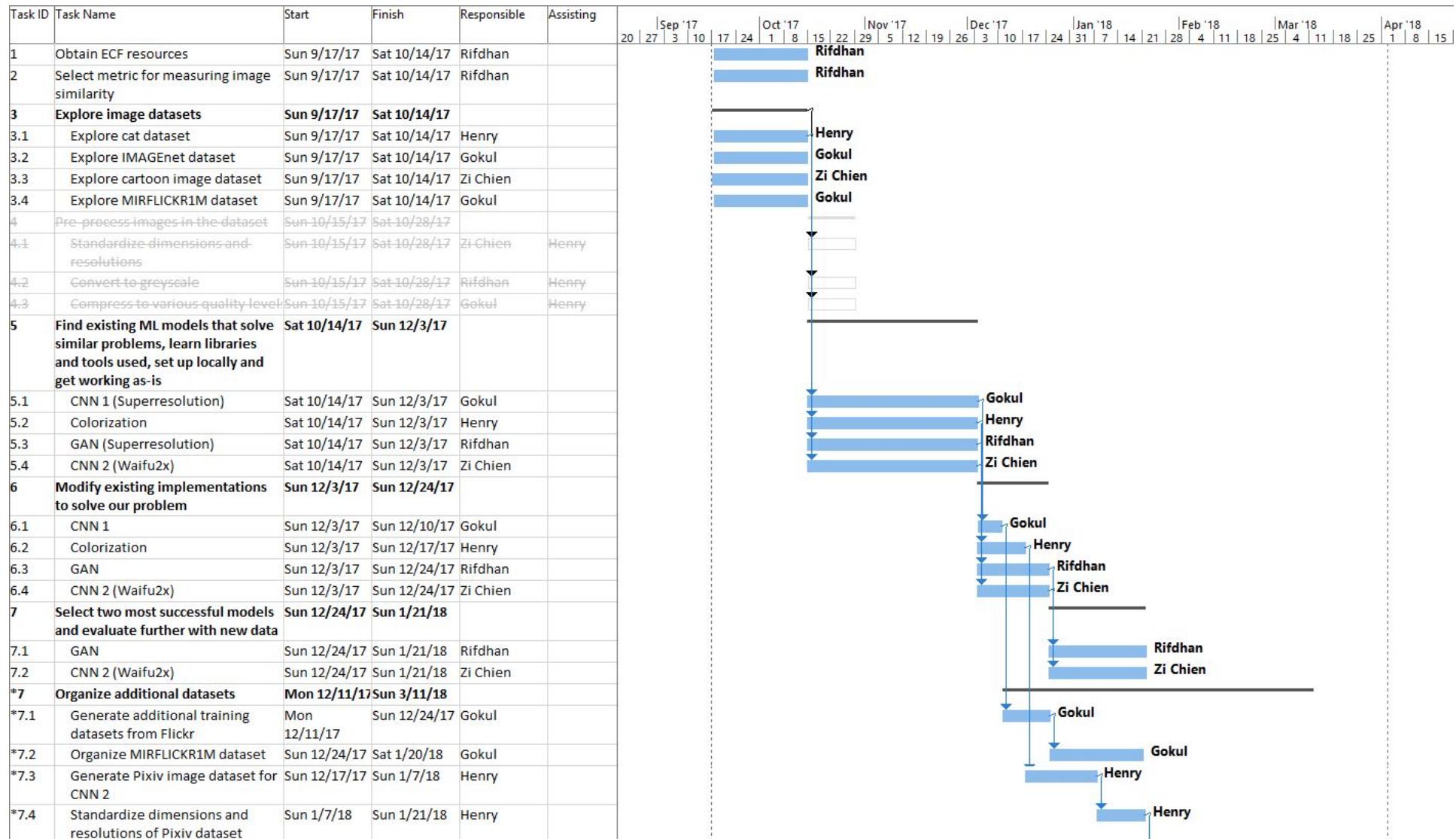


Figure 8 - Final Gantt Chart (part 1 of 2)

10	Optimize implementation against objectives	Sun 1/7/18	Sun 3/11/18		
10.1	Obtain GPU support	Sun 1/7/18	Sat 1/20/18	Rifdhan	
10.3	Modify code to utilize GPUs and train models using GPUs	Sat 1/20/18	Sun 2/18/18	Rifdhan	Zi Chien
10.2	Explore generalizing to wider selection of input images	Sun 2/18/18	Sun 3/11/18	Gokul	Henry
11	Train and test final models across a variety of datasets and compare performance	Sun 1/21/18	Sun 3/4/18		
11.1	GAN	Sun 1/21/18	Sun 3/4/18	Rifdhan	
11.2	CNN 2 (Waifu2x)	Sun 1/21/18	Sun 3/4/18	Zi Chien	
8	Select the most successful model and refine implementation	Sun 3/4/18	Mon 3/19/18		
8.1	Improve model	Sun 3/4/18	Mon 3/19/18	Rifdhan	Zi Chien
8.2	Explore new training and testing combinations and scenarios	Sun 3/4/18	Mon 3/19/18	Rifdhan	Zi Chien
9	Verify design against test criteria	Sun 3/4/18	Mon 3/26/18		
9.1	Make tool to measure SSIM and MSE errors	Sun 3/4/18	Sun 3/18/18	Gokul	
9.2	Document statistics on trained models	Sun 3/4/18	Mon 3/26/18	Gokul	Zi Chien
12	Develop interactive system to demonstrate project at design fair	Sun 3/4/18	Mon 4/2/18		
12.1	Frontend mobile app to take photos and display images	Sun 3/4/18	Mon 4/2/18	Henry	
12.2	Backend server to interface with ML model	Sun 3/4/18	Mon 4/2/18	Henry	

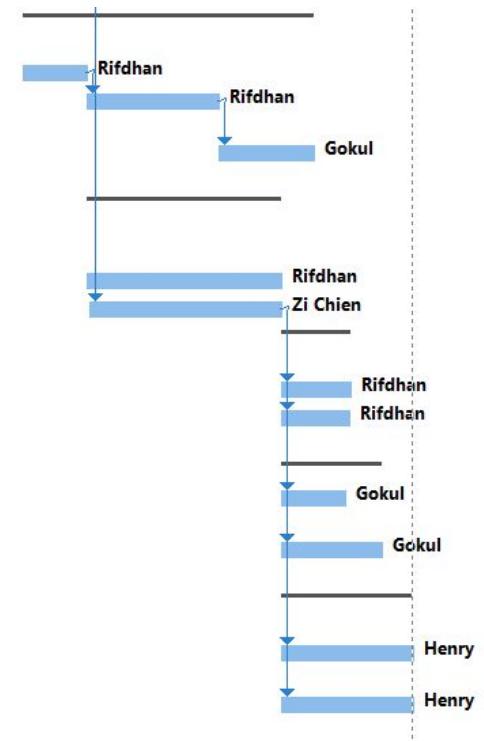


Figure 8 - Final Gantt Chart (part 2 of 2)

### 3.3 Challenges Faced

Listed in Table 3 below are some of the anticipated risks and major problems encountered during the project. Some of the anticipated risks from the Project Proposal did not end up significantly impacting us, but are included for completeness. In addition, some new challenges were encountered since the submission of the Project Proposal, and have been listed below.

**Table 3 - Challenges**

Risk or Problem Encountered	Solution Found
<b>Excessive Computation Time:</b> The ML model could potentially require too long to train or test.	Not a problem. Computation obviously took time, but given the nature of the project, it was reasonable and within expectations.
<b>Limited Storage Space:</b> The disk space on ECF workstations machines is limited, and more space than is available may be required.	While the ECF provided us with a large shared storage location (about 1.5 TB), this turned out to be insufficient to store all our generated Output Images for the various test conditions.  Our solution was to evaluate the performance of the generated images, then delete them. We created infrastructure to generate individual Output Images as required. This frees up the of the disk space used, without sacrificing the ability to inspect individual images when desired.
<b>Inability to Regenerate Lost Data:</b> The ML model may be unable to produce an output image of higher quality than the input image.	Not a problem. The ML model was able to regenerate data with a positive SSIM Score. The only exceptions were a small number of outliers, as discussed in Section 4.2.2.
<b>Learning Curve:</b> Learning to use the Machine Learning libraries and tools took more time than anticipated during the planning of the project.	This proved to be a significant challenge. A lot of time was spent learning and documenting the learning (for the benefit of all members). This contributed a week of delay to the project and is reflected in changes to the Gantt chart.

<p><b>Limited Graphics Memory:</b> The GPUs in the ECF workstations had limited graphics memory, meaning we were unable to explore very complex models.</p>	<p>This capped the level of model complexity (number of layers) that could be experimented with. Through trial and error, we trained up to the maximum complexity given the GPU constraints. Please refer to Section 2.5 for a diagram of the model layers in the GAN.</p>
<p><b>Problems with Long-Running Programs:</b> Training and testing Machine Learning models can take up to several days. However, sometimes our programs were being stopped mid-execution for unknown reasons.</p>	<p>This problem was the primary reason behind the project's delays in the second half of the year. Despite troubleshooting the problem with the ECF lab manager, we have yet to find a solution.</p> <p>Our temporary workaround was to employ a system of progress snapshots which stored the state of training/testing at regular intervals. Should the program be stopped at any point, the last snapshot would act as a new starting point. This system reduced the impact of random stoppages. However, it is still not an ideal solution, and we are continuing to work with the ECF to find a more permanent solution.</p> <p>Overall, this issue added about a week's delay to our project timeline.</p>

# 4.0 Testing and Verification

Author: Gokul

This section details the methodologies used to validate our system, and the results of these tests.

## 4.1 Verification Matrix

Table 4 below discusses how we measured if each project requirement was met. For items which have associated tests, the testing methodology is described in a corresponding section after the table. Lastly, the results for each test are listed in the table as a pass or fail, with details provided in the following section (Section 4.2).

**Table 4 - Final Verification Matrix**

ID	Description	Design Review	Test	Results
1	The system will generate an output image.	X		Pass (Section 4.2.1)
2	The SSIM Score must be strictly positive (i.e. SSIM 2 - SSIM 1 > 0). This means that the Output Image must have a greater SSIM score than the Input Image with reference to the Target Image for any JPEG compression level (between 0% and 100%).		X (Section 4.1.1)	Pass (Section 4.2.2)
3	The system should be able to accept input images in the JPEG format.	X		Pass (Section 4.2.3)
4	The design should produce the Output Image in under 20 seconds on the ECF workstations.		X (Section 4.1.2)	Pass (Section 4.2.4)
5	The design should achieve an SSIM Score of at least 0.05 with Input Images of 10% JPEG compression or lower. This implies that SSIM 2 is greater than SSIM 1 by at least 0.05.		X (Section 4.1.3)	Fail (Section 4.2.5)

7	Using a test set of 50 random Input Images (from a dataset with varied image content), the difference between the highest and lowest SSIM scores among the Output Images should be less than 0.2.		X (Section 4.1.4)	Pass (Section 4.2.6)
8	The Image Regeneration should be done using an ML model.	X		Pass (Section 4.2.7)
9	The model will run on ECF workstations.	X		Pass (Section 4.2.8)
10	The system will use publicly-available datasets.	X		Pass (Section 4.2.9)
11	System will use an existing ML library.	X		Pass (Section 4.2.10)
12	The difference between training and testing SSIM Scores should be less than 0.3.		X (Section 4.1.5)	Pass (Section 4.2.11)

#### 4.1.1 Test: Image Data Regeneration

Measuring the SSIM scores is done with a Python script which evaluates any two given images. We used this script to compute SSIM 1 and SSIM 2, and subtracted them to find the final SSIM Score, as depicted in Figure 2 in Section 2.2. If the SSIM Score is found to be positive, it passes the test.

#### 4.1.2 Test: Timing Model Performance

We used the Linux time utility [9] to evaluate how long a trained models take to produce an Output Image for a given Input Image. This value does not vary significantly between images, so testing it on a few samples was sufficient to determine if it is consistently under the 20 second threshold on ECF workstation machines.

### **4.1.3 Test: Maximizing Regeneration**

A set of test images were compressed to 10% JPEG quality, and the model's performance on them was evaluated in the manner described in Section 4.1.1 above. If the average SSIM Score for this set is greater than 0.05, the test is passed.

### **4.1.4 Test: Variance Testing**

A set of 50 images were randomly selected from the MIRFLICKR-1M dataset, which contains images of various categories [7]. We then employed the testing methodology in Section 4.1.1 to determine the SSIM Score for each of the 50 images, and find the difference between the highest and lowest Scores. If this difference is less than 0.2, the test is passed.

### **4.1.5 Test: Testing for Overfit**

The testing methodology outlined in Section 4.1.1 was used to evaluate the performance of the model when given a two datasets: one consisting of images it saw during training, and one consisting of previously-unseen images. If the difference between the average SSIM Scores for these two sets is below 0.3, the model passes the test.

## **4.2 Module-Level Test Results**

This section contains the results for each of the test outlined in Section 4.1. Evidence is provided for each result when appropriate.

### **4.2.1 Result: Output Image Generation**

The GAN model is able to generate output images without issue. Several examples are available in Appendix F, in the last two sections.

### **4.2.2 Result: Image Data Regeneration**

The test outlined in Section 4.1.1 required a strictly positive SSIM score. Our final model (the GAN) was able to achieve an average SSIM score that was strictly positive for all Standard Test Conditions. Standard Test Conditions are when the model is tested on the same image category that it was trained on, and at the same JPEG compression level. Figure 9 below shows the average SSIM Scores (which are all positive) versus training level for all Standard Test Conditions. However, while the average SSIM Scores were positive, there were individual images which had negative SSIM Scores. These negative-Scoring images constituted about 3.5% of all images tested (see Table 5 below). Therefore, we consider this result as passing the test, since the majority Score positively. An

investigation into these outlier images revealed that there was a considerable degree of pollution in the datasets (see Appendix E). Additionally, there are limitations with SSIM metric itself (see Appendix D), meaning negative Scores do not always indicate visually poorer quality images.

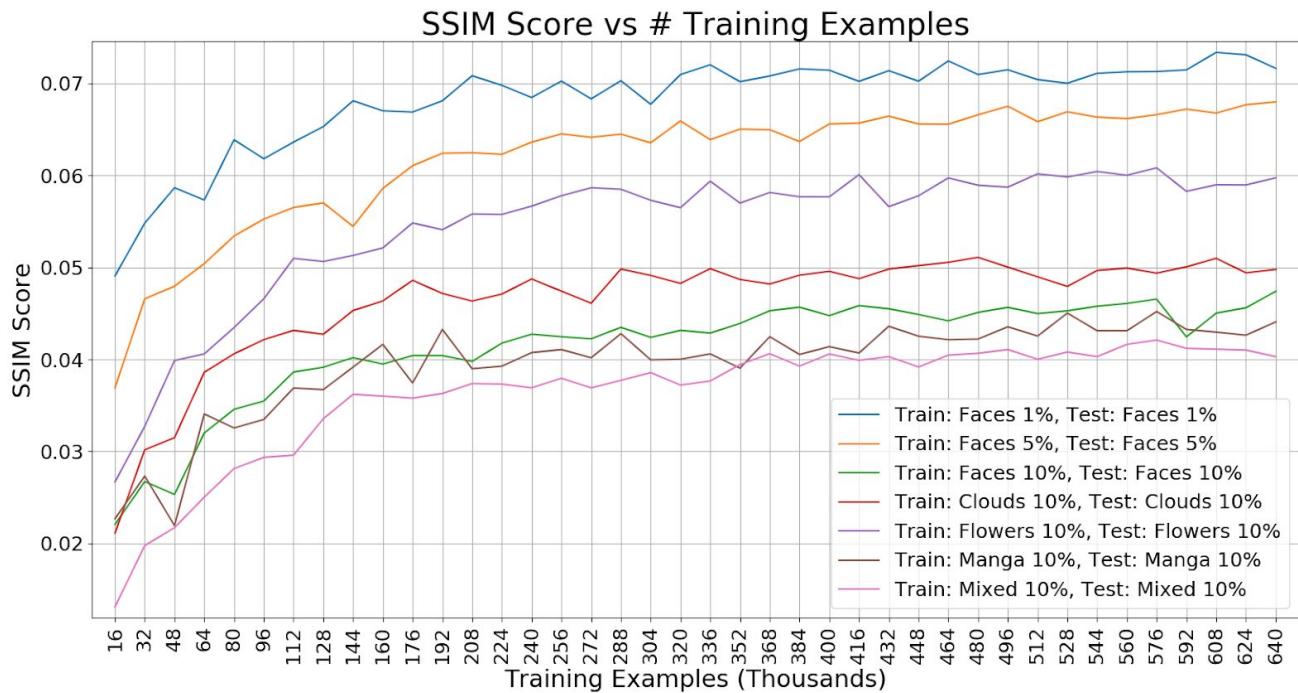


Figure 9 - Performance of model in Standard Test Conditions

**Table 5 - Analysis of Results with Negative SSIM Scores**

Image Category	Compression Level	Quantity of SSIM Scores Below Zero
Faces	10%	2.1% (74/3524)
Faces	5%	1.9% (67/3524)
Faces	1%	7.7% (272/3524)
Clouds	10%	1.5% (28/1856)
Flowers	10%	3.1% (109/3444)
Manga	10%	1.1% (33/3000)
Mixed	10%	6.2% (187/3000)
Average		3.5% (770/21872)

### **4.2.3 Result: JPEG Input Format**

The system uses the PIL Image Processing library for Python [15], which is able to read image files in any standard image format, including JPEG.

### **4.2.4 Result: Timing Model Performance**

The time taken to regenerate 50 random test images from the Faces dataset, at 10% JPEG quality, are presented in Table 6 below. Even while using CPU for computations on the ECF workstations, timings are well below the threshold of 20 seconds as defined in the test.

**Table 6 - Regeneration Time Analysis, Faces Dataset, 10% JPEG Compression**

<b>Test Conditions</b>	<b>Regeneration Time Per Image</b>	
	<b>Mean (s)</b>	<b>Variance (s<sup>2</sup>)</b>
Using CPU for compute	6.3560	1.1073
Using GPU for compute	0.4148	0.0146

### **4.2.5 Result: Maximizing Regeneration**

The average SSIM scores for Standard Test Conditions are presented in Figure 10 below. Note the position of the 0.05 line on the y-axis: only 3 of the 7 test conditions achieved an average SSIM Score above this line (the red graph was very close to passing but was just below it). Therefore, our model is not able to satisfy this objective consistently. A possible factor that contributed to the failure is the inconsistency of SSIM as a metric. Refer to Appendix D for a discussion about this. Moreover, if the compressed Input Images already have high SSIM values, there is not much room for our model to improve upon, and the SSIM Scores are more tightly bounded. This may have been a contributing factor to the poor performance of the Manga Dataset (refer to Appendix E).

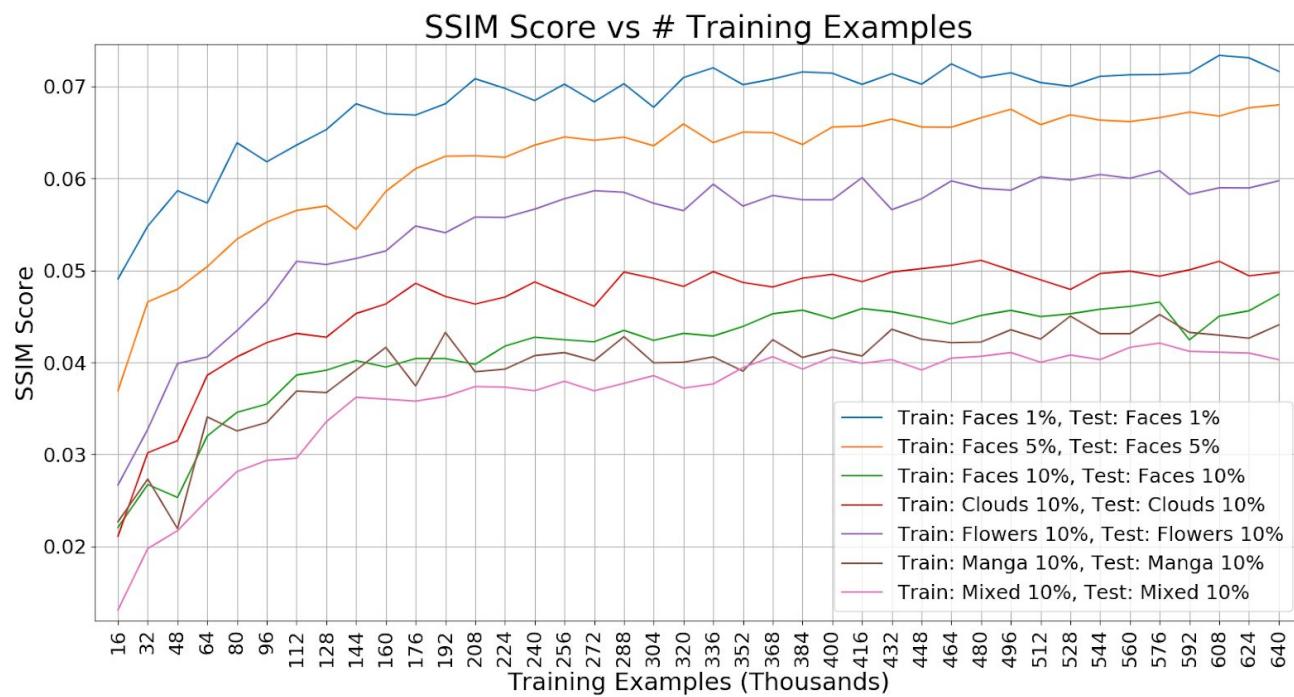


Figure 10 - Model performance in Standard Test Conditions

#### 4.2.6 Result: Variance Testing

The model was trained and tested on a Mixed dataset, and the performance results in terms of SSIM Score versus training level is presented in Figure 11 below. The difference between the minimum and maximum SSIM Score after 640,000 training examples was just over 0.15, and is below the threshold of 0.2 set by the test.

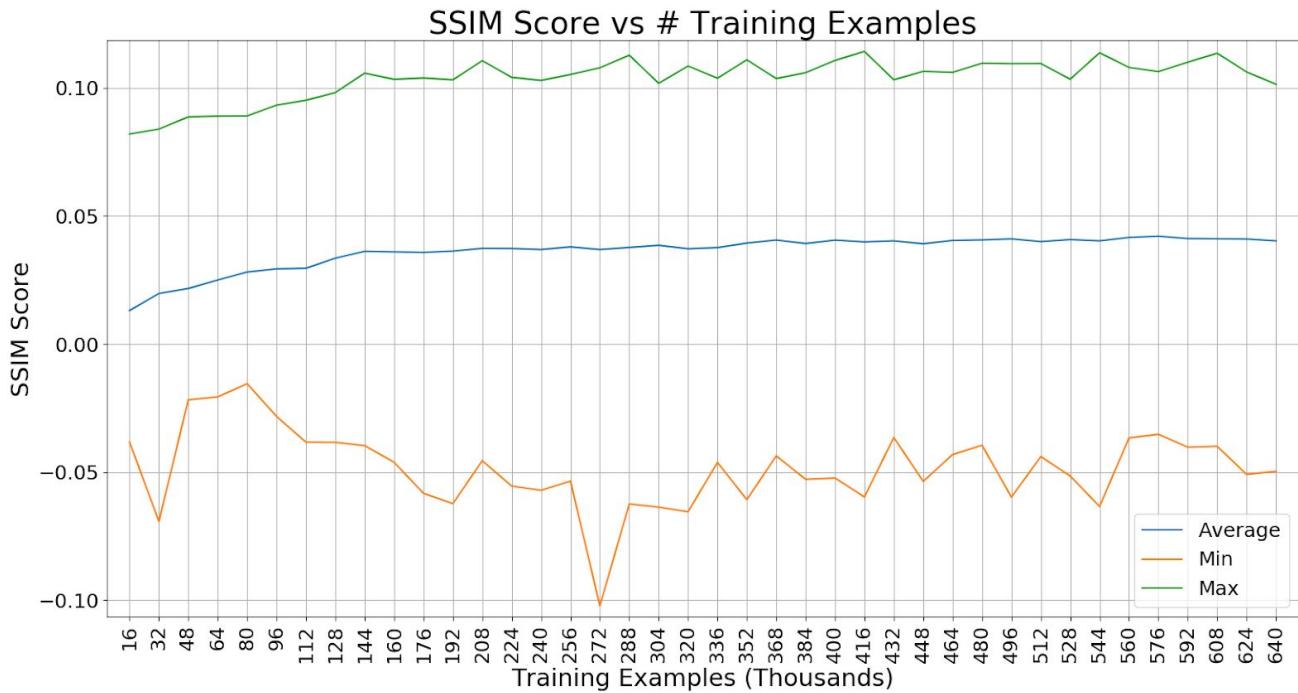


Figure 11 - Performance of model on Mixed dataset, with minimum and maximum individual scores per training level

#### 4.2.7 Result: Uses a Machine Learning Model

The GAN (Generative Adversarial Network) model is a type of Machine Learning model. It consists of two separate neural networks internally, and is discussed in more detail in Section 2.5.

#### 4.2.8 Result: ECF Workstation Environment

All of our models were trained and tested on the ECF workstation machines. Refer to Section 3.3 for the challenges and limitations of using the ECF machines.

#### 4.2.9 Result: Public Datasets

The two datasets used in the project were the MIRFLICKR-1M dataset [7], and the Manga 109 dataset [16], both of which are publicly-available. Please refer to Appendix E for a discussion of datasets used.

#### 4.2.10 Result: Existing Machine Learning Libraries

Both the GAN and the CNN models use the Chainer Machine Learning library [17] on Python. This is a well-established library and is available for use free of charge.

#### 4.2.11 Result: Testing for Overfit

This test evaluates the degree of overfit in the model. We measured the SSIM Scores on images the model has already seen (from the training set), and the SSIM Score on images the model has never seen before (from the testing set). The difference between these Scores was much less than the 0.3 maximum for this test. The two plots are presented in Figure 12 below. The actual difference between these ends up being under 0.0025 after 640,000 training examples, thus satisfying the requirement.

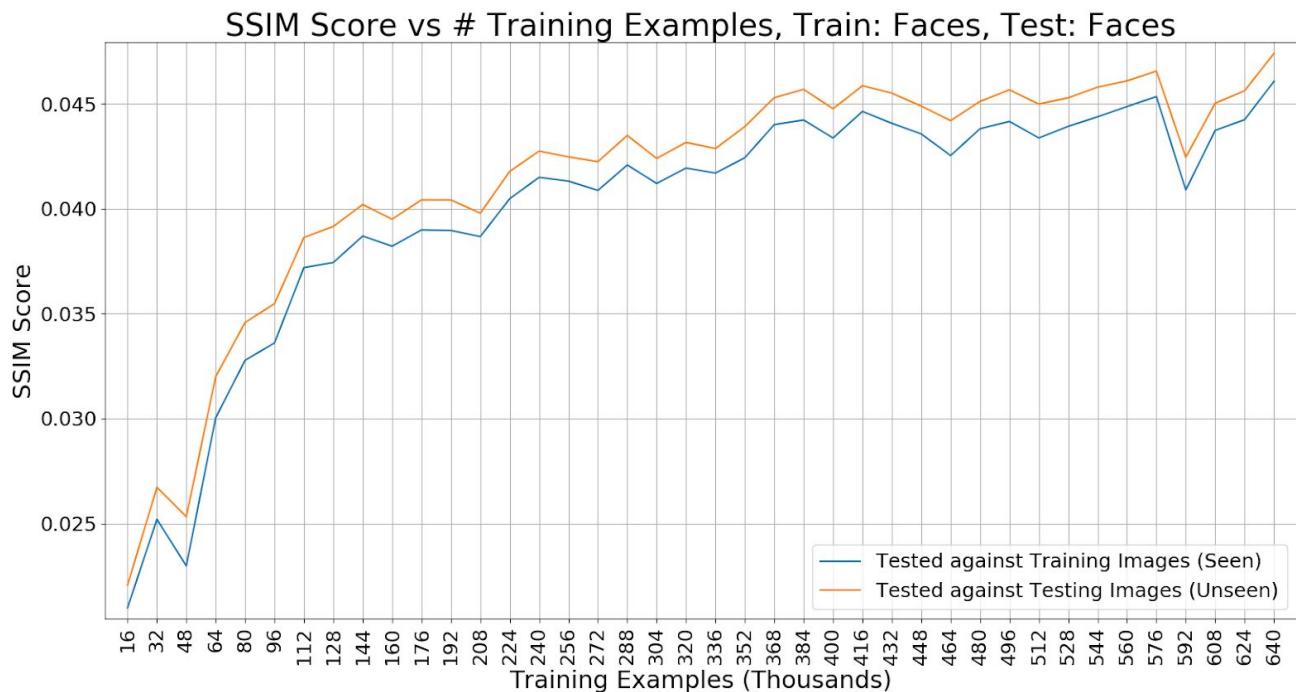


Figure 12 - Performance of GAN on seen and unseen images

## 5.0 Conclusion

Author: Gokul

The project's intended goal was to explore the feasibility and effectiveness of using Machine Learning techniques to regenerate data lost through JPEG Compression. We were able to prove, through experimentation and analysis of the resultant data, that not only are ML techniques feasible, they are remarkably effective for Image Regeneration.

Over the course of the project, we developed two working models: the GAN and the CNN. The CNN was discarded due to its inferior performance, and the final model selected was the GAN model. The GAN model meets all Functions, Objectives and Constraints except the Objective to maximize regeneration. The models had SSIM Scores in the range of 0.04 to 0.06 for most test cases, which resulted in half of them failing to meet the 0.05 SSIM Score threshold of the objective, but this is mostly due to the limitations of SSIM Score as a metric, and pollution in the datasets.

The viability of SSIM and MSE as metrics for determining model performance was also explored during the project. We found SSIM to be a much less accurate metric than originally anticipated. Though the performance numbers achieved by the model seem negligible, in actuality, the performance was subjectively very good. Additionally, we discovered some counter-intuitive relationships between various properties, such as oscillatory behaviour in SSIM Values at different JPEG compression levels.

Our group is satisfied with the level of progress we have achieved in this project despite our limited initial knowledge of the subject, and the difficulties encountered with utilizing the ECF workstations. There are many practical applications to this technology, such as reducing the file size impact that images have on mobile devices, freeing up valuable storage space. Our initial foray into this technique could pave the way for even more intensive studies to come.

We are currently two weeks behind schedule, but we have adjusted tasks in the work plan to ensure the project is complete in time for the Design Fair on April 4th. For the Design Fair, we plan to showcase the successfulness of our project by having a live demonstration that takes portraits of participants, JPEG-compresses them to a low quality level, and regenerates the compressed images using a model trained on Faces.

## 6.0 References

- [1] Sayood, K. (2017). Introduction To Data Compression. 4th ed. [S.L.]: Morgan Kaufmann Publisher.
- [2] "Historical yearly trends in the usage of site elements, October 2017", W3techs.com, 2017. [Online]. Available: [https://w3techs.com/technologies/history\\_overview/site\\_element/all/y](https://w3techs.com/technologies/history_overview/site_element/all/y). [Accessed: 26- Oct- 2017].
- [3] Xiang Xie, GuoLin Li, ZhiHua Wang, Chun Zhang, DongMei Li and XiaoWen Li, "A novel method of lossy image compression for digital image sensors with Bayer color filter arrays," 2005 IEEE International Symposium on Circuits and Systems, 2005, pp. 4995-4998 Vol. 5. doi: 10.1109/ISCAS.2005.1465755.
- [3] "Usage Statistics of Image File Formats for Websites, October 2017", W3techs.com, 2017. [Online]. Available: [https://w3techs.com/technologies/overview/image\\_format/all](https://w3techs.com/technologies/overview/image_format/all). [Accessed: 26- Oct- 2017].
- [4] L. Handayani, "Read more »", *Gabrielladabreau.blogspot.ca*, 2017. [Online]. Available: <http://gabrielladabreau.blogspot.ca/2014/10/read-more.html>. [Accessed: 26- Oct- 2017].
- [5] Z. Wang, A. Bovik, H. Sheikh and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.
- [6] R. West, "Training Set vs. Test Set", *Content.nexosis.com*, 2017. [Online]. Available: <http://content.nexosis.com/blog/training-set-vs.-test-set>. [Accessed: 26- Oct- 2017].
- [7] S16a.org. (n.d.). *MIRFLICKR-1M s16a extension | Semantic Multimedia*. [online] Available at: <http://s16a.org/mirflickr> [Accessed 15 Jan. 2018].
- [8] pixiv. (n.d.). *pixiv*. [online] Available at: <http://pixiv.net> [Accessed 15 Jan. 2018].

- [9] Man7.org. (n.d.). *time(1) - Linux manual page*. [online] Available at: <http://man7.org/linux/man-pages/man1/time.1.html> [Accessed 15 Jan. 2018].
- [10] Chris Nicholson, S. (2017). A Beginner's Guide to Generative Adversarial Networks (GANs) - Deeplearning4j: Open-source, Distributed Deep Learning for the JVM. [online] Deeplearning4j.org. Available at: <https://deeplearning4j.org/generative-adversarial-network#a-beginners-guide-to-generative-adversarial-networks-gans> [Accessed 15 Jan. 2018].
- [11] Stergiou, C. and Siganos, D. (n.d.). *Neural Networks*. [online] Neural Networks. Available at: [https://www.doc.ic.ac.uk/~nd/surprise\\_96/journal/vol4/cs11/report.html](https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html) [Accessed 15 Jan. 2018].
- [12] Staff, I. (n.d.). *Overfitting*. [online] Investopedia. Available at: <https://www.investopedia.com/terms/o/overfitting.asp> [Accessed 15 Jan. 2018].
- [14] Nvidia.ca. (n.d.). *Machine Learning Applications for High Performance Computing - NVIDIA*. [online] Available at: <http://www.nvidia.ca/object/machine-learning.html> [Accessed 16 Jan. 2018].
- [15] "Python Imaging Library (PIL)", *Pythonware.com*, 2018. [Online]. Available: <http://www.pythonware.com/products/pil/>. [Accessed: 22- Mar- 2018].
- [16] "Manga109", *Manga109.org*, 2018. [Online]. Available: <http://www.manga109.org/en/>. [Accessed: 22- Mar- 2018].
- [17] *Chainer*, 2018. [Online]. Available: <https://chainer.org/>. [Accessed: 22- Mar- 2018].

# Appendix A: Gantt Chart History

Author: Rifdhan

## A.1 Original Gantt Chart - Project Proposal

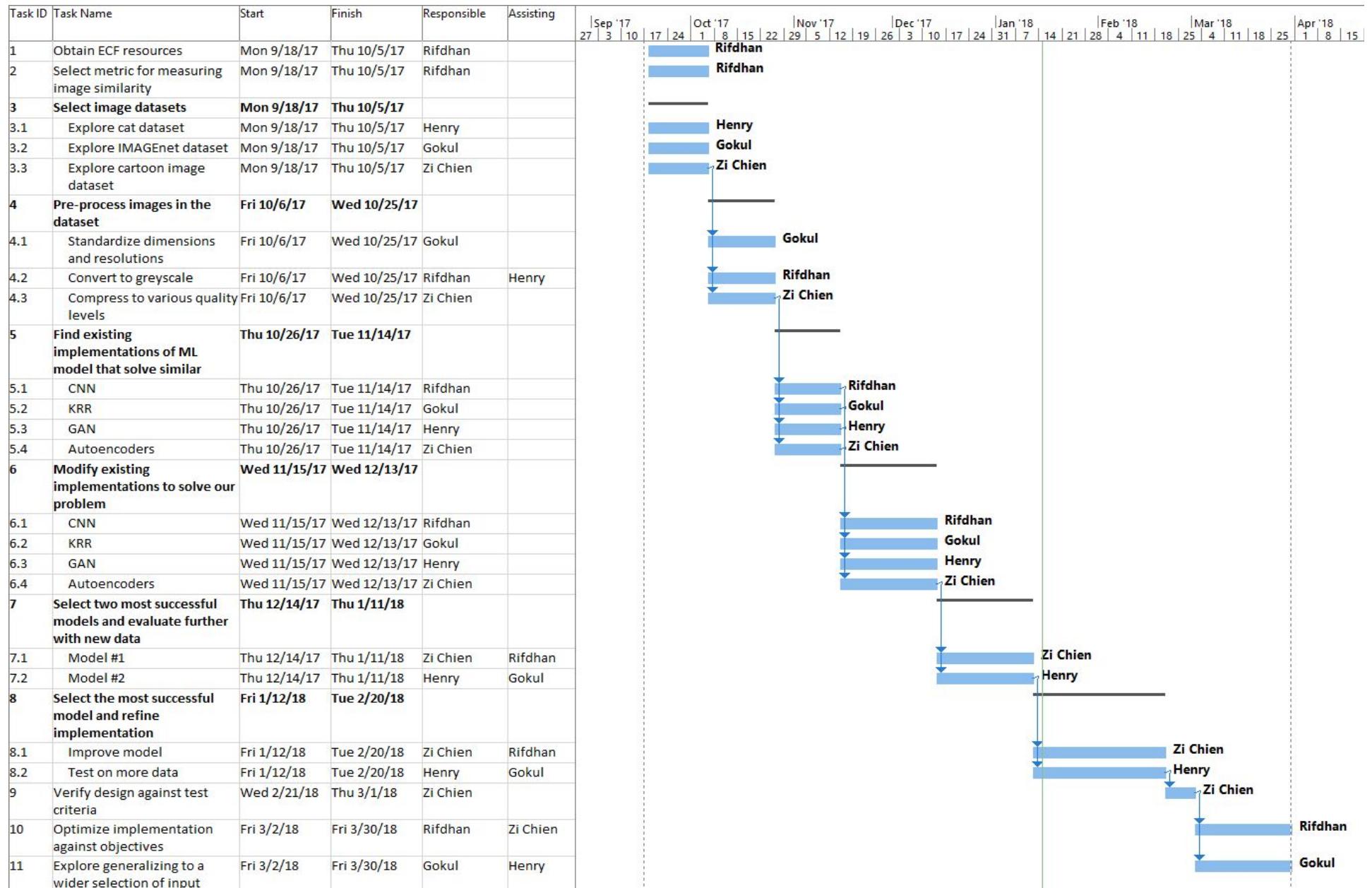


Figure A1 - Original Gantt Chart from Project Proposal.

## A.2 Updated Gantt Chart - Progress Report

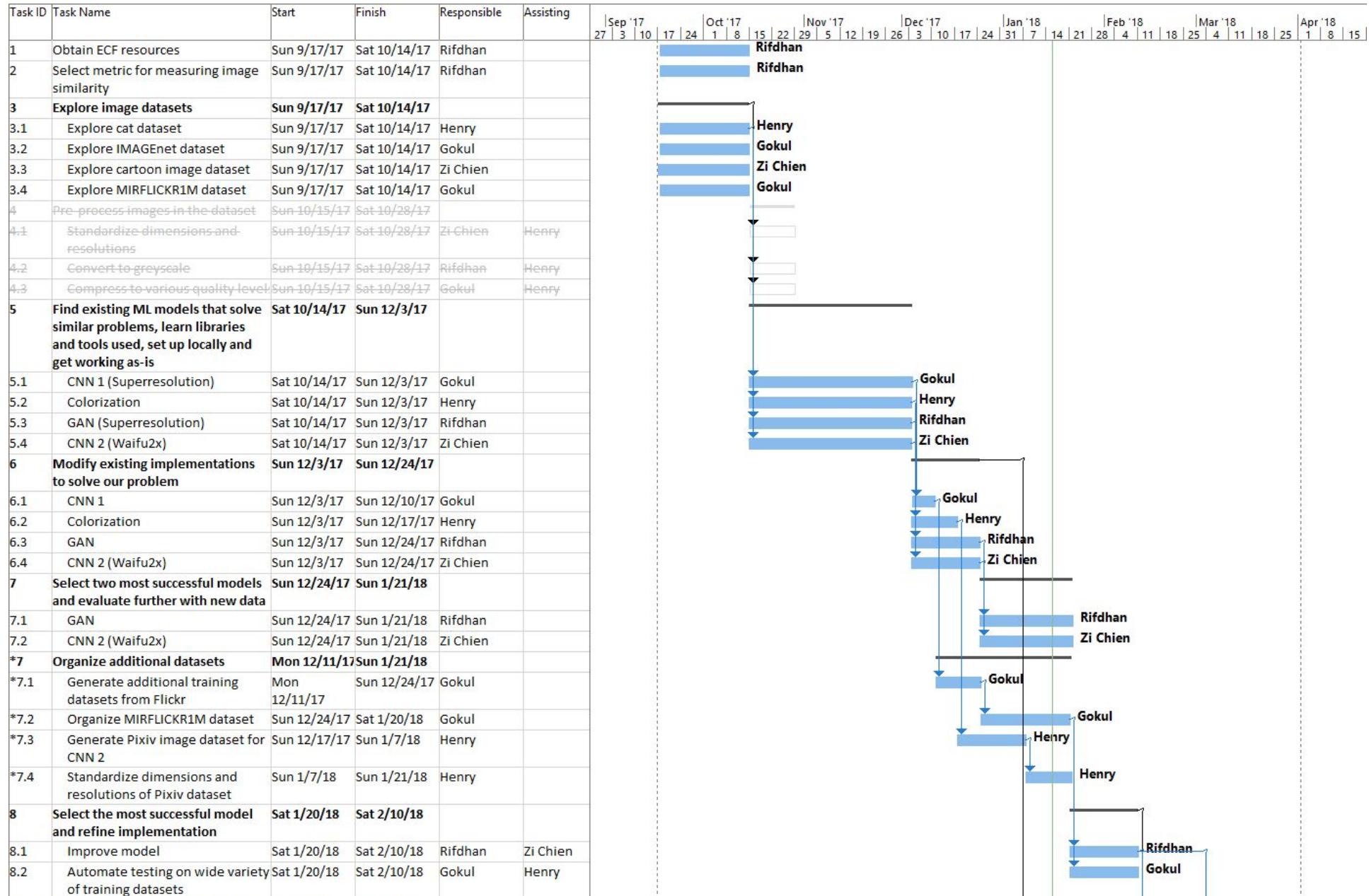


Figure A2 - Gantt Chart from Progress Report (part 1 of 2).

<b>9</b>	<b>Verify design against test criteria</b>	<b>Sun 1/7/18</b>	<b>Sat 3/3/18</b>		
9.1	Make tool to measure SSIM and MSE errors	Sun 1/7/18	Sat 1/20/18	Gokul	
9.2	Document statistics on trained models	Sat 2/10/18	Sat 3/3/18	Zi Chien	Gokul
<b>10</b>	<b>Optimize implementation against objectives</b>	<b>Sun 1/7/18</b>	<b>Sun 3/11/18</b>		
10.1	Obtain GPU support	Sun 1/7/18	Sat 1/20/18	Henry	
10.2	Explore generalizing to wider selection of input images	Sat 2/10/18	Sat 3/3/18	Gokul	Zi Chien
10.3	Modify code to utilize GPUs and train models using GPUs	Sat 2/10/18	Sun 3/11/18	Henry	Rifdhan
11	Train and test final models across a variety of datasets and compare performance	Sat 3/3/18	Sat 3/31/18	Gokul	Henry
<b>12</b>	<b>Develop interactive system to demonstrate project at design fair</b>	<b>Sun 3/4/18</b>	<b>Sun 4/1/18</b>		
12.1	Frontend mobile app to take photos and display images	Sun 3/4/18	Sun 4/1/18	Zi Chien	
12.2	Backend server to interface with ML model	Sun 3/4/18	Sun 4/1/18	Rifdhan	

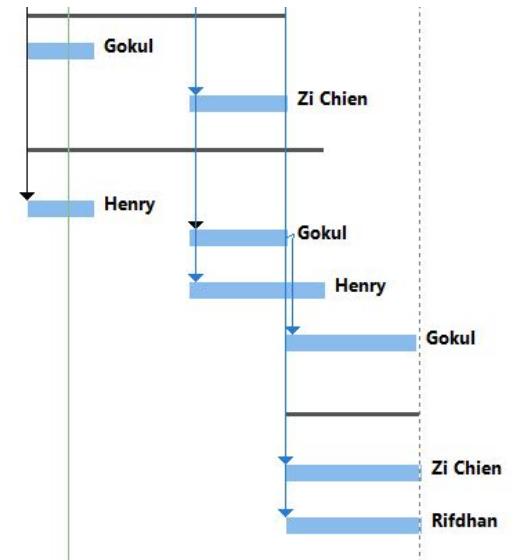


Figure A2 - Gantt Chart from Progress Report (part 2 of 2).

## Appendix B: Original Validation Tests

Author: Rifdhan

Table B1 below contains the original Verification Matrix from the Project Proposal. The sections following it detail the two tests outlined in the table.

**Table B1 - Verification Matrix**

ID	Description	Review of Design	Test
1	The system will generate an output image.	X	
2	The system will regenerate some of the data lost when compressing the input image.		X (Section B.1)
3	The input will be in the JPEG format.	X	
8	The image regeneration will be done using an ML model.	X	
9	The model will run on standard computing hardware.	X	
10	The system will use publicly-available datasets.	X	
11	System will use an existing ML model.	X	
12	ML model should not overfit on training data		X (Section B.2)

### B.1 SSIM Measurement Methodology

The SSIM measurement methodology, developed by Wang et al., was designed specifically for the comparison of two images, with the goal of being close to how humans evaluate similarity in images [8]. Its value is inversely proportional to the difference between the images. A value of 1 indicates no difference, and a value of 0 indicates no similarity. It is robust against minor changes across the entire image (eg. change in contrast), but identifies large changes, even in a small portions of the image.

### B.2 Testing for Overfit

The standard method of verifying a trained ML system is to test it on previously-unseen data [9]. We will set aside a portion of the dataset (referred to as the test data) specifically for this purpose prior to training the system. We will use around 20% of the dataset as the test set. The error in the testing set should be no more than 3x the error in the training set.

# Appendix C: Existing Image Quality Metrics

Author: Rifdhan

One of the project's primary objectives is to achieve a higher quality image than the source. Therefore, we need a methodology to quantify similarity in images. Outlined below are a few possible methods.

## C.1 Mean Square Error (MSE)

MSE (Equation C1 below) is a fairly common method for measuring error which has applications outside image comparison. The MSE value is directly proportional to the difference between the images. A value of 0 indicates no difference. It is not very useful for comparing image data, as it compares individual pixels, without considering context or changes in neighbouring pixels.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$

Equation C1 - Mean-Square Error

For example, if the entire image became slightly darker, the MSE would be large, even though the perceived change in the image would not be significant. Conversely, if a portion of the image was modified completely with the rest remaining unchanged, the MSE would not be as high, as the unchanged pixels represent the majority.

## C.2 Structural Similarity (SSIM)

The SSIM measurement methodology (Equation C2 below), developed by Wang et al., was designed specifically for the comparison of two images, with the goal of being close to how humans evaluate similarity in images [5]. Its value is inversely proportional to the difference between the images. A value of 1 indicates no difference, and a value of 0 indicates no similarity.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Equation C2 - Structural Similarity (SSIM)

It is robust against minor changes across the entire image (eg. change in brightness), but identifies drastic changes, even in a small portions of the image.

Presented in Figure C1 below is a comparison between MSE and SSIM when comparing three images. For the contrast example, the MSE error is 1401, while SSIM gives 0.78. For the Photoshopped example, the MSE error is 1077, while SSIM gives 0.69. These results show that SSIM estimates perceived differences better than MSE. Therefore, we will use SSIM as the comparison method between two images for the project.



Figure C1 - A comparison between the MSE and SSIM methods for evaluating similarity between images (taken from [5]). Contrast: MSE=1401, SSIM=0.78 and Photoshopped: MSE=1077, SSIM=0.69

# Appendix D: Metric Evaluation

Author: Gokul

Various tests are used to verify the correct and desirable functionality of the model (Section 4.0). This section analyses the effectiveness of metrics used in the project, and presents some insights into relationships between them.

## D.1 The Ideal Image Storage Format

In our view, an ideal image storage format has three properties: High Quality, Low File Size, and Quick Viewing. The Figure D1 below illustrates how the three objectives compete with each other. Each of the three corners of the triangle illustrate maximizing an objective at the cost of the other two.

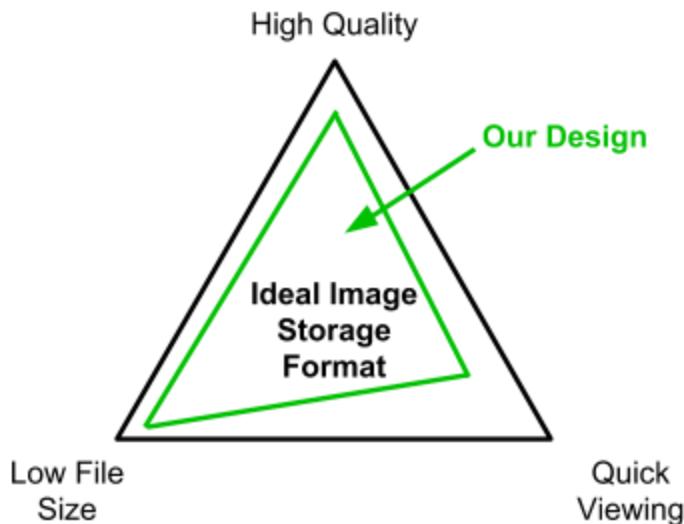


Figure D1 - Competing Objectives in the Ideal Image Storage Format

Our design, with reference to these competing objectives, seeks to reduce storage space and maintain image quality (as much as possible), at the cost of viewing speed. Each of the three objectives can be measured using different metrics. The metrics are listed in Table D1 below.

**Table D1 - Metric Types**

Image Quality (High Quality)	Storage Space (Low File Size)	Viewing Speed (Quick Viewing)
SSIM Value	File Size	Regeneration Time (per Pixel)
SSIM Score		MACs (Multiply Accumulates)
MSE Value		
JPEG Compression Level		

## D.1.1 Image Quality

Image quality metrics measure how high or low quality an image is. There are different popular techniques used to approximate the quality of an image. The metrics listed above (with the exception of JPEG Compression Level, which is not a metric) are not absolute benchmarks of quality. Rather, they compare two images, and their value is proportional (directly or indirectly) to the difference between the images. SSIM Value and SSIM Score are used to approximate how humans perceive quality in images. Human test subjects are used to justify the accuracy of their metrics. In contrast, MSE is used to calculate the per pixel difference between two given images.

Image Quality Considerations:

- SSIM Value: A measure of the similarity between two images. Identical images have a value of 1 and dissimilar images have values lower than 1.
- SSIM Score: A measure of the difference in similarity between two image pairs. Each pair contains an original image and the score is reflective of which image pair is more similar to the original. For more details, refer to Section 2.2.
- MSE Value: The Euclidean distance difference between two images. Identical images have a value of 0 and non-identical images have values higher than 0.
- JPEG Compression Level: The percentage quality at which a JPEG image is compressed at. Lower percentages incur more data loss, but have smaller file sizes.

## D.1.2 File Size

The amount of storage space required for a file can be measured by its file size. A higher file size indicates more disk space required for the image. It is desirable to reduce the amount space taken up by a given image, so that the storage medium can accommodate more data.

## D.1.3 Viewing Speed

Viewing speed measures the amount of time to load and render an image. With modern computers, images in standard file formats appear almost instantaneously on the display when opened. However, as our project needs to regenerate images on demand, the load time is noticeably increased. It will certainly not be as instantaneous, as machine learning models are computationally intensive.

There are two ways of measuring the speed. One method is to record the amount of physical time the processor takes to regenerate an image and divide it by the number of pixels in contained in the

image. This is called Regeneration Time. It is specific to the processor used (an older, slower processor will take more time than a newer, faster one).

The second method is to measure viewing speed as proportional to model complexity. A more complex model (with more layers and weights) involves more computations, which for a any given processor will take up more real time. This method is processor-agnostic and is measured by calculating the number of Multiply-Accumulate operations (MACs) done by the Neural Network to regenerate the output image. See Appendix F, Section F.4 for a MACs analysis of the GAN model.

## D.2 The Effects of JPEG Compression on Images

JPEG Compression works by removing higher frequencies components from an image in the frequency domain. It lowers the file size of an image at the cost of reduced image quality. The relationship between JPEG compression, file size and image quality will be presented in this section. This will be done for each of the selected image categories. For more details on the image categories, please refer to Appendix E.

### D.2.1 Effects on Manga Images

Manga images are the most radically different category explored in the project. The significant differences in performance and characteristics between Manga images and the other categories can be attributed to the following properties:

- Manga images are primarily in the monochrome spectrum (with only black or white pixels), with almost no colour usage. Other image categories contain a full range of colours.
- Manga images generally contain blocks of solid colours, with little or no gradients. Shading is done by using black lines instead of a different colour. This is unique to the Manga dataset - most other categories contain gradients.
- Manga images are drawn by hand, whereas other image categories are taken from digital cameras. As a result, Manga images do not contain the imperfections and minute details of the real world, which results in less details that can be lost during compression.

### D.2.2 File Size Effects

A lower JPEG Compression level implies lower file size. In Figure D2 below, the file size ratio of the compressed file to the original file (a PNG) is measured versus various JPEG Compression levels. File size ratio is computed as the Compressed Image file size divided by the Original Image file size, expressed as a percentage. The metric was evaluated for various image categories.

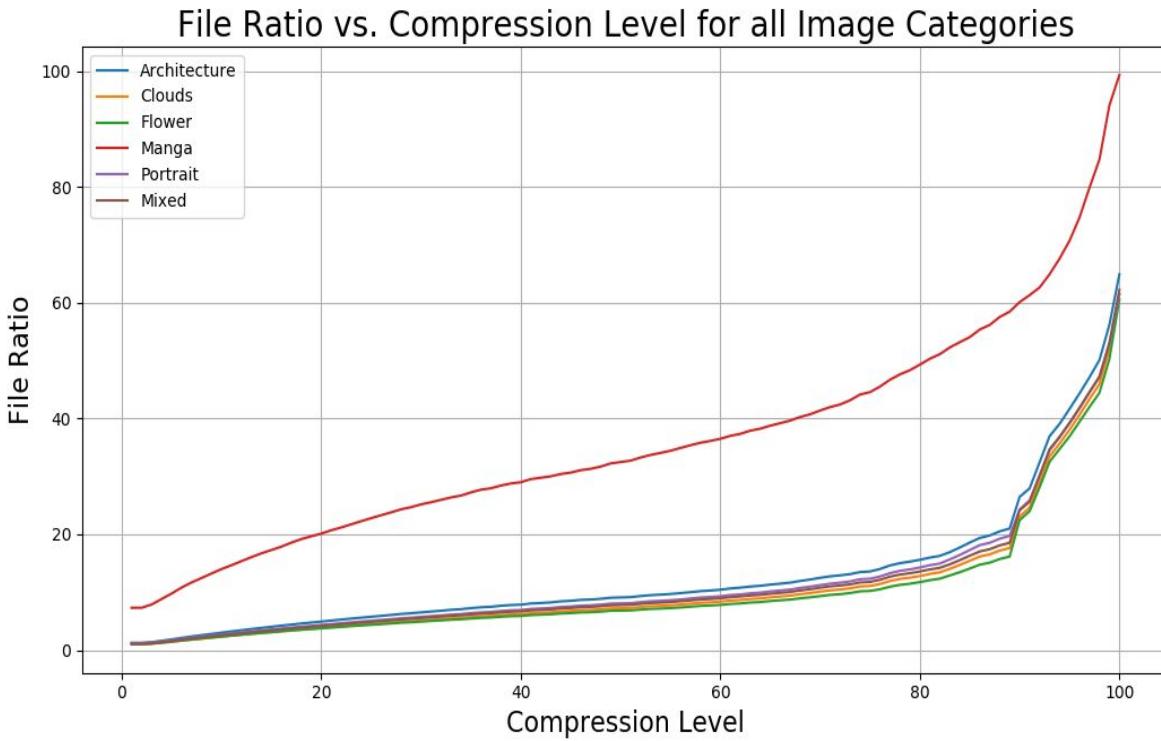


Figure D2 - File Ratio (percentage of the original file size) vs JPEG compression level

Observations:

- The trend (across all categories) is not linear. Having a JPEG Compression level of 20% provides for a file size of significantly less than 20% of the original PNG image.
- The non-linear trend varies depending on the level of compression. From JPEG Compression levels of 100% to 90%, the trend is steeper than the second segment of 90% and lower.
- The effect of JPEG Compression on file size is category specific. Manga images have a different trend with an inflection point between the 40% to 60% JPEG Compression range. Moreover, the 100% JPEG Compression file size is almost equal to the original PNG file. This could be attributed to no (to very few) high frequencies components to discard initially. This is different from the other categories which lose around 40% immediately upon conversion from PNG to JPEG. Note that 100% JPEG compression is not lossless like it might imply.

### D.2.3 Image Quality Effects

A lower JPEG compression level should indicate higher levels of distortion in the image and therefore a lower SSIM value. In Figure D3 below, the SSIM value between the Original Image and the Compressed Image (for various compression levels) have been measured. The average SSIM value per compression level for a category has been plotted below for all categories.

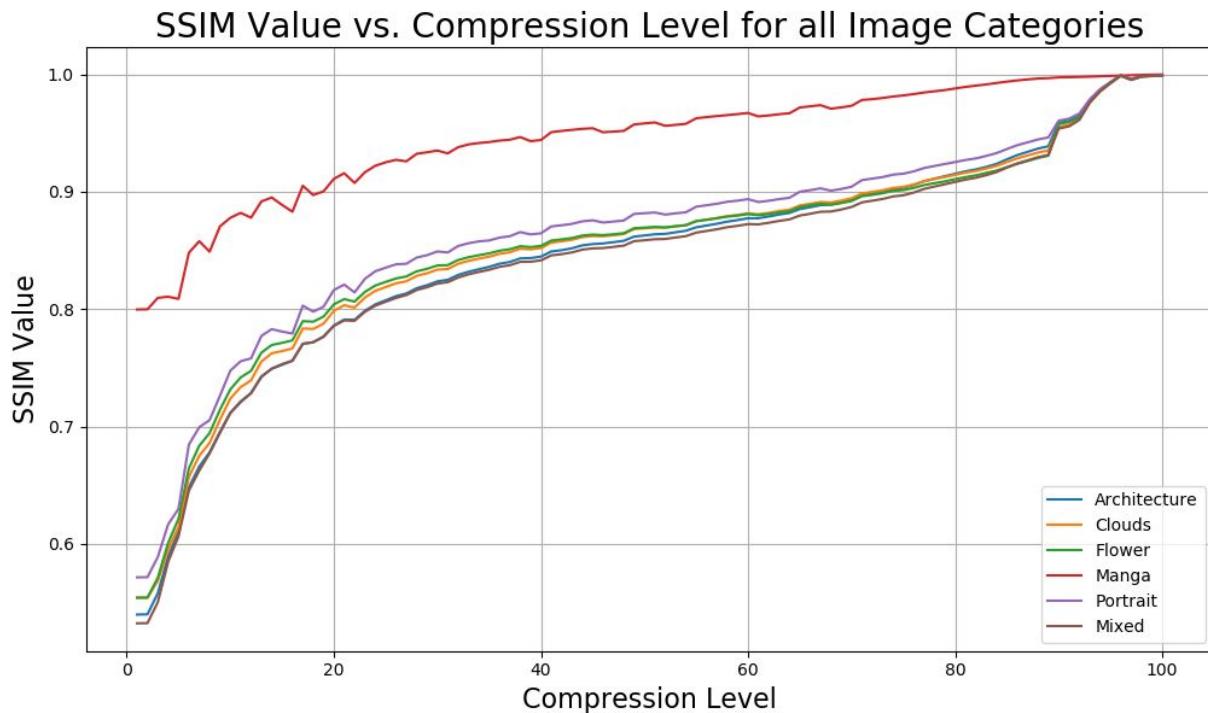


Figure D3 - SSIM Value vs JPEG compression level for select image categories

#### Observations:

- The decrease in SSIM Value with respect to JPEG compression level is not linear. Assuming that the SSIM trend measures image quality similar to the way that a human would, this implies that JPEG Compression distortion to the quality of image is not linear with JPEG compression level.
- Manga Images are quite resistant to structural distortion caused by JPEG. The worst JPEG Compression level, 1% still produces a 0.8 SSIM value for manga images versus the approximately 0.55 SSIM value for other categories. This significantly impacts the amount of improvement (measured by SSIM Score) that our model can achieve, since the SSIM Value for Manga images is already very high at any compression level.

#### D.2.4 Non-Linearity of SSIM Value with JPEG Compression Level

In the Figure D3 above, we deal with the average value of many images for a single compression value for a single category. The average trends containing averages are smooth (for most categories). In Figure D4 below, we plot the SSIM Value vs. Compression level for three select sample images from the Mixed image category.

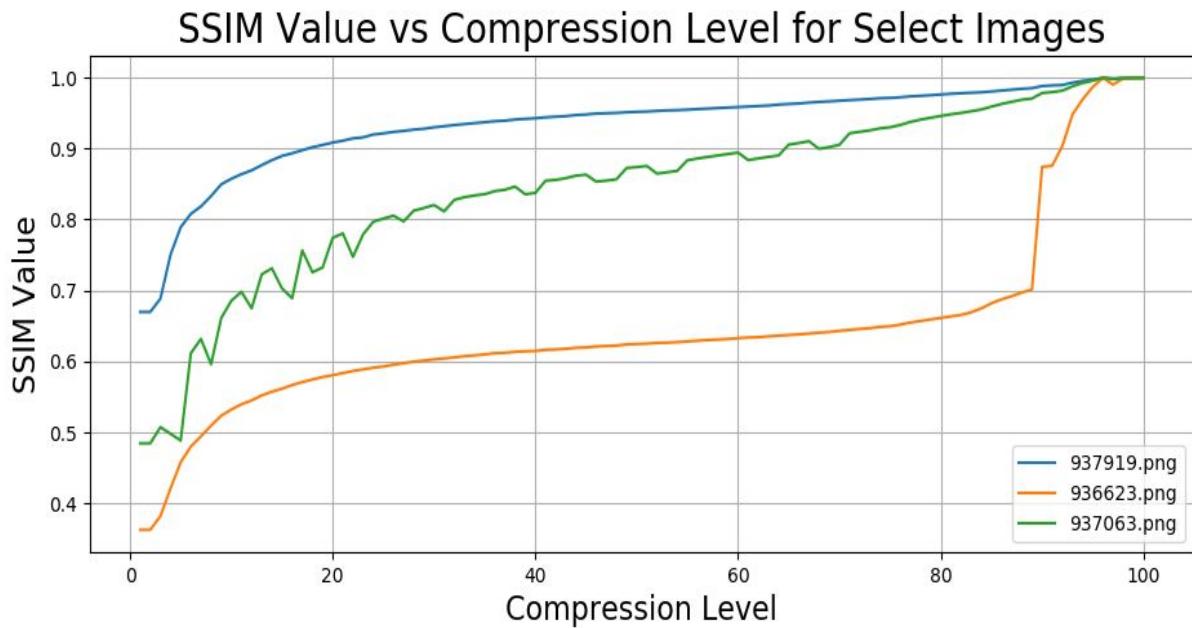


Figure D4 - SSIM Value vs JPEG compression level for select images

Observations:

- The removal of JPEG frequencies have inconsistent and disproportionate effects on the SSIM Value. SSIM is based on the similarity between the structures of two images. In the graph above, the SSIM Value oscillates as JPEG compression changes. As a result, sometimes lower compression levels can achieve higher SSIM Values. This is a counter-intuitive result, as the frequency removal would imply a monotonically-decreasing structural quality. This behaviour was observed in several images.

## D.2.5 Samples of JPEG Compressed Images

Presented below are a few examples of JPEG-compressed images, demonstrating some of the aforementioned properties and characteristics.

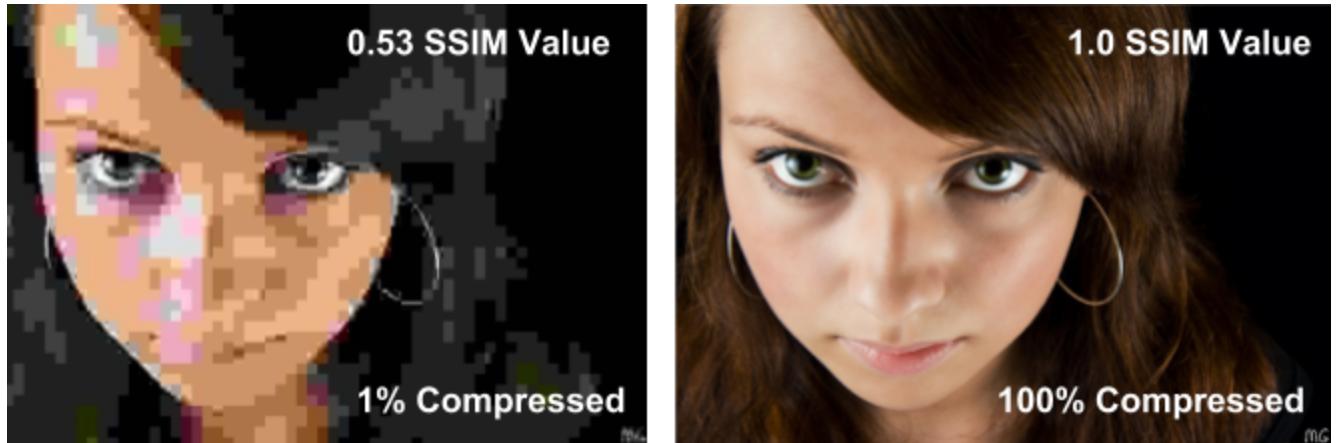


Figure D5 - JPEG compression on Face images

Figure D5 demonstrates the effects of extreme JPEG compression on an image. At 1% JPEG quality, the image data is severely degraded.

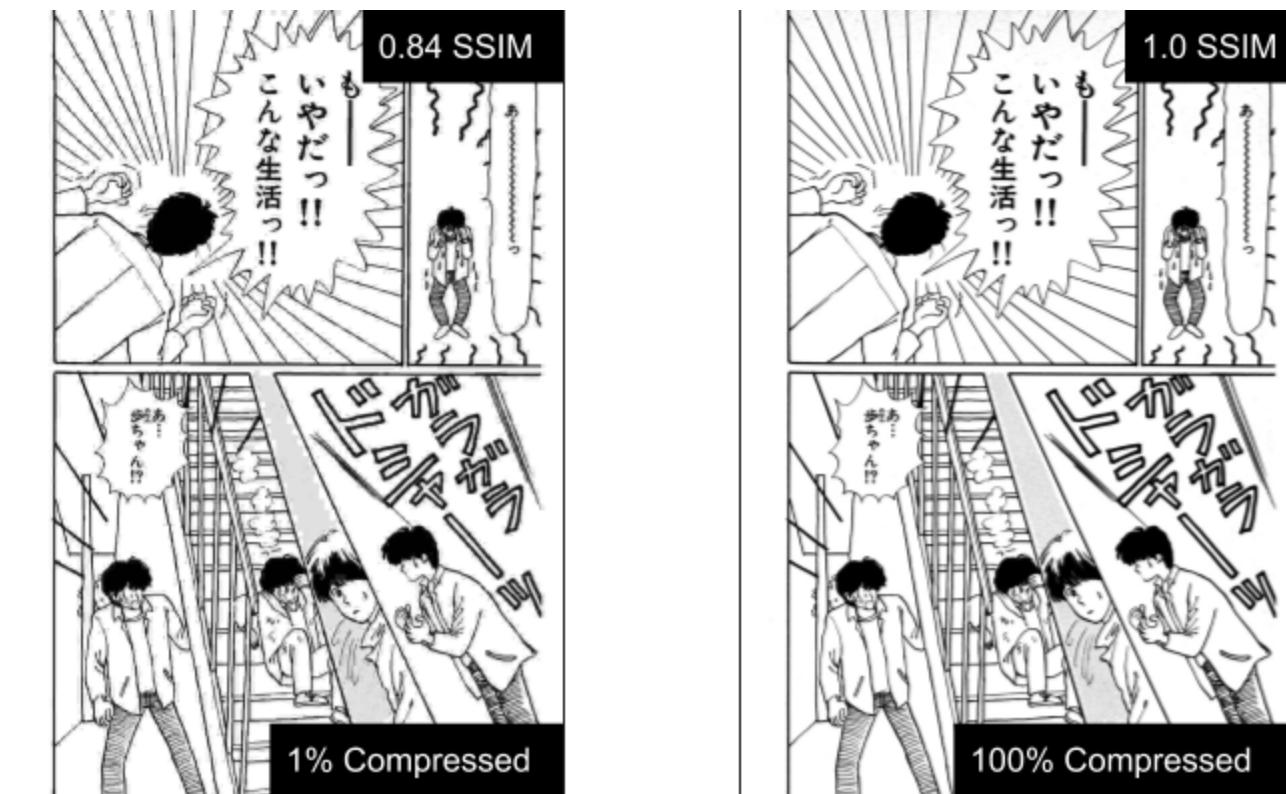


Figure D6 - JPEG compression on Manga images

Figure D6 shows how little reduction in quality there is when JPEG compression is applied to Manga images. Even at 1% quality, the image looks mostly the same. Compare this to the Face image in Figure D5 from before, also at 1% JPEG quality.

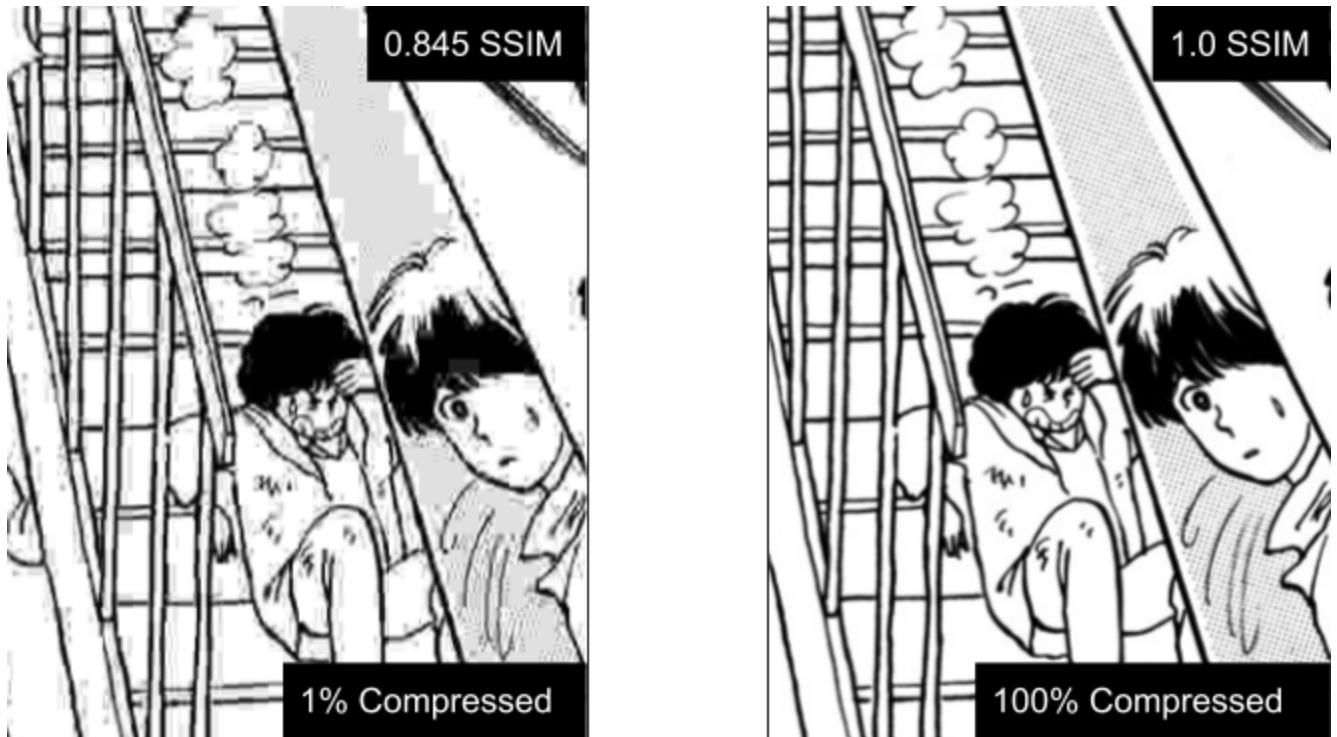


Figure D7 - JPEG compression on Manga images, close-up

Figure D7 shows a zoomed-in crop of the image from Figure D7. The JPEG compression artifacting is more noticeable when inspecting the image closely in this way, but is still not very severe.

### D.3 Shortcomings of SSIM

While SSIM strives to be a good metric to approximate how humans perceive similarity in images, unfortunately it falls short of achieving this goal in the context of JPEG compression. As seen in the results in the previous sections, there are many counter-intuitive properties of SSIM as compared to compression level. Presented in Figure D8 below is the performance of SSIM alongside three other metrics, as compared to data provided by human test subjects, taken from [5]. The y-axes in each graph show the quality ratings assigned by test subjects for a number of images, and the x-axes show the score provided by each metric. SSIM, at the bottom-right, is definitely not linear, and suffers a great degree of inaccuracy below around 30% human-rated quality. For example, human ratings of 20% quality could result in SSIM Values as low as 0.5, or as high as 0.8.

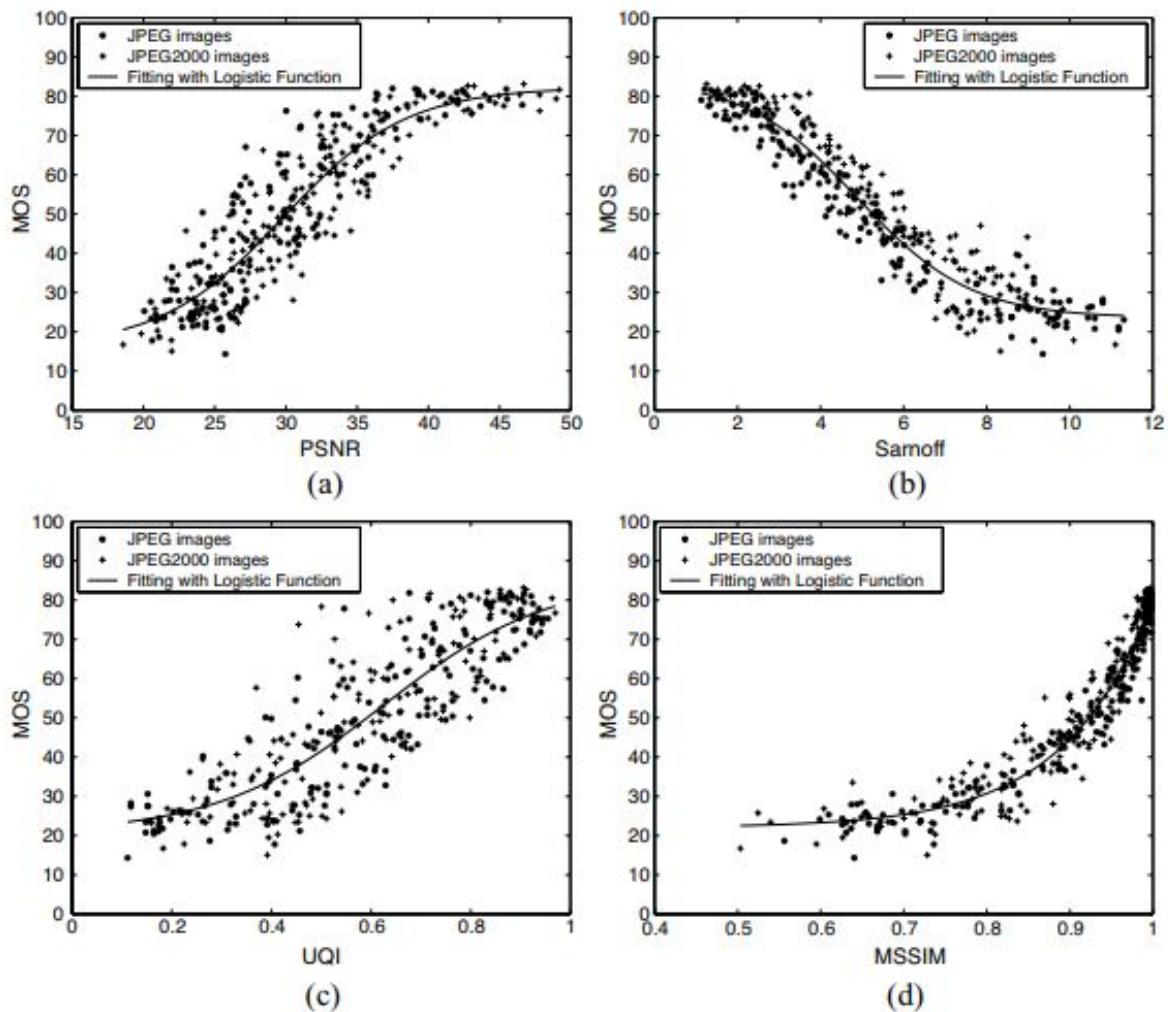


Figure D8 - Performance of SSIM as a metric to approximate human perception of similarity [5]

# Appendix E: Image Categories and Datasets

Author: Gokul

This section describes the data sources used for images in the project.

## E.1 Image Categories

We initially prepared several datasets consisting of particular image categories from the MIRFLICKR-1M dataset. The sizes of each prepared category is shown in Figure E1 below. However, we didn't have enough time to evaluate all of these datasets. The datasets we actually used are presented in Table E1 following the figure.

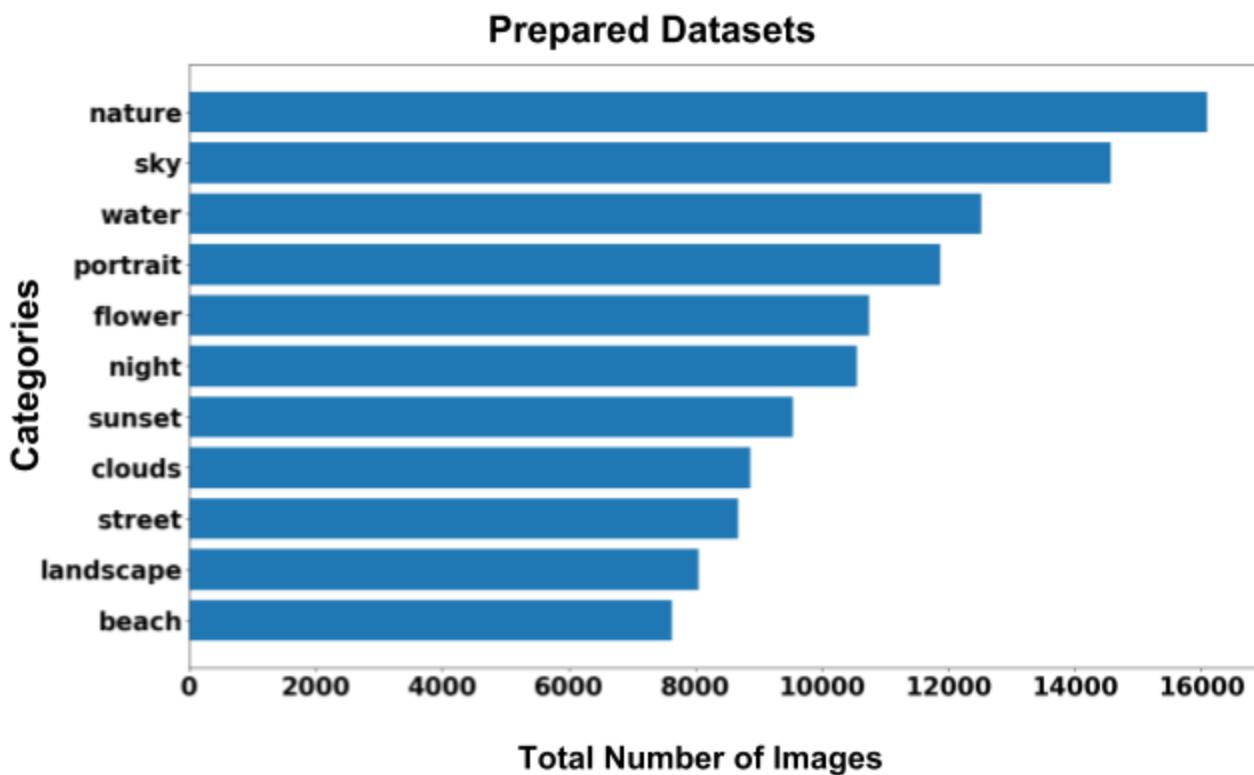


Figure E1 - Sizes of image categories in the MIRFLICKR-1M dataset

Table E1 - Sizes of image categories used in project

Category	Image Count (Train/Test)	Source
Faces	Train: 14200 images Test: 3524 images	MIRFLICKR-1M [7]
Flowers	Train: 13778 images Test: 3444 images	MIRFLICKR-1M [7]

Clouds	Train: 7429 images Test: 1856 images	MIRFLICKR-1M [7]
Manga	Train: 16914 images Test: 3000 images	Manga 109 [16]
Mixed	Train: 15,001 images Test: 3,000 images	MIRFLICKR-1M [7]

The table below contains a list of publicly available dataset sources which the project has used, along with a description of their images.

**Table E2 - Dataset sources**

Dataset	Description
MIRFLICKR-1M [7]	A collection of 1 million photos uploaded to Flickr that have been curated and tagged.
Manga 109 [16]	A collection of manga images, predominantly black and white.

## E.2 Dataset Pollution

Upon inspecting the MIRFLICKR-1M dataset, it has been observed that there are some images inside each category that do not belong to that category, or represent poor examples of an image for that category. It is difficult to determine an exact number of such anomalies, as such an inspection would have to be manually done for each category. Each category has tens of thousands of images, so a manual analysis would be too time-intensive, and the removal of such images becomes an intractable process. We refer to this situation as dataset pollution, and it degrades the performance of our models to some degree, but is not prevalent enough to have a major impact. Figure E2 below provides a few examples in the Faces dataset that aren't good samples for that category.

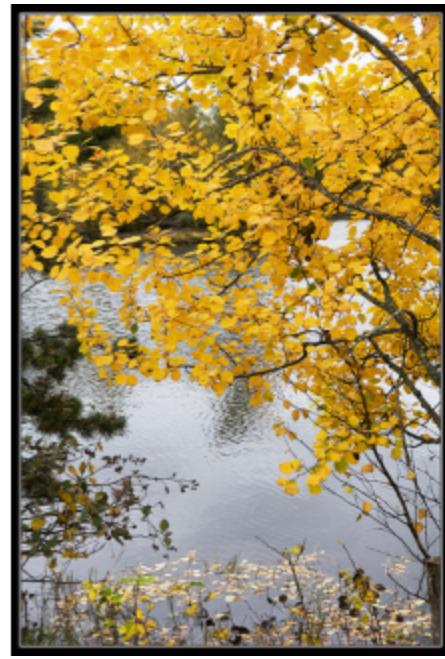
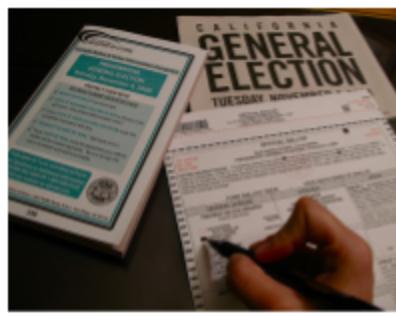


Figure E2 - Examples of poor sample images in the Faces dataset

# Appendix F: Results from GAN Model

Author: Rifdhan

This section contains some key results obtained from the GAN model, and some corresponding observations and conclusions for these results.

## F.1 Performance on All Datasets

Presented below are the SSIM Scores and Values for all Standard Test Conditions (trained and tested on the same image category and JPEG compression level), when trained and tested using the Default Model (see Section F.4 in this appendix for results when modifying the model itself). Note that SSIM Values are when comparing the Output Images to the Original Images, with the value at 0 training level corresponding to the SSIM Value between the Input Images and Original Images. A graph for SSIM Score is provided in Figure F1, followed by a graph for SSIM value in Figure F2.

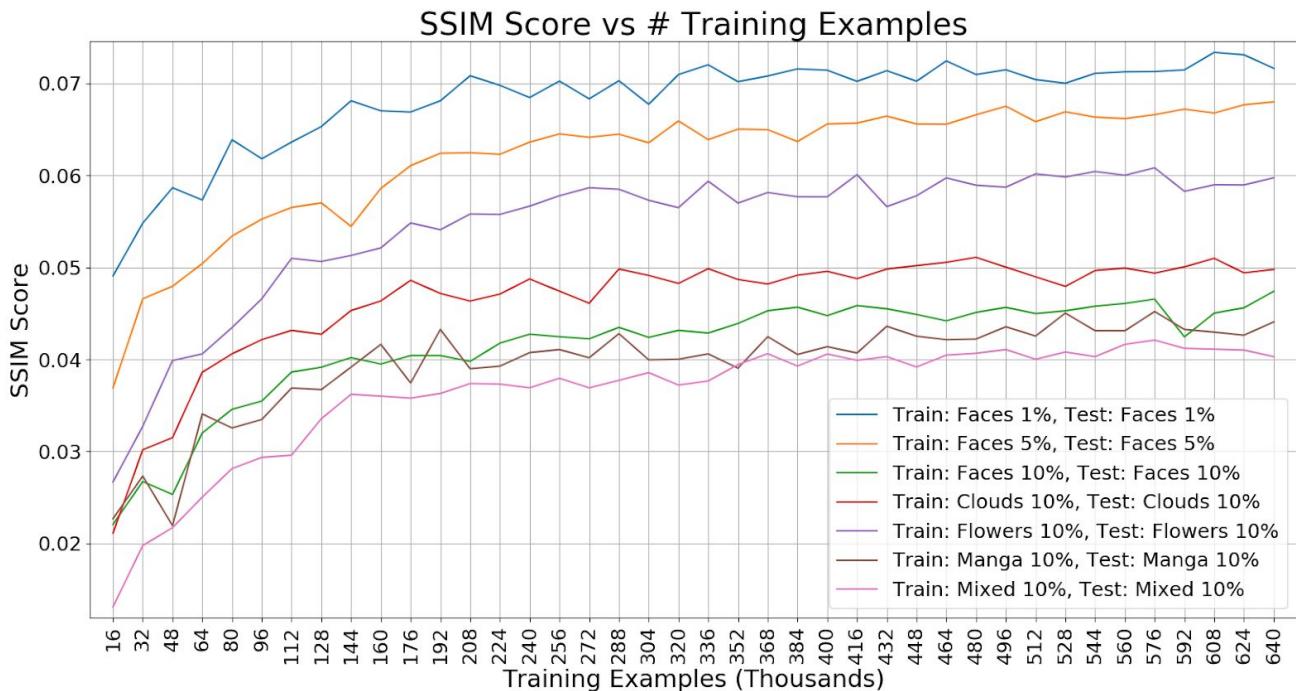


Figure F1 - Performance for Standard Test Conditions (SSIM Score)

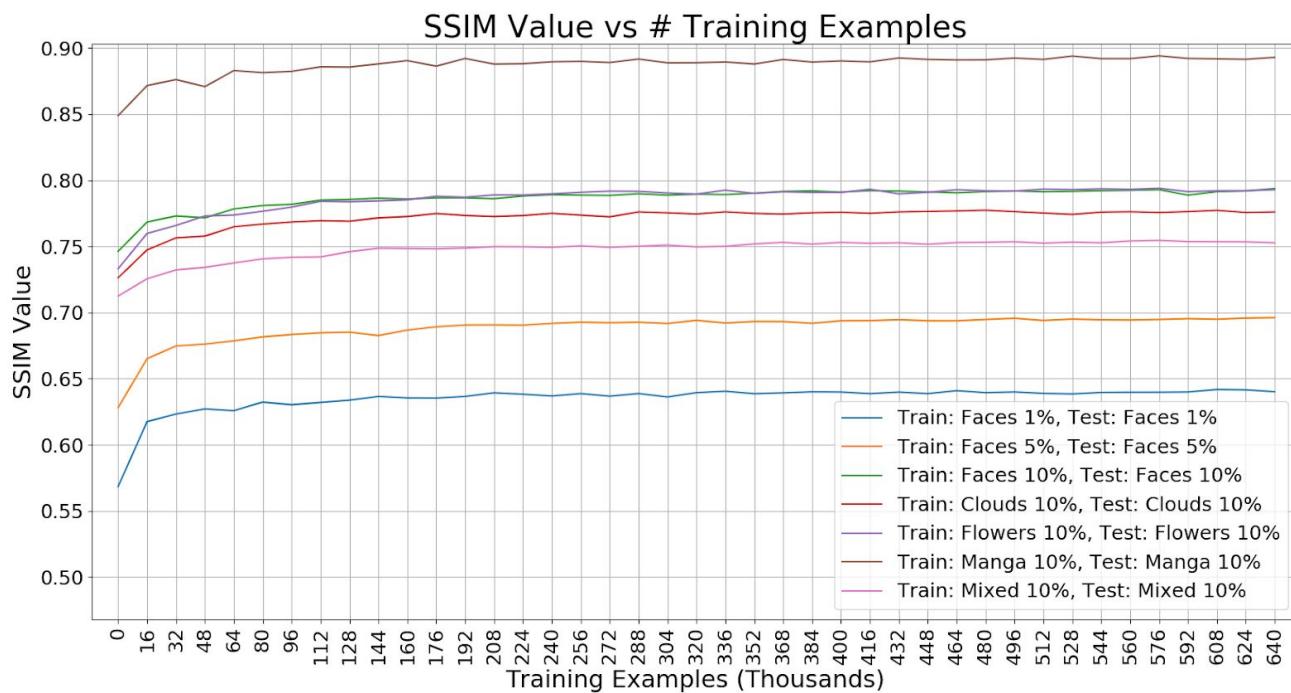


Figure F2 - Performance for Standard Test Conditions (SSIM Value)

#### Observations:

- Models which had a higher performance with SSIM Score were typically worse performers on the SSIM Value graph. This inverse relationship between the two graphs shows that when the Input Images are at a lower SSIM Value to begin with, there is greater potential for regeneration to occur, resulting in higher SSIM Scores.
- Training and testing on lower JPEG compression level Input Images (such as 5% or 1%) results in the highest SSIM Scores, but the lowest SSIM Values.
- The Manga dataset seems to have the lowest performance, most likely due to the limited degradation JPEG compression has on these types of images, and thus the limited regeneration potential (see Appendix D for a discussion on this).
- The model trained and tested on a Mixed image dataset performed worse than the three models trained and tested on a specific image category (Faces, Clouds, and Flowers respectively). This suggests that the model is able to learn features specific to each image category to gain higher performance over just a generic image dataset.

## F.2 Performance when Crossing Datasets, by Training

This section explores performance when testing on a category different than the one trained on, and compares these results to training and testing on the same image category. The results are organized by what dataset was used for training.

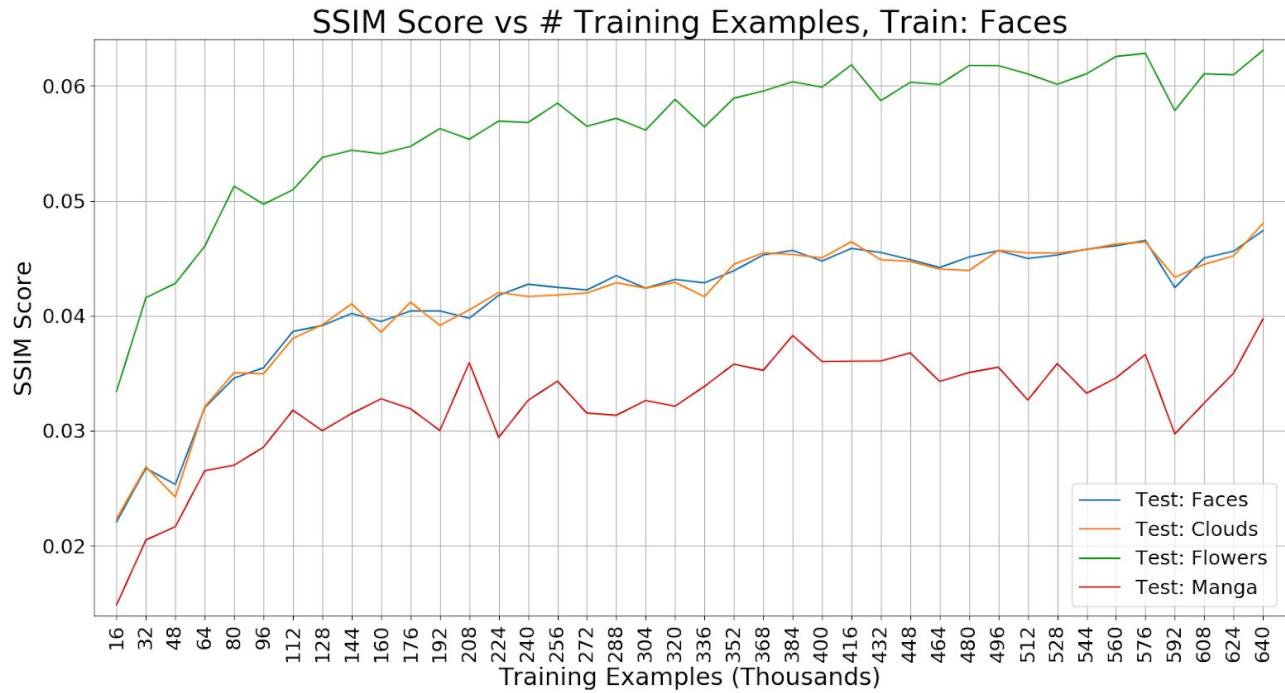


Figure F3 - Performance when trained on Faces

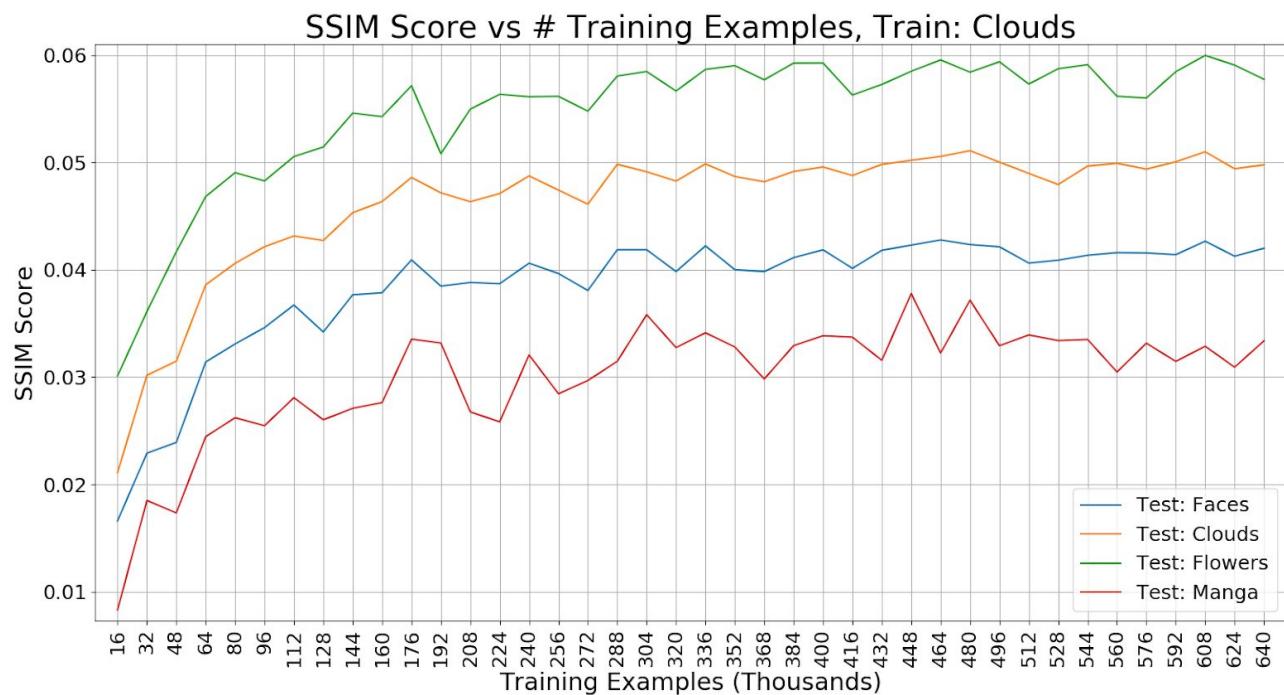


Figure F4 - Performance when trained on Clouds

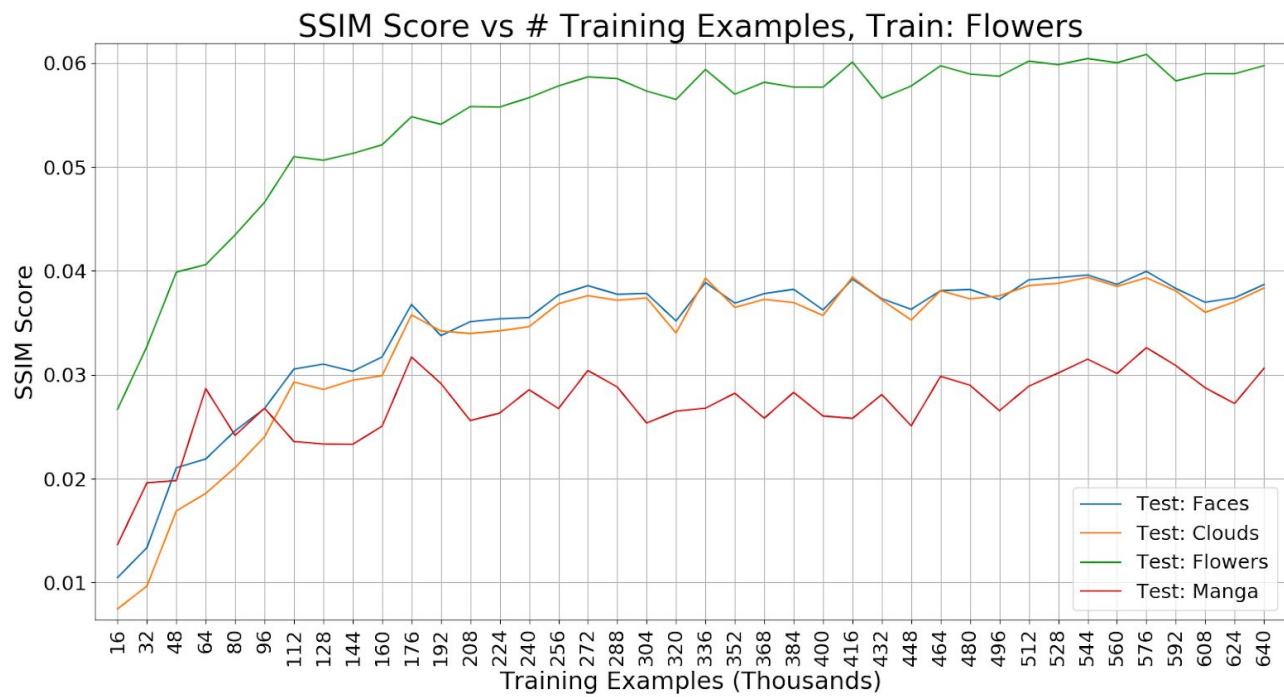


Figure F5 - Performance when trained on Flowers

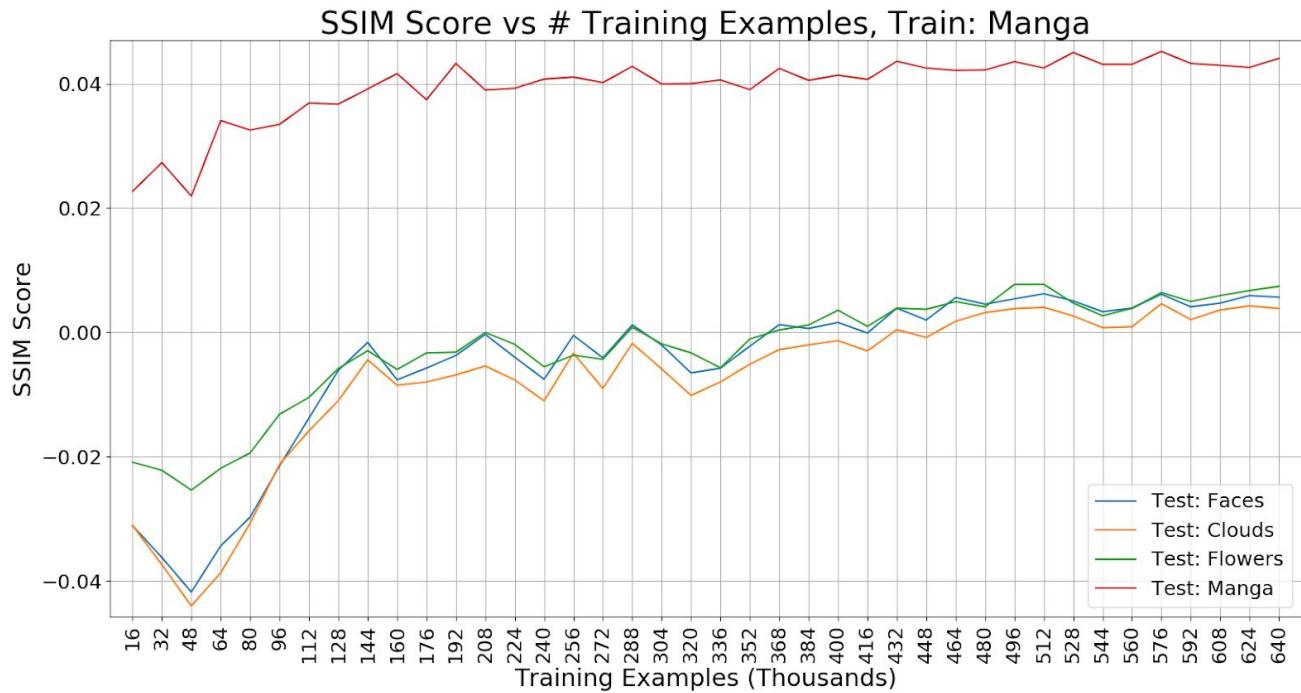


Figure F6 - Performance when trained on Manga

Observations:

- Performance was best on the Flowers test set regardless of training image category, except for the model trained on Manga images. This suggests the Flowers dataset was the easiest to perform regeneration on.
- All models performed poorly when tested on Manga images, except the model trained on Manga. However, the model trained on Manga only performed well on Manga, and performed poorly in every other image category. This is likely due to the very limited degradation Manga images experience during JPEG compression (discussed in Appendix D).

### F.3 Performance when Crossing Datasets, by Testing

This section explores performance when testing on a category different than the one trained on, and compares these results to training and testing on the same image category. The results are organized by what dataset was used for testing.

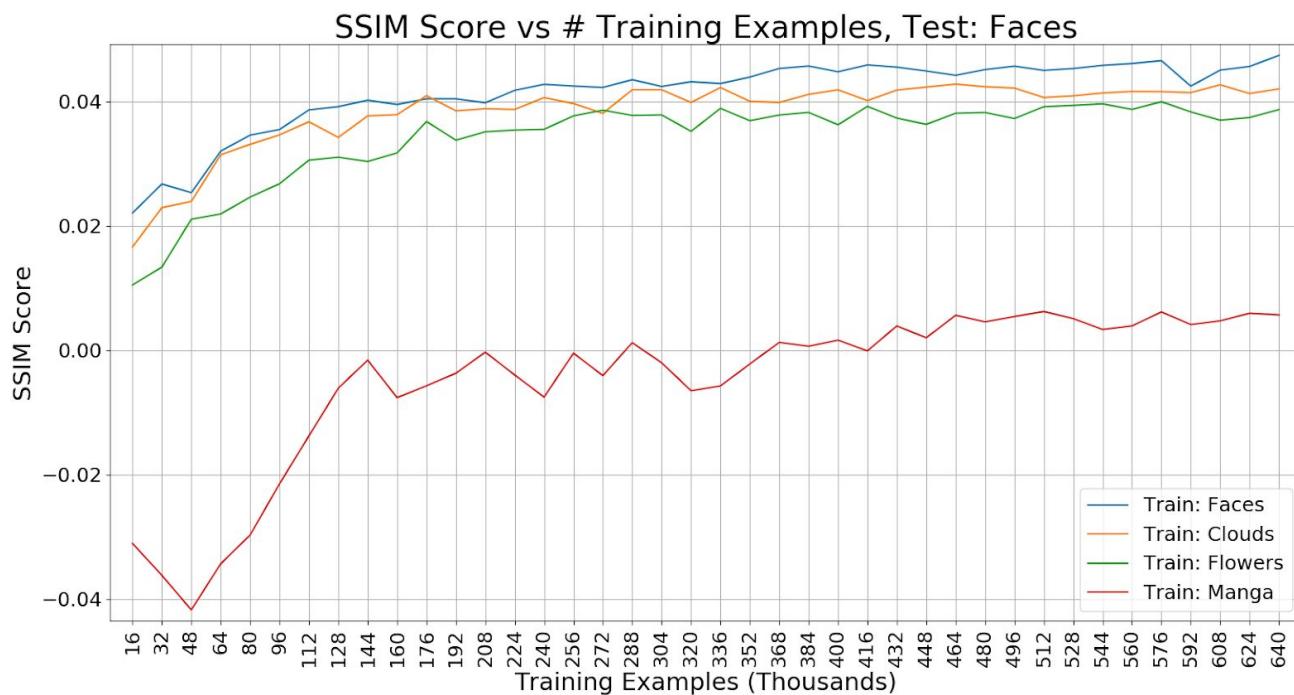


Figure F7 - Performance when tested on Faces

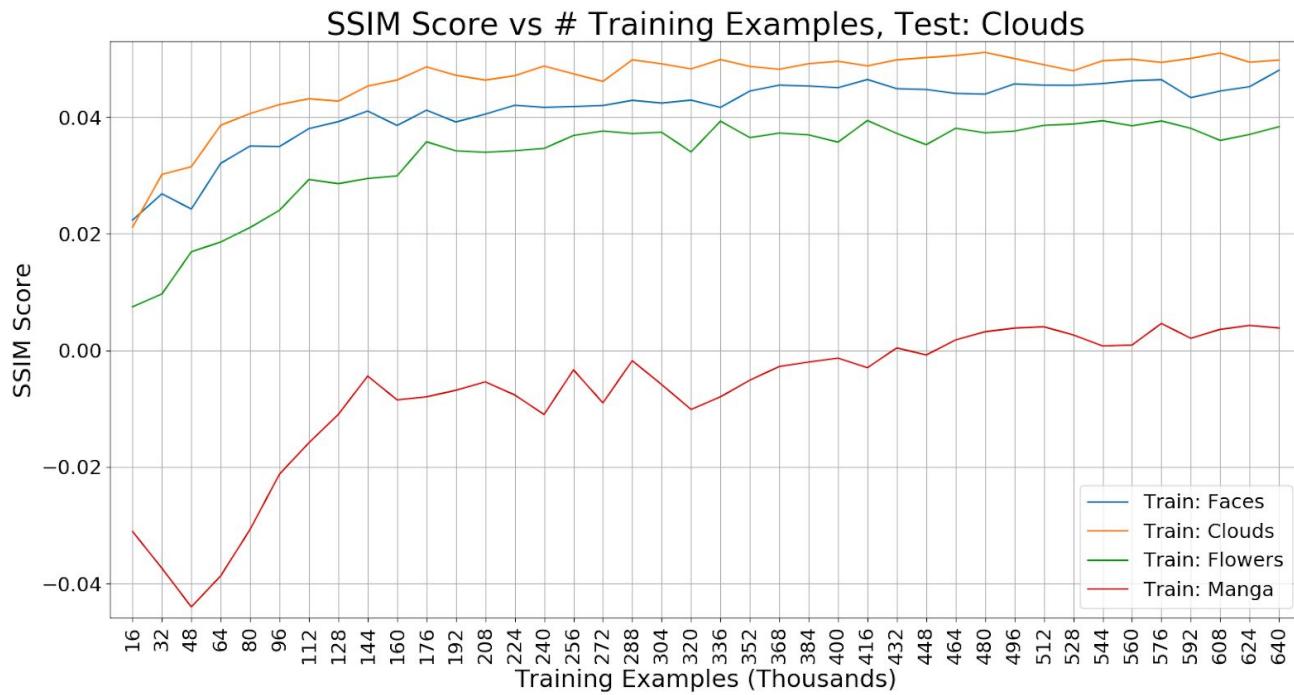


Figure F8 - Performance when tested on Clouds

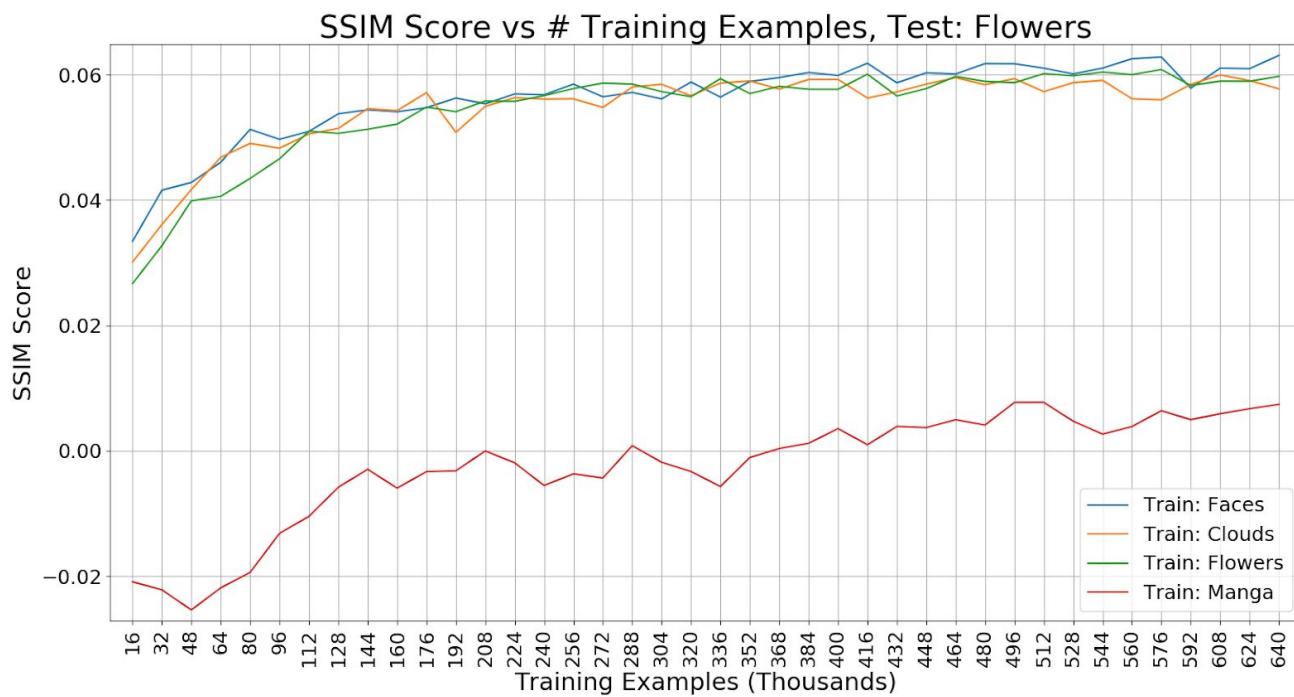


Figure F9 - Performance when tested on Flowers

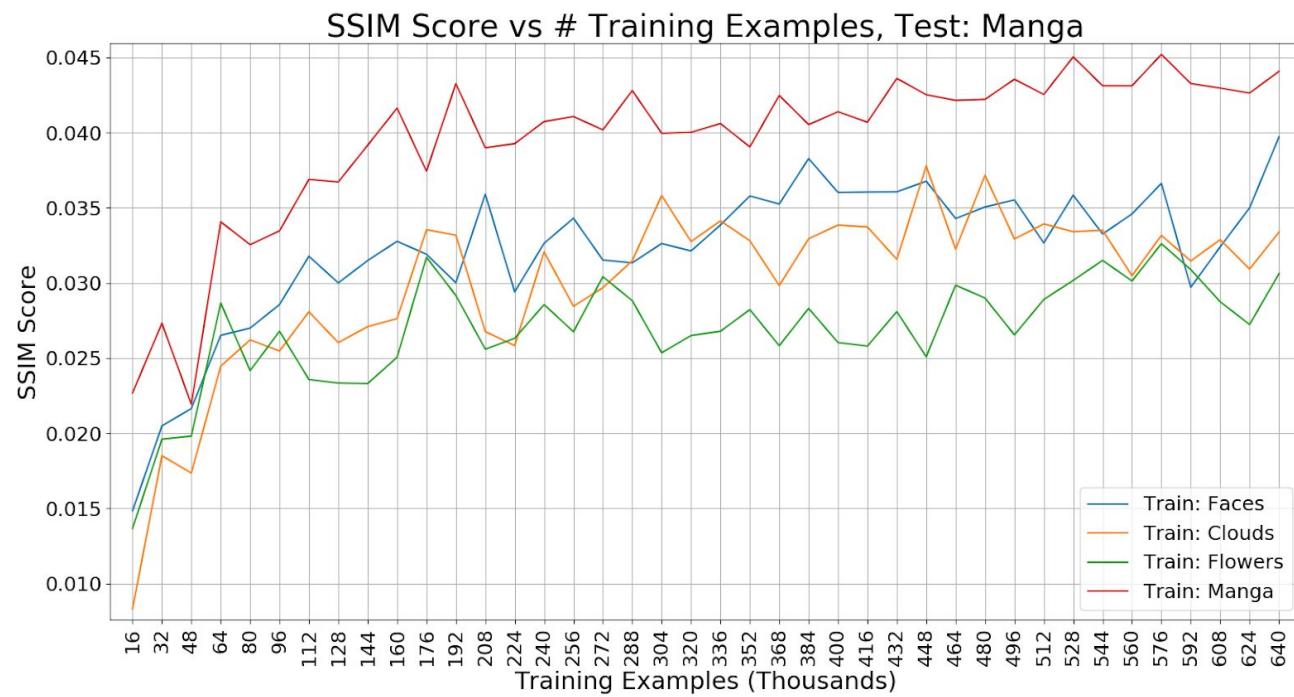


Figure F10 - Performance when tested on Manga

Observations:

- The model that performed the best on each test category was the model that was trained on the same image category, with the exception of Flowers. The model that performed the best on the Flowers test set was the model trained on Faces, though not by a large margin. With the exception of Flowers, these results indicate that training on a specific image category does indeed permit the model to learn features specific to that category, giving it an advantage in performance against models trained on other categories.

## F.4 Performance when Modifying the Model

This section details performance characteristics when the model was modified to contain more or fewer layers, thereby increasing or decreasing the complexity respectively.

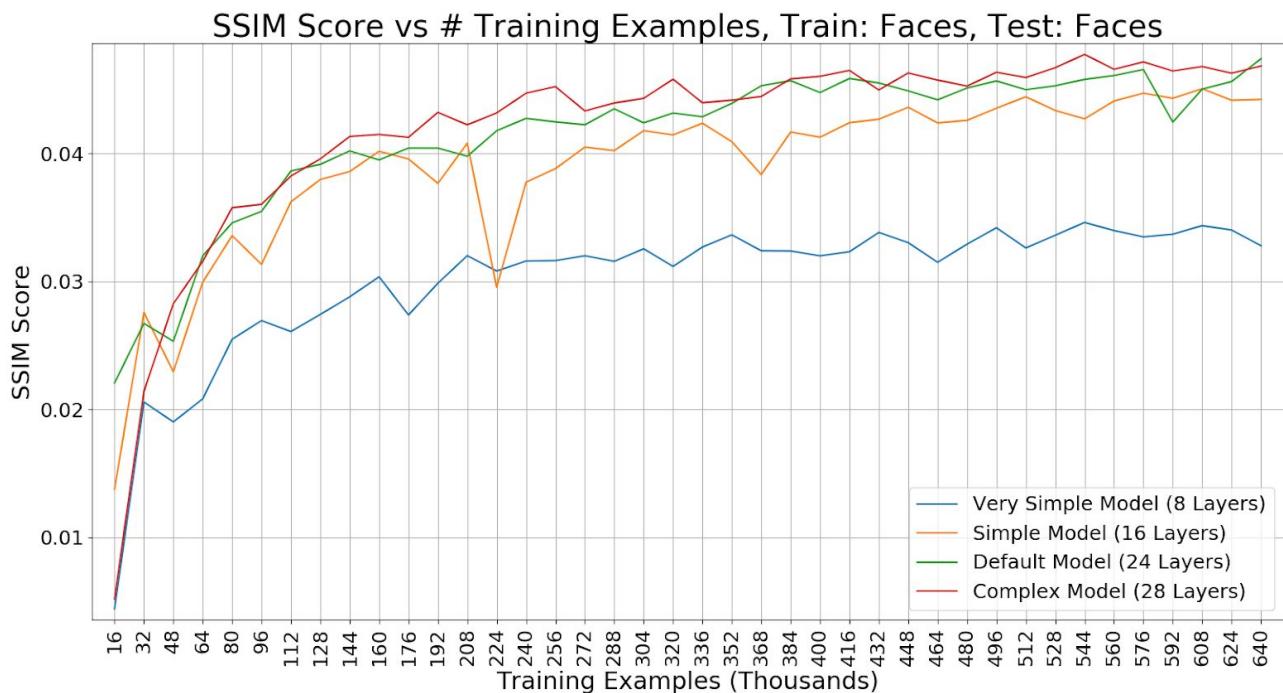


Figure F11 - Performance for various model complexities (trained and tested on Faces at 10% JPEG)

Observations:

- As expected, adding more layers improved the model performance, while removing layers degraded the performance. However, since the gains in performance were marginal between the Complex and Default models, we decided to use the Default model for all further tests to minimize computation time. Also worth noting: we were unable to test a model with more than 28 layers due to memory limitations on the ECF machines (see Section 3.3).

When comparing Machine Learning models with different internal structures, a common metric used is the number of Multiply-Accumulate operations (MACs) performed per pixel. Presented in Table F1 are the MACs per pixel for each of the 4 model complexities explored.

**Table F1 - Multiply-Accumulates per Pixel**

Model Variation	MACs per Pixel
Very Simple (8 Layers)	114,368
Simple (16 Layers)	262,208
Default (24 Layers)	410,048
Complex (28 Layers)	483,968

Table F2 below presents the change in MACs and SSIM Score for each of the model variations, compared to the baseline of the Default model. While the numbers may suggest the simpler models are worth sacrificing performance to gain lower computation time, our goal with the project was to generate high-quality Output Images, and the quality of images exhibited by the two simpler models was visibly inferior. As for the more complex model, we found the small increase in performance was not worth the significantly higher computation time required.

**Table F2 - Performance Changes versus Computational Overhead**

Model Variation	Change in MACs (versus Default model)	Change in SSIM Score after 640,000 training examples (versus Default model)
Very Simple (8 Layers)	-295,680 (-72.1%)	-0.0147 (-36.7%)
Simple (16 Layers)	-147,840 (-36.1%)	-0.0033 (-7.0%)
Default (24 Layers)	0 (0%)	0 (0%)
Complex (28 Layers)	+73,927 (+18.0%)	-0.0007 (-1.5%)

## F.5 Performance at Various JPEG Compression Levels

Presented below are performance evaluations of the model at various levels of JPEG compression. We were able to explore 10%, 5%, and 1% JPEG compression.

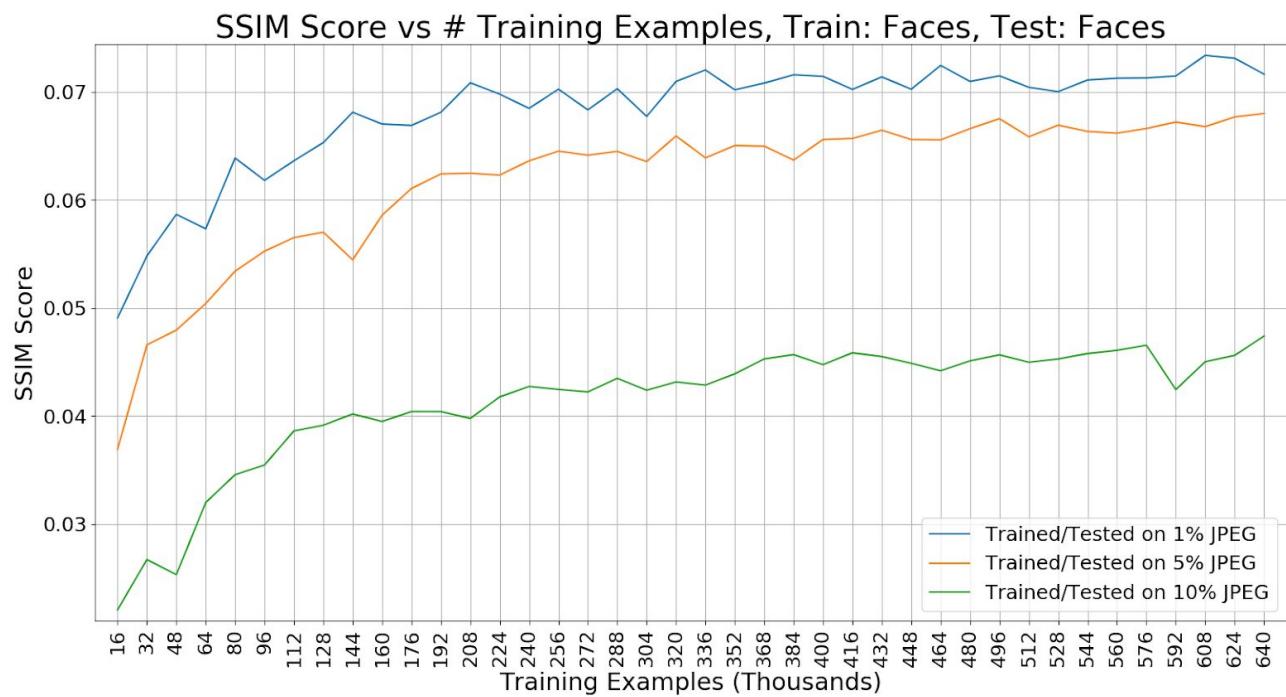


Figure F12 - Performance at 3 JPEG compression levels (trained and tested on Faces)

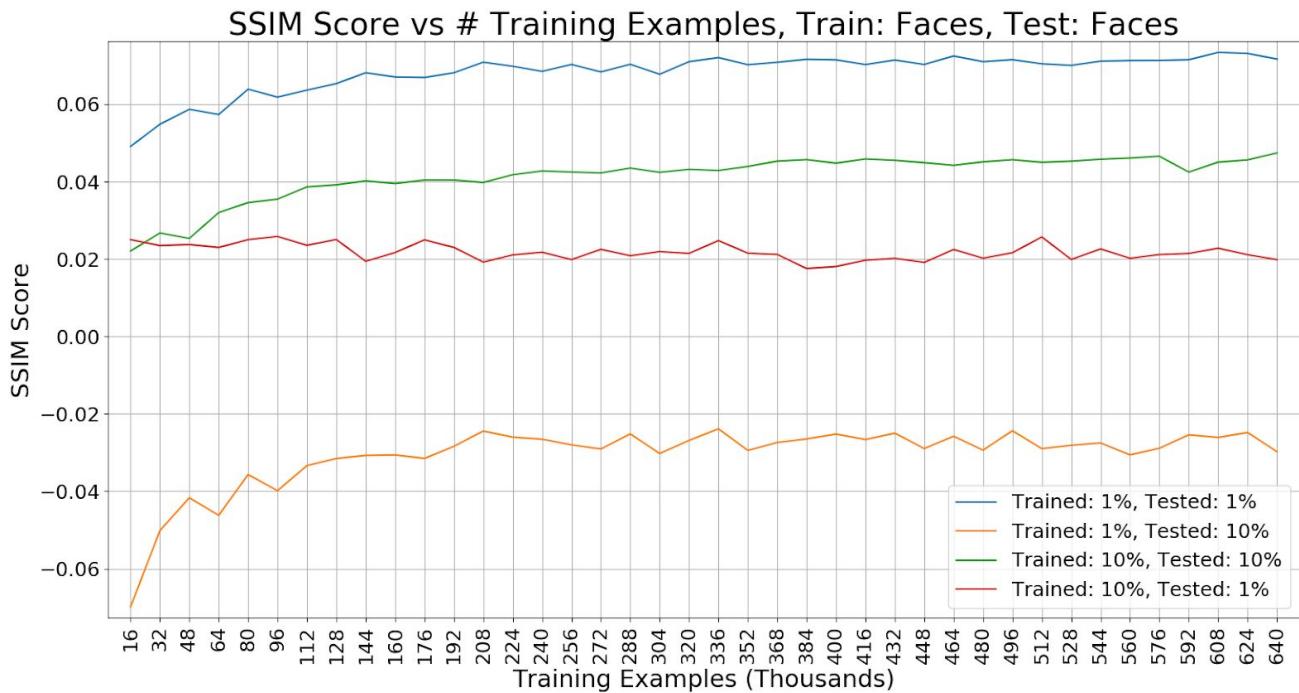


Figure F13 - Performance when training and testing on two different levels of JPEG compression  
(trained and tested on Faces)

## Observations:

- Higher levels of compression (corresponding to lower percentages) exhibit higher amounts of data regeneration. This is likely due to the fact that at higher compression, more data is discarded, so there is more potential for data to be regenerated.
- When the model is tested on a different level of JPEG compression than it was trained on, the performance is very poor, as seen in the red and orange plots in Figure F13. The orange plot (trained at 1% JPEG, tested at 10% JPEG) consistently has a negative SSIM Score, indicating the model deteriorated the image quality, which is a very undesirable result.

## F.6 Performance on Training versus Testing Data

Comparing the performance of the model on data that it has seen during training and on data that it has never seen allows us to gain an understanding of how much overfit there is in the model. Presented in Figure X14 below is the performance on the Faces dataset, against a subset of the training set and a separate testing set.

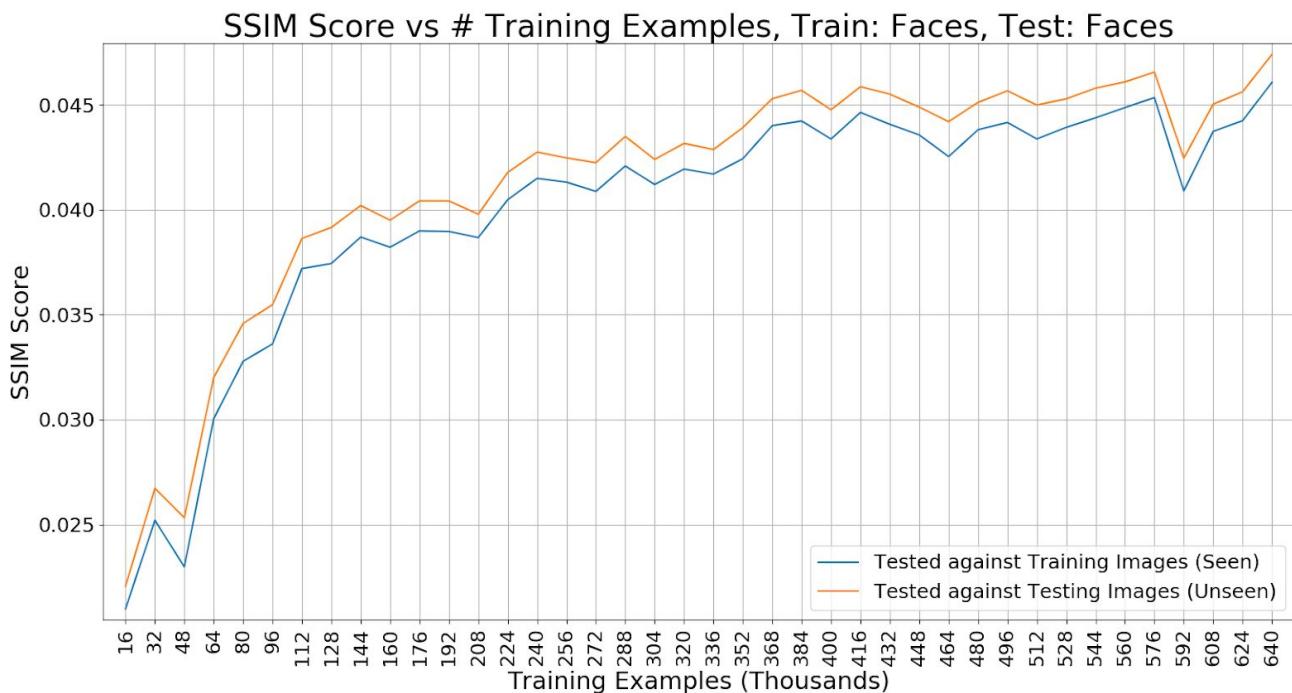


Figure F14 - SSIM Score vs training for seen vs unseen images

Observations:

- The performance is very similar, exhibiting the same micro-variations as the model trains on more images. This indicates a very low amount of overfit. Nonetheless, the performance is marginally higher on the previously-seen images than the unseen images.

## F.7 Samples with Good Performance

This section presents a few test images which performed well with the GAN model, when evaluated using the SSIM Score metric. All figures display from left-to-right, top-to-bottom: the Original Image, Input Image, and Output Image (as indicated by the image titles).

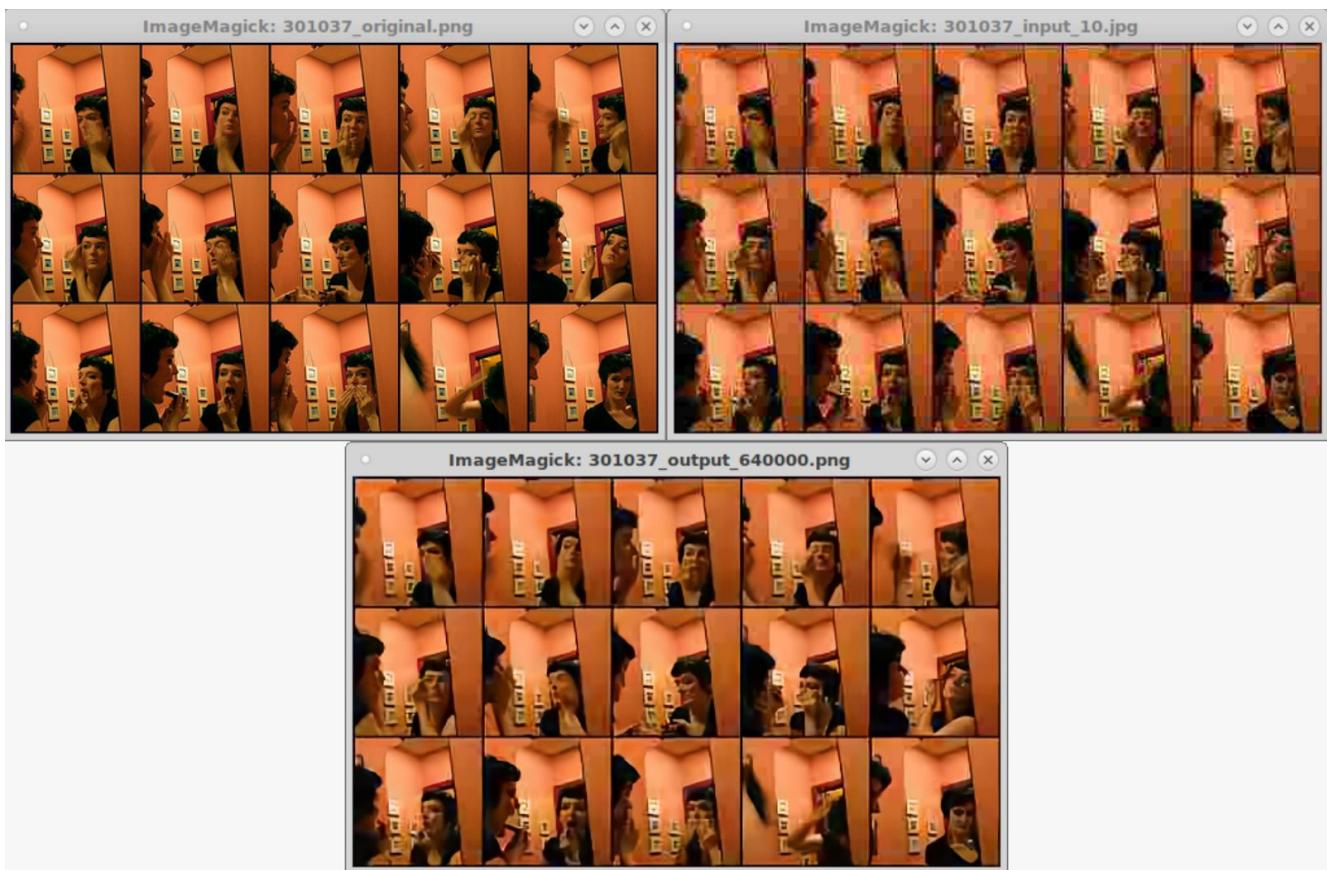


Figure F15 - Highest SSIM Score when trained/tested on Faces at 10% JPEG. SSIM Score: 0.126.



Figure F16 - Highest SSIM Score when trained/tested on Clouds at 10% JPEG. SSIM Score: 0.124.



Figure F17 - Highest SSIM Score when trained/tested on Flowers at 10% JPEG. SSIM Score: 0.162.



Figure F18 - Highest SSIM Score when trained/tested on Mixed at 10% JPEG. SSIM Score: 0.126.

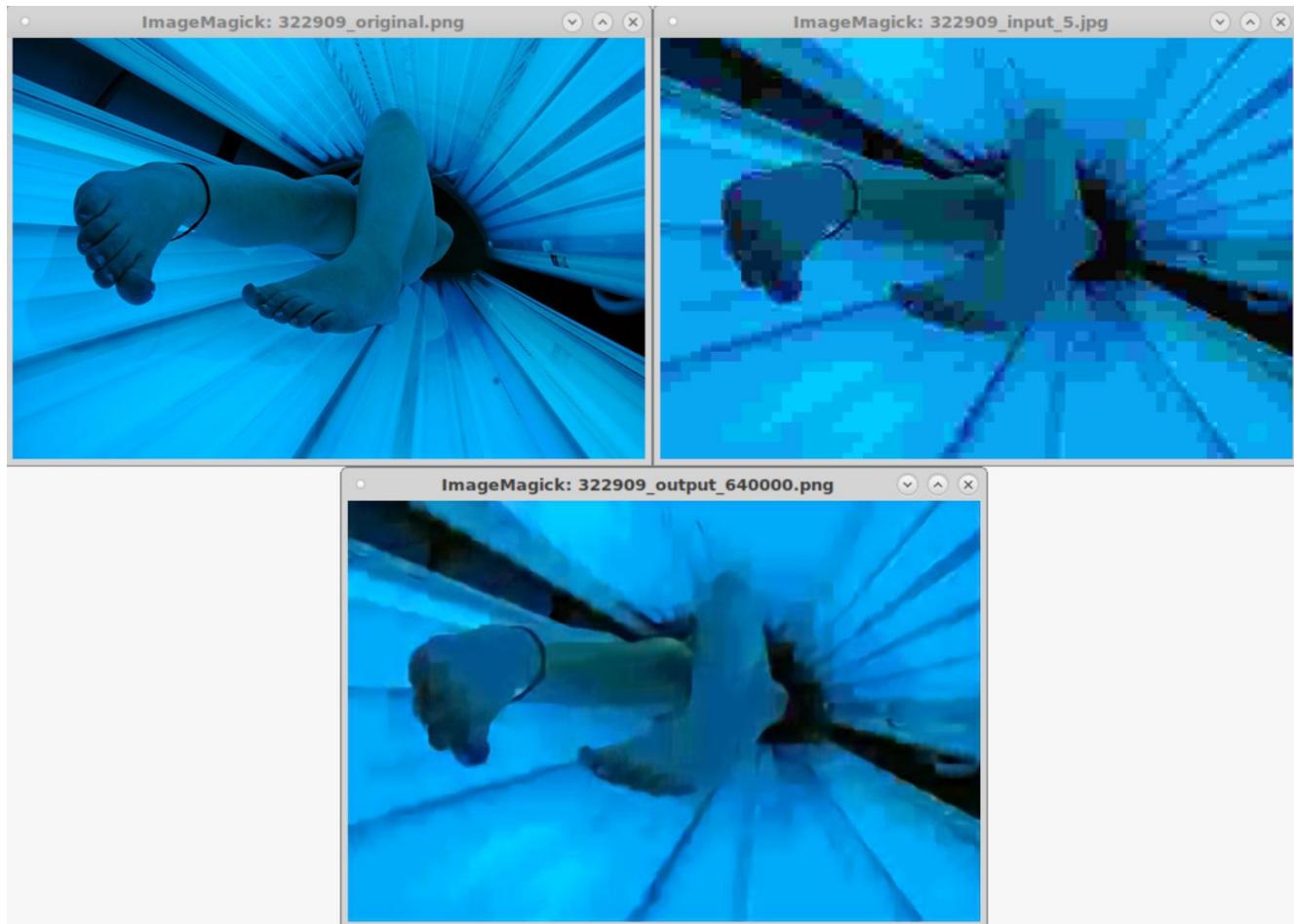


Figure F19 - Highest SSIM Score when trained/tested on Faces at 5% JPEG. SSIM Score: 0.163.

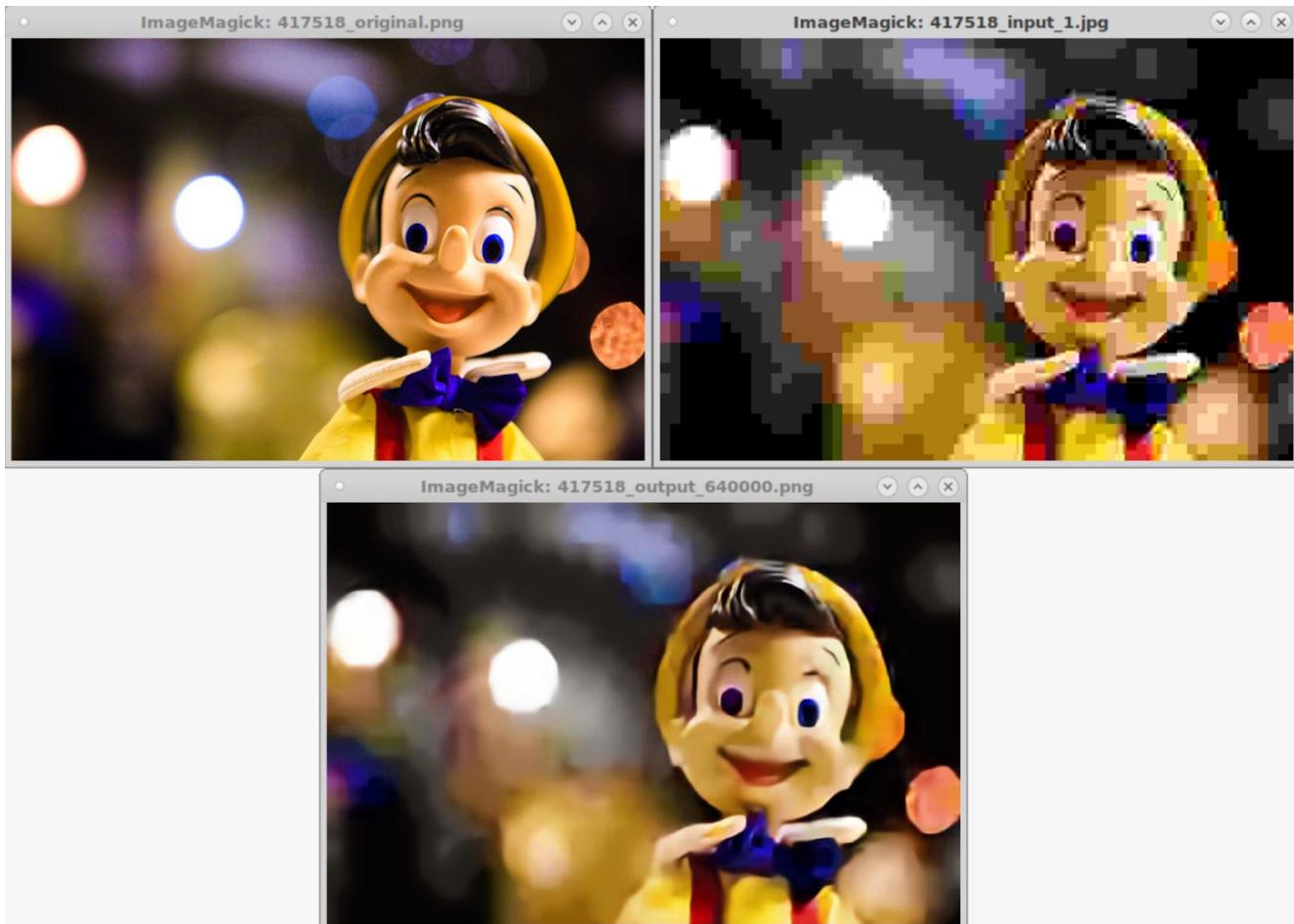


Figure F20 - Highest SSIM Score when trained/tested on Faces at 1% JPEG. SSIM Score: 0.225.

Observations:

- Most of the images have lots of flat colour tones, or gradual gradients, which do not lose much detail during JPEG compression. There are few jagged objects in the images. These properties result in good regeneration properties following JPEG compression.
- Some of the images are not good examples for their image category (such as Figure F19). Dataset pollution is discussed in Appendix E.

## F.8 Samples with Poor Performance

This section presents test images which performed very poorly with the GAN model, when evaluated using the SSIM Score metric. In all examples, the images had negative SSIM Scores, indicating the Compressed Image was further degraded by the model. All figures display from left-to-right, top-to-bottom: the Original Image, Input Image, and Output Image (as indicated by the image titles).



Figure F21 - Lowest SSIM Score when trained/tested on Faces at 10% JPEG. SSIM Score: -0.0914.



Figure F22 - Lowest SSIM Score when trained/tested on Clouds at 10% JPEG. SSIM Score: -0.0620.

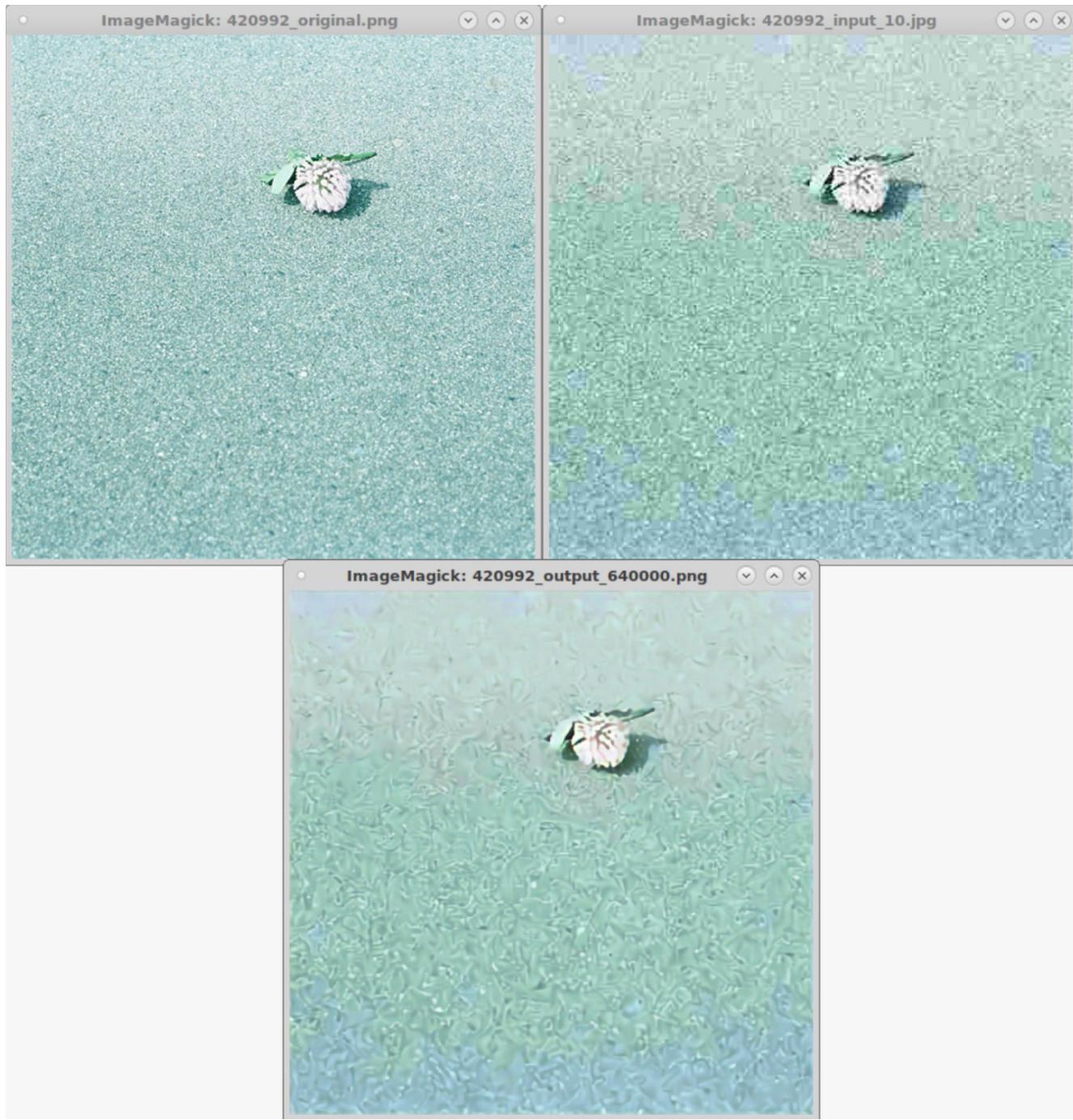


Figure F23 - Lowest SSIM Score when trained/tested on Flowers at 10% JPEG. SSIM Score: -0.0878.

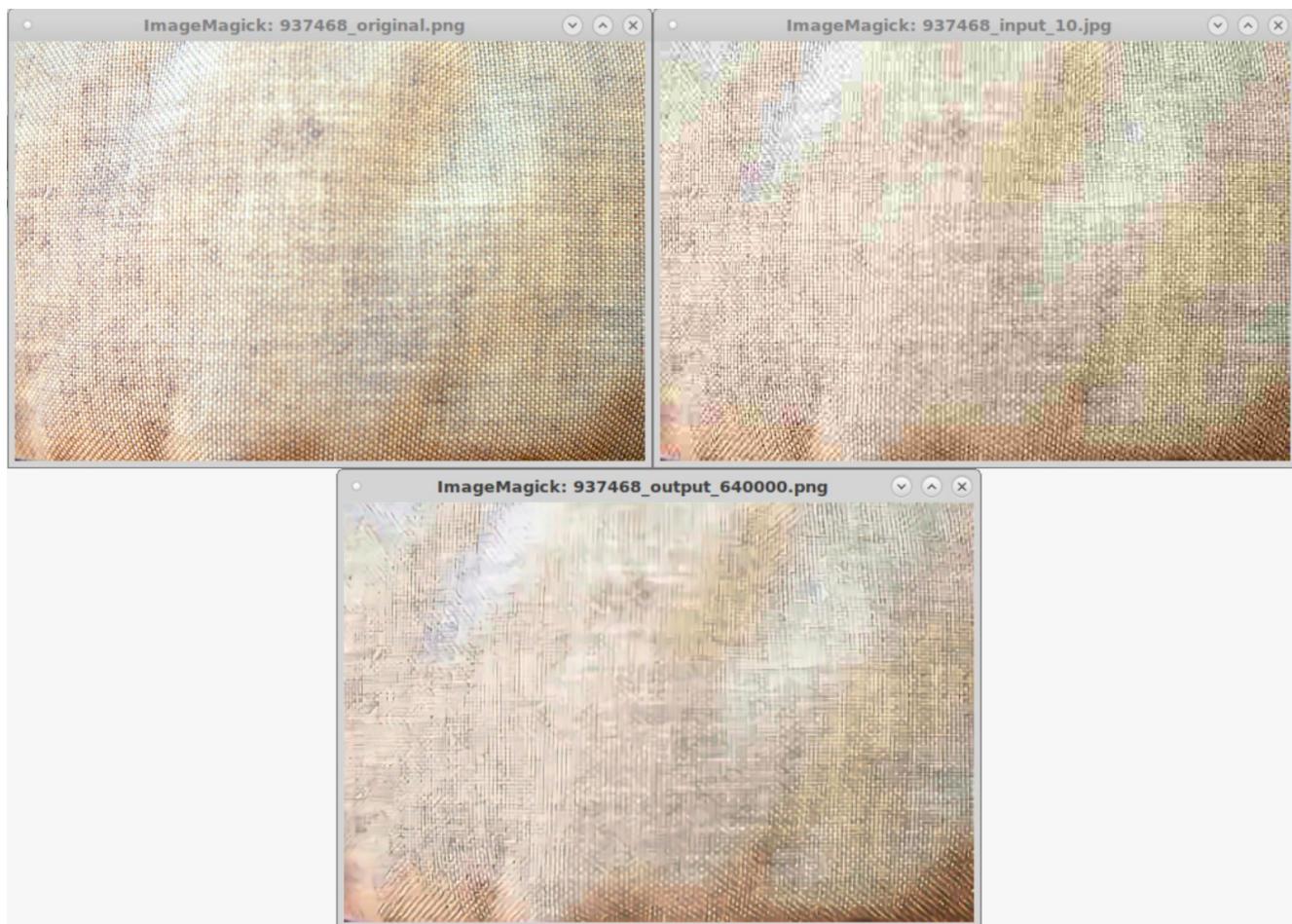


Figure F24 - Lowest SSIM Score when trained/tested on Mixed at 10% JPEG. SSIM Score: -0.125.

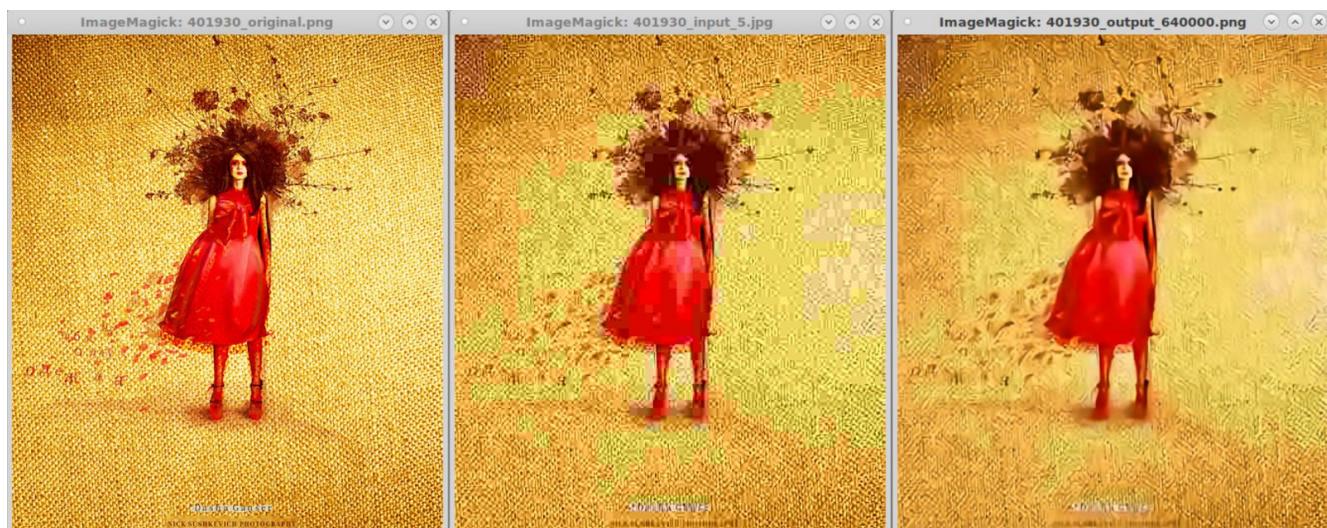


Figure F25 - Lowest SSIM Score when trained/tested on Faces at 5% JPEG. SSIM Score: -0.0856.

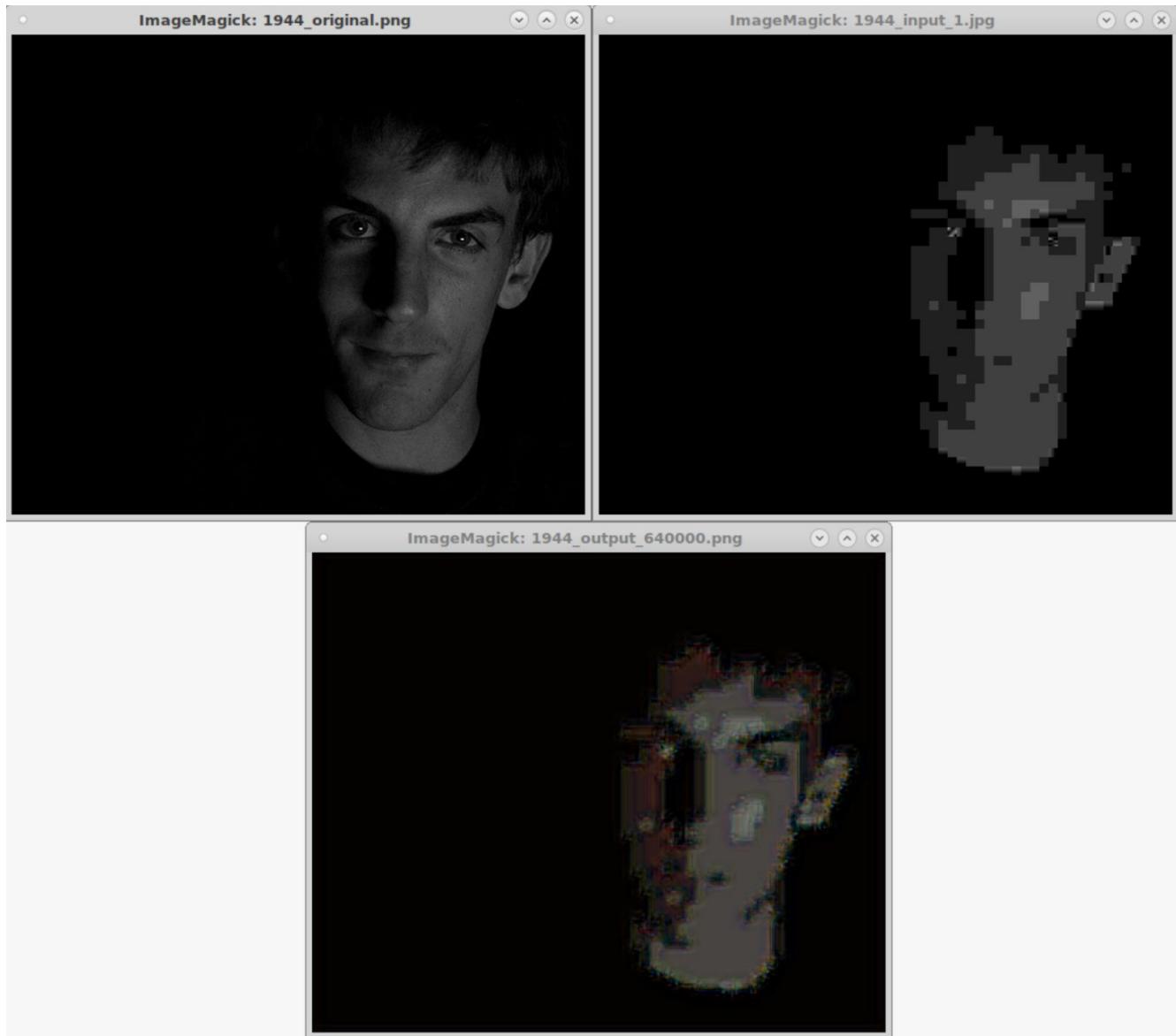


Figure F26 - Lowest SSIM Score when trained/tested on Faces at 1% JPEG. SSIM Score: -0.194.

#### Observations:

- Most of the images have fine-grained textures and/or noise, which gets extremely distorted when JPEG compression is applied. This results in very poor regeneration performance.
- Several images (Figures F22, F23, and F25) are poor samples for their respective image categories, serving as examples of pollution in the datasets (discussed in Appendix E). This potentially also contributes to poorer performance, since the models are specialized for a particular category in each of these cases.

## Appendix G: Results from CNN Model

Author: Zi Chien

This section presents the results generated by the CNN model.

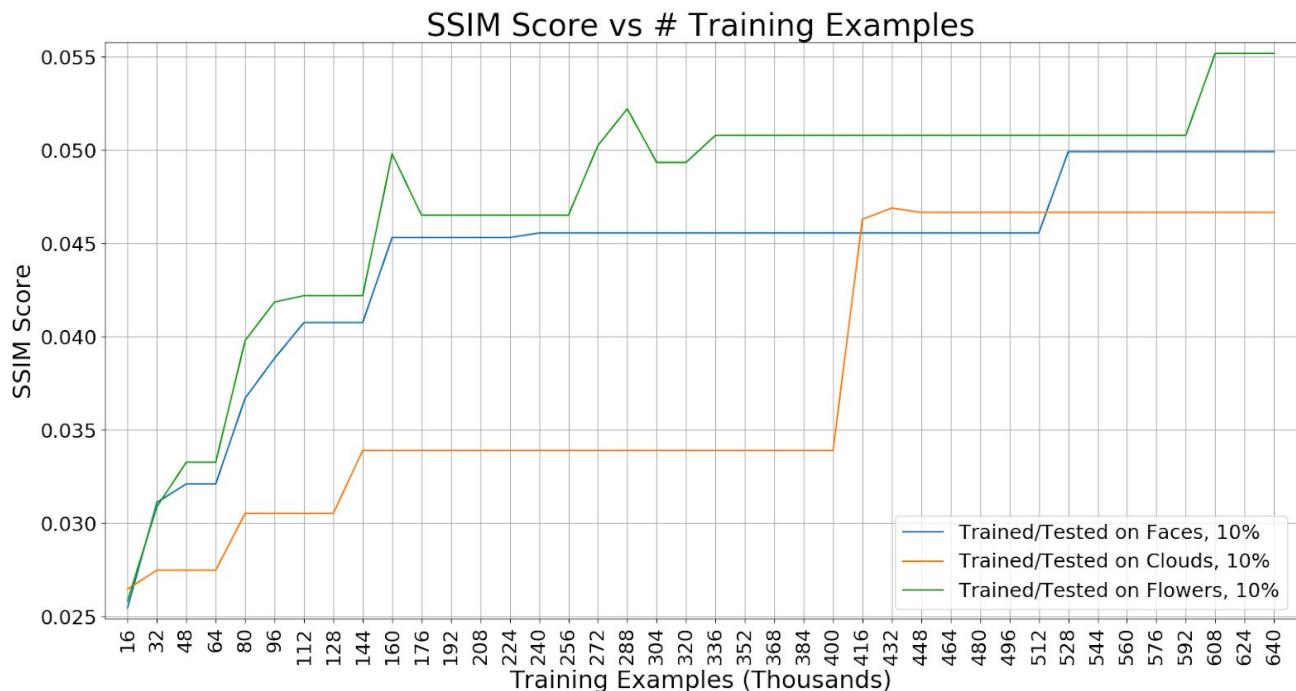


Figure G1 - Performance of CNN model in Standard Test Conditions

Observations:

- The model starts with a similar SSIM Score after 16,000 training examples regardless of image category, and improves in performance at different rates for different datasets. The final SSIM Scores after 640,000 training examples is around 0.05, and varies by dataset.

## Appendix H: Comparing Results (GAN vs CNN)

Author: Zi Chien

This section compares the performance of the GAN and CNN under select test conditions.

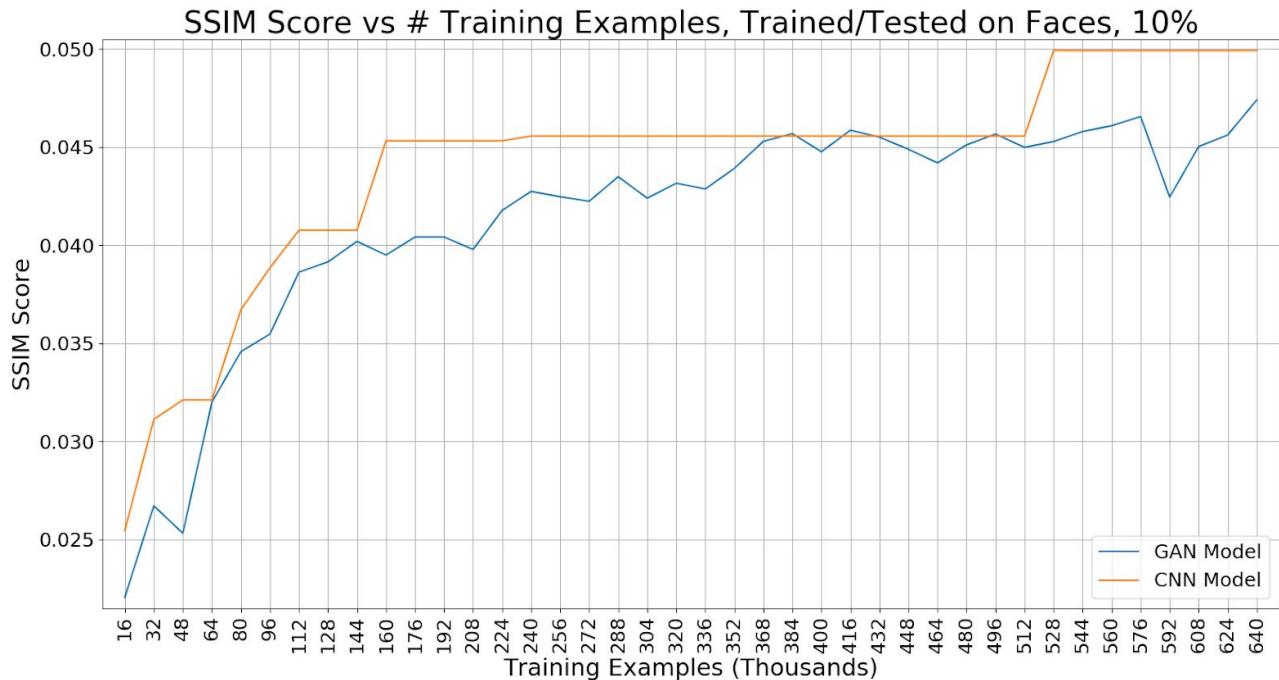


Figure H1 - Performance of GAN and CNN on the Faces dataset

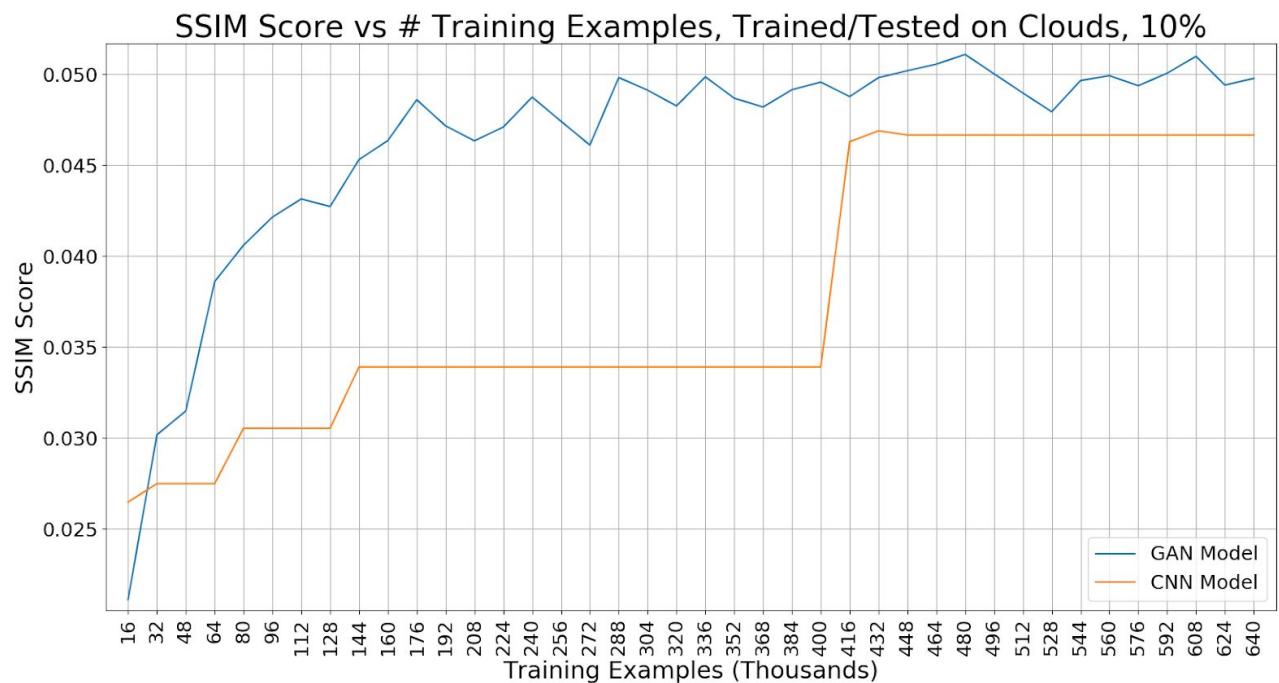


Figure H2 - Performance of GAN and CNN on the Clouds dataset

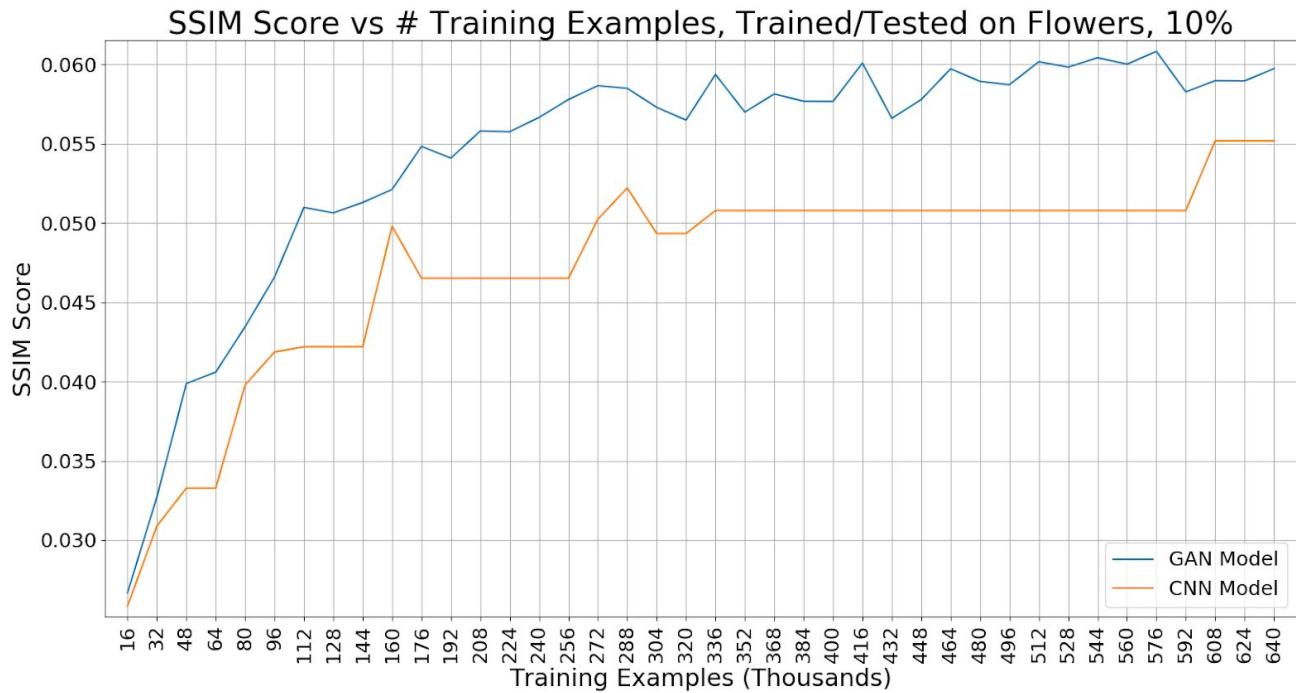


Figure H3 - Performance of GAN and CNN on the Flowers dataset

#### Observations:

- The GAN has better performance in two of the three test datasets. Furthermore, we found the GAN had lower resource requirements on the ECF machines, and was easier to use and work with, which is why it was selected as the final model for the project.

## Appendix J: Existing Regeneration Technology

Author: Henry

As part of evaluating the performance of our Machine Learning solution to the problem, we examined the performance of existing generic image processing tools. Specifically, we applied the Smart Sharpen filter in Adobe Photoshop CC 2018. Although this tool was not specifically designed to reverse JPEG compression loss (we were unable to find any such tool), we feel it was the most appropriate popular consumer tool we could find for the task. Table J1 below presents a few key performance results for this method, when tested on 50 random test images at 10% JPEG quality. Figure J1 below the table shows a histogram of the SSIM Scores for the 50 test images.

**Table J1 - Performance of Adobe Smart Sharpen**

Performance Characteristic	Value
Average SSIM Score	-0.0625
Variance of SSIM Score	0.00145
Minimum SSIM Score	-0.175
Maximum SSIM Score	-0.00475

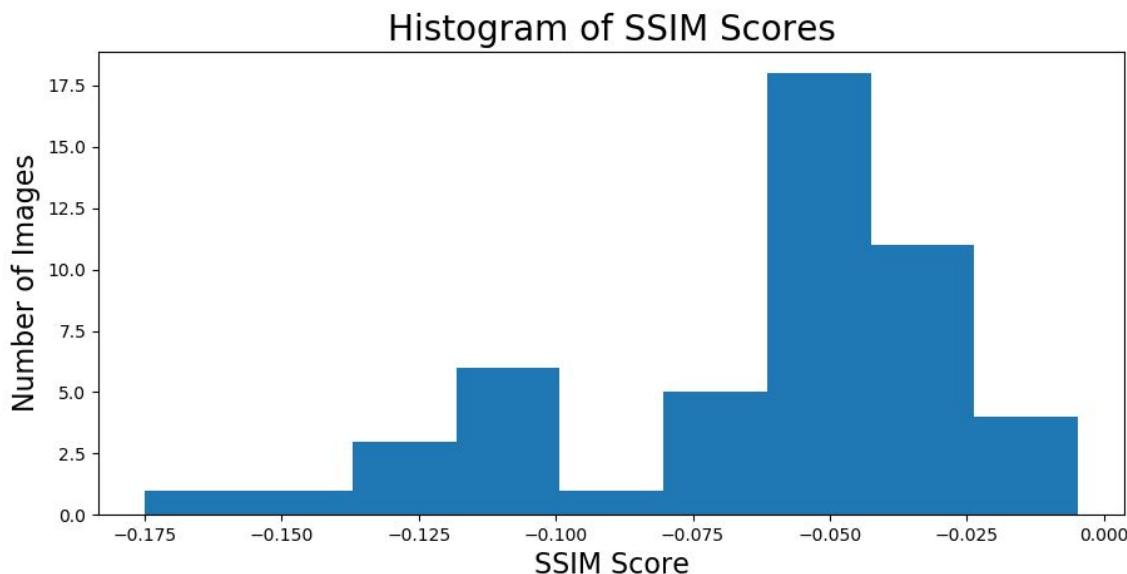


Figure J1 - Histogram of SSIM Scores for 50 test images

Surprisingly, all 50 test images had negative SSIM Scores after using the Adobe Smart Sharpen tool, indicating they were degraded further. These results are clearly inferior to the results we are able to achieve with our Machine Learning approach.

For a visual comparison, the images with the lowest and highest SSIM Scores are presented in Figures J2 and J3 below. In both figures, the 10% JPEG Input Image is on the left, and the Output Image produced by the Adobe Smart Sharpen tool is on the right.

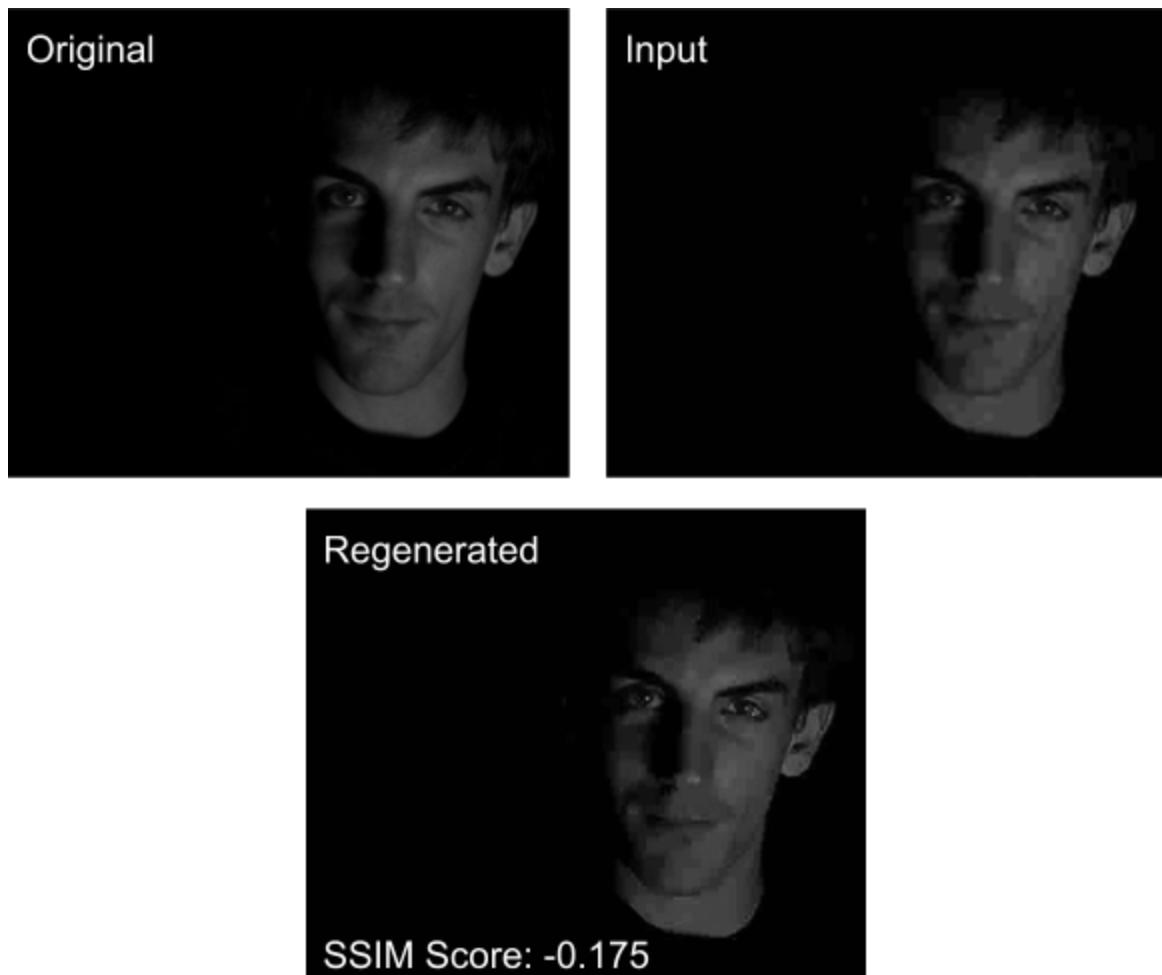


Figure J2 - Test image with lowest SSIM Score



Figure J3 - Test image with highest SSIM Score

# Appendix K: Glossary

Author: Henry

The table below contains some key terminology and definitions pertaining to the project.

**Table K1 - Key Terminology**

Term	Definition
<b>Convolutional neural network (CNN)</b>	Set of deep neural networks that is successful in analyzing visual imagery. It improves the cost of the network by the reuse of weights in its sliding windows.[10]
<b>Data Recovery</b>	Salvaging corrupted data and reverting it back to its original form.
<b>Deprecated</b>	The discouragement of use of features and practices that have been superseded or is no longer considered efficient or safe but is still available.
<b>Generative Adversarial Network (GAN)</b>	Deep neural net architectures comprised of two nets, pitting one against the other (thus the “adversarial”). [10]
<b>GPU</b>	Specialized electronic circuit designed to accelerate the creation of images, their highly parallel structure makes them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data is done in parallel. Modern GPUs are very efficient at manipulating computer graphics, image processing and subsequently training ML models.
<b>High-Level</b>	Operations that are more abstract in nature, where overall goals and systemic features are typically more concerned with the wider, macro system as a whole.
<b>Image Denoising</b>	Minimizing random variations in colour or brightness in pixels in an image.
<b>Image Regeneration</b>	Using an ML model to attempt to restore an image to its original state after it has undergone lossy JPEG compression.
<b>Input image</b>	An image that the model transforms according to parameters it learned from the target image.
<b>Lossless Compression</b>	Class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data.
<b>Lossy Compression</b>	Class of data encoding methods that uses inexact approximations and partial data discarding to represent the content.

<b>Machine Learning Model</b>	This term refers to the model artifact that is created by the training process.
<b>Neural network</b>	An information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. [11]
<b>Output image</b>	An image transformed by the model according to the parameters learned from the target image.
<b>Overfit</b>	A modeling error which occurs when a function is too closely fit to a limited set of data points. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study. [12]
<b>SSIM (SSIM Value)</b>	(SSIM) index is a method for predicting the perceived quality of digital images and videos. A SSIM values range from 0 to 1 with values close to 1 representing more similarities.
<b>SSIM Score</b>	A metric devised for the purposes of this project. It is the resultant between the SSIM Value of the (Original, Input) image pair subtracted from the SSIM Value of the (Original, Generated) image pair. If the SSIM Score is less than 0, then the model has further degraded the Input image. If the SSIM Score is greater than 0, then the model has regenerated some of the data lost in compression.
<b>Standard Test Conditions</b>	Test cases where the model is tested on the same image category that it was trained on.
<b>Super Resolution</b>	Class of techniques that increases the resolution of an image.
<b>Target image</b>	An image that the model “learns” in order to produce an output image.
<b>Testing Dataset</b>	A dataset of examples used only to assess the performance (i.e. generalization) of a fully specified classifier. [6]
<b>Training Dataset</b>	A dataset of examples used for learning, that is to fit the parameters (e.g., weights) of, for example, a classifier. [6]
<b>Underfit</b>	A modeling error where the model is not adequately complex to learn the intricacies of the relationships in the data being trained on. Can also happen if the amount of training is insufficient.