University of Exeter
College of Engineering, Mathematics and Physical Sciences

Final Report

Research Project

Interactive Image Generation

Abstract

Machine Learning techniques can be used to automatically extract what are the elements that characterize a set of preferred items. In this project a user will start from a collection of selected images and will be able to interact with an image generator. The system will consider the user feedback as to which part of the image needs to be changed. The use of Graph Theory is used to represent images into graphs and Computer Vision to change the image pixels to create the final image. In this project we will 1) use a dataset that prevents us from developing segmentation techniques to automatically identify relevant elements in images, 2) convert images to networks of connected elements, 3) employ graph grammars (F. Costa, Learning an efficient constructive sampler for graphs. Artificial Intelligence 2017) and graph kernel machine learning techniques (F. Costa, K. De Grave. "Fast neighbourhood subgraph pairwise distance kernel." ICML 2010) to induce the generative rules and inform an optimization algorithm on which actions to perform to generate a novel image. [1]

We certify that all material in this dissertation which is not our own work has been identified

Submitted for the formal marking as part of ECMM428, Individual Research Project, August 2020

Contents

1 Introduction

The field of generating images with human interaction has seen exciting development. The most prominent in recent years is creating new image from scene-based text descriptions . Such methodologies have concentrated mainly on producing photos from a summary of a static text and are restricted to creating images in one step. We can not create a picture interactively based on a definition of incrementally additive text (something that is more natural and close to the way we interpret a picture). This project suggests a method for producing an image progressively based on a series of scene analysis (scene-graphs) with the aid of Graph Theory, Machine Learning Algorithm and Computer Vision.

Humans can easily rate images based on their personal preferences (this could be influenced by how the image looks i.e. the colour, objects in the image, etc.). However, for a computer to precisely extract the reasons for those personal choices is perplexing. To extract the elements that make up the perfect image, many generative systems and Machine Learning techniques have been employed. This involves considering the preferred elements in the preferred combination to make that perfect image. This project will involve a user interacting with a collection of set images initially on the image generator system, to create new but custom-made images. The imaging system will look at the user's feedback (what they would like to keep and what they would like to remove) and will continuously generate complicated yet pleasant images [1]. To make this possible element combinations have to be represented in a form of graph or network of nodes and edges (using graph theory). The imaging system will look at the user's (a predefined set of preferences) feedback for adding,removing or replacing nodes or edges to continuously generate novel scene graphs which in turn is a new image.

This novel way of generating images involves the use of Computer Science topics like Graph Theory to represent and analyse objects and the relationship between objects in a scene, Machine Learning to classify each data point in the given dataset, and Computer Vision for Object Classification (categorises objects found in an image), Object Segmentation (determines what pixel makes up an object), and Image Inpainting.

1.1 Motivation

There are quite a few renowned methods of generating new images with user interaction. These methods mostly involve working on high level of image manipulation. For example, Photoshop involves changing pixels of an image with operation like cropping,pasting,copying, blurring,etc. However there are very little methods that involve working on a lower level using graphs to represent images and working on them.

In addition, many groundbreaking advances in the field of visualisation from scene-based text representations have been observed in recent years. Such methods primarily rely on the construction of images from a static text definition and on the processing of pictures in a single paragraph. They can not produce an image on the basis of an intuitive text description. This project suggests a way to construct an image slowly based on a visual series of region descriptions (a scene description).

Another motivation involves using Graphs and Machine Learning. Additional information renders models of machine learning far more predictive. But getting all the data required to feel confident in a training model is a challenge. Relationships are among the strongest behavioural predictors. This project will add these highly predictive elements to improve machine learning by incorporating relationship information and also apply the graph functionality to an existing machine learning workflow and enhance existing processes [2].

1.2 Research Problem

This project takes the form of research, where an attempt to answer a simple but important question in interactive image generation: **Can graph theory and machine learning generate novel images?** In other words, how can machine learning techniques and graph theory improve the way images are represented and changed.

1.3 Report Outline

This document is a concise report, detailing the scope and complexity of the system produced, its specification and design, the research procedure, its sustainability and its use. The main body of the document consist of the following topics:

1. The context of the system, i.e. what scientific question is it intended to help the problem owner answer. Also description of similar projects compared to this work.
2. A description of the final system product, including its components, relationships and functionality. It should then present an overview of the system specification, highlighting any changes to/omissions from/refinements of those detailed in the prototype report. Any departures from the revised requirements at the prototype stage should be well-justified.
3. A description of the research process used, including points where critical decisions were made, including justifications.
4. A description of the testing (for code correctness, functionality, etc.) that has been conducted on the final system solution and during its construction.
5. A description of any experimentation/analysis and results regarding the performance/suitability of the system produced to tackle the scientific problem (image generation based on preference).
6. The result of final system and a good solution to the original problem, any departures from the original requirements or limitation have been well justified.
7. The conclusion sums up the document and introduces future work on this project.

2 Scope and Complexity

2.1 Related Work

1. iGAN: Interactive Image Generation via Generative Adversarial Networks (by Jun-Yan Zhu) - Generate photo-realistic samples that best satisfy user editing in real-time, provided a few user strokes. Their project is based on deep generative models like the Generative Adversarial Networks (GAN) and DCGAN. The device performs two functions [3]:

    - An advanced sketching interface to create images influenced by the colour and form of the brush strokes automatically.
    - An immersive visual debugging platform designed to understand and imagine deep generative modeling. By engaging with the generative model, a developer can understand whether the model will generate visual content, as well as the model's limit.

2. Interactive Image Generation using Scene Graphs (by Gaurav M., S. Agrawal, Anuva A., Sushant M., T. Marwah) - A theoretical approach for incrementally producing an picture centred on a collection of graphs from observations of a scene (scene-graphs). A recurrent network architecture which preserves the image content generated in previous steps and modifies the cumulative image according to the newly given information on the scene. In conjunction with the Generative Adversarial image translation networks, the model uses Graph Convolutional Networks (GCN) to care for vector scene graphs to produce practical multi-object images without the requirement for intermediate supervision during training. [4].

3. SG2IM: Image Generation from Scene Graphs (by Justin Johnson, Agrim Gupta, Li Fei-Fei) - Recent groundbreaking developments have been made in producing photographs from accounts of natural languages. Such methods offer impressive results on specific domains such as descriptions of birds or flowers, but fail with other artefacts and relationships to faithfully replicate complex phrases. We suggest a method for producing images from scene graphs to solve this constraint, allowing clear reasoning about objects and their relationships. The model uses graph convolution to process input graphs, calculates a scene layout by predicting object bounding boxes and segmentation masks and transforms the layout into an image with a cascaded network of refinements. The network is subjected to a couple of discriminators in order to ensure practical outputs [5].

2.2 Objectives

We adopted an experimental approach in the undertaking of this project. In this project the aim is to research, implement and create a way, which uses graph theory in order to generate new image in response to the project proposals.

- First objective is to develop segmentation techniques to automatically identify relevant elements in images. This is necessary as each element of the image is used for the next objective.
- Second objective is to convert images to networks of connected elements.
- Third objective employ graph grammars and graph kernel machine learning techniques to induce the generative rules and inform an optimisation algorithm on which actions are to be performed to generate a novel image.

3 Design and Specification

In this section we will briefly highlight important aspects of the design with the aid of images. The research process will entail; a more in-depth explanation, the specification from the Literature review and discuss any deviation from it.

3.1 Project Design

As mentioned in the section 2.2, the project designs are tailored to the three objectives i.e develop segmentation techniques to automatically identify relevant elements in images, convert images to networks of connected elements, employ graph grammars and graph kernel machine learning techniques.

3.1.1 Develop Segmentation Techniques

Semantic segmentation is a high-resolution embodiment of computer vision problems, others include human pose estimation, object detection, and so on. The aim of segmentation technique is to automatically identify relevant elements in images by assigning each pixel in an image/scene to category label. There were two segmentation techniques that we have originally looked closely at in the literature review to aid decision to implement in this project. Pyramid Scene Parsing Network (PSPNET) and Fully Convolutional Network (FCN). However, after implementing Detectron2 API segmentation technique (Detectron2 is the next generation software system from Facebook AI Research, which implements state-of-the-art object detection algorithms [6].) (see Figure 3.1) we realised that the dataset used in this project Visual Genome already collected annotations of objects and attributes in an image but it also uses them to form relationships between these objects and their attributes as well as creating several region graphs showing how objects interact with each other. Another major reason why we decided to go with the Visual Genome (V.G.) dataset is the fact that it offers 1,343,636 more objects compared to the next best thing MS-COCO dataset [7]. Following from this, we have already established that each object has at least one attribute and a relationship is how two or more objects interact in a scene, therefore, there is bound to be more attributes and relationships in the Visual Genome dataset. This is vital as it aids in making a bigger and more comprehensive scene graph with more region graphs.

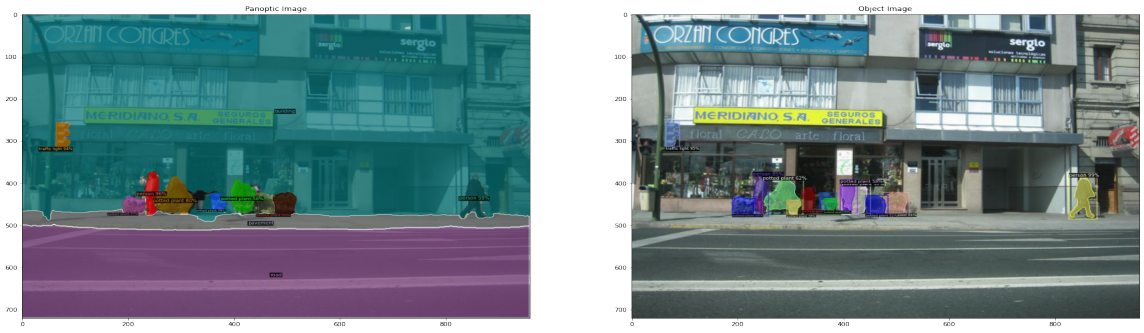


Figure 3.1: An example of a Panoptic and object segmentation on Visual Genome dataset image [7].

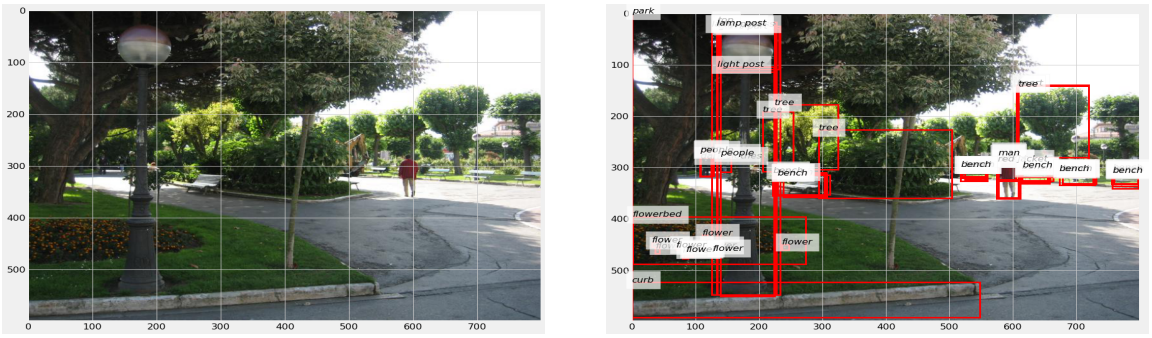
Visual Genome dataset has 7 important parts that makes for a logical image segmentation: Multiple regions and their descriptions, multiple objects and their bounding box, a set of attributes, a set of relationships, a set of region graphs, and one scene graph or image [7].

Region Description - A collection of descriptions on numerous regions of a scene can be used to improve this problem. In Visual Genome, there is a collection of different human-made image region descriptions, and these areas are restricted by a boundary box. Regions overlap if the descriptions are different. The Visual Genome dataset has a mean total of fifty region descriptions per image, and these descriptions have a phrase length extending from 1 to 16 words describing that area.

Objects - An Object is an element that makes of an image. For example a woman, a tree, the road. The mean number of objects in the dataset is 35, and each portrayed by a small bounding box see Figure 3.2. Each object is legally added to a sysnet ID in WordNet [8]. An example would be the word "woman" (it is commonly used to identify a female human being). Likewise, a "girl" is also used to refer to a female human being. In WordNet, these two words are connected in such a way that it is clear that they both refer to a human being (mainly female in gender). This is done to prevent a lot of labels for one object, and to link information throughout images.

Attributes - Objects are made up of some or no attributes affiliated with them. These attributes could be stated (e.g. sleeping), colour (e.g. red), e.t.c. In each image in the Visual Genome, there is roughly a mean of 26 attributes. This dataset gathers attributes that are attached to objects that have been collected from the area of interest (region description). The attributes are for each object is legally add to WordNet [8]. With the phrase "woman standing", we can concentrate on the object "woman". We know that one of the many attributes of an object is its state, in this case, we can conclude that the state "standing" is connected as an attribute to the object "woman".

Relationships - This is one of the most crucial parts of moving forward in this project because it connects two objects. Relationships between two objects highlight how they interact with each other i.e. via actions (e.g. running, walking), preposition (e.g. under, over), comparative (e.g. bigger than, worse than), descriptive verb (e.g. looking, holding), or spatial (e.g. in front of, next to).



(a) An example for a normal image in the Visual Genome dataset

(b) An example for the segmented and label elements in the same image

Figure 3.2: These images are from the Visual Genome dataset to aid in better understanding of objects, attributes and region description with their respective bounding boxes [7].

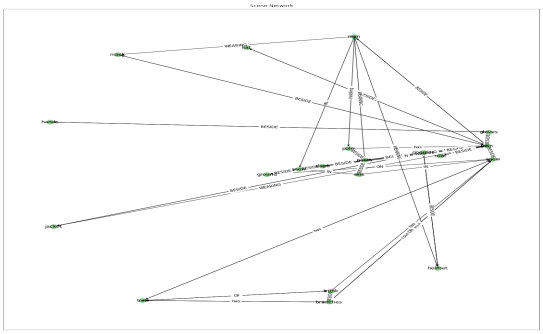
3.1.2 Convert Images into networks of connected edges

In this stage of the design we have already acquired the necessary tools to make a region graphs and in turn a scene graph with the aid of graph theory implemented using networkx - a Python programme used for the development, manipulation and analysis of complex network structure, dynamics and functions.

Region Graphs - A region graph is a directed graph. It is an established representation of a section of the image that is created by joining the nodes (attributes, objects, and relationships) together from a region description for each area of interest, where objects are connected to their corresponding attributes and the relationships are what connects objects. It also shows an example with the region description "man and woman sit on a park bench along a river" contains the objects: woman, man, bench and river. The relationships: "sit on", and "in front of". The combination of these makes a region graph.

Scene Graph - A scene graph is the combination of local region graphs together to represent an entire scene in an image. The reason for doing this is to add many parts of a scene information together in a more meaningful way.

(a) An example for a normal image in the Visual Genome dataset



(b) An example of the scene graph for the same image

Figure 3.3: These images are from the Visual Genome dataset to aid in better understanding of scene graphs [7].

3.2 Specification

The system setup will initially have the user exposed to a collection of images where they have to give their preference (what they would like to keep and where they would like to change) to guide the system on the next set of images generated. This process is repeated using a Machine Learning algorithm and graph kernel to rate and generate new graphs respectively. The initial set of images will come from the Visual Genome dataset. This setup is a higher level of interactive image generation where we will use graphs. The graphs will have nodes and edges. A Machine Learning algorithm that can learn from graphs to respond with a LIKE or DISLIKE (Check that the user likes the graph/image). Rendition of this graph is generated as an image after the final scene graph fulfils the user's preferences and it logically makes sense. The following will be phrases in the final system (aided with the Figure 3.4):

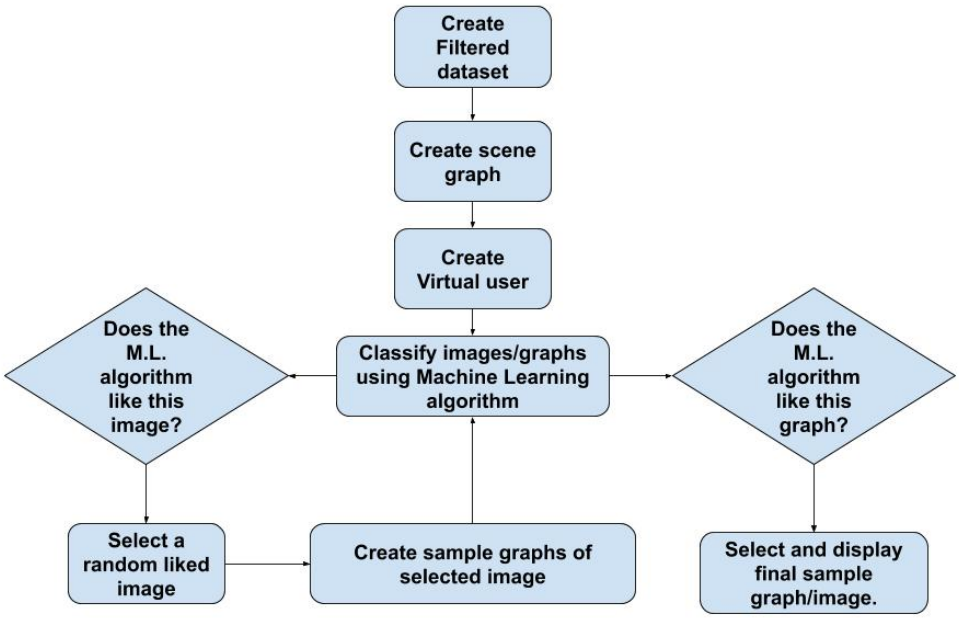


Figure 3.4: A flow chart that describes the flow of the system, where M.L is machine Learning.

1. Learn how to generate a scene graph.
2. Learn what the good and bad graphs look like i.e. what graph makes logical sense (as a region description).
3. Use graph Kernel or graph neural networks to train graphs to say yes or no to answer the question: "Will the user like this graph?".
4. Machine Learning algorithm to score the graph produced after changing a node (with different possibilities) in it. Repeat step 2 to 4.
5. Materialise the final graph and make an image from it. However this has been changed to focus more on the final sample graph generated.

3.2.1 Functional Requirements

Graph Generation - This part of the project has two separate stages; one stage involves the graph already available but changes need to be made in accordance to the user's preference (this context will be expressed in the scene graph). The second stage involves an image where a graph is extracted from. This requirement aims to make a graph that is used in the later phases of the system.

Check Graph Generated - This requirement aims to ensure that the graph generated makes logical sense (The region description makes sense). There should be a set of region graphs produced as a result.

Create Good Image Graph - To make a good graph I will have to check that the nodes (objects, and attributes) in the graph are those that the user said they like. There should be one scene graph created as a result.

Rate Goodness of Graph Created - There should be a machine learning algorithm that can score the scene graph produced in correspondence to the user's preference. The graphs with a higher score are kept. The cycle is repeated from the graph generation requirement (i.e a new graph is created from the already existing one by changing nodes to better suit the user's preference) to rating the goodness of the graph again iteratively.

4 Research Process

In this section we will discuss the process of using Visual Genome dataset, use of python graph library Networkx and Machine Learning algorithm (Random forest classifier) for selecting and creating sample graphs needed to create the final image.

4.1 Milestones

The research process milestones are; produce a filtered Visual Genome set, find an effectively ways of turning all the Visual Genome scene graphs into connected Networkx graphs with all the data in (with the aid of a json file), vectorize them with Eden, have got a basic virtual user to train the classifiers on the data, make sample graphs base on the virtual user choices, display possible good final images.

4.1.1 Create Filtered Image Dataset

As mentioned in the specification, we will use the Visual Genome dataset which contains roughly 108,077 images. An issue arises when searching through this volume of data for objects that the user wants every time the system is run - it takes hours.The total number of objects in the dataset is 3,909,697,so, on average the number of objects in one of the images in the Visual Genome dataset is 36. Therefore on average you have to search through 36 objects to find out if the image has what the user want which is

time consuming.

Algorithm 1: Get selected Images

```
Input: Preference List
Output: List of Image Ids that have those preferences
Select all image ids in the Visual Genome dataset
Select image data for all image ids
Initialise list of interested id
Select a list of image ids
for id in image ids do
    Select image data of id
    Select scene graph of id
    for object in scene graph objects do
        for label in object name do
            if label is in preference list then
                Add the id to list of interested ids;
            end
        end
    end
    if list of interested ids not empty then
        Write the ids into a text file
    end
    else
        Let the user know there is no image that contains preference
    end
end
```

To solve this we used built-in functions from Visual Genome to get the image id (unique number assigned to each image for future reference). To make this possible we search through each image and each object present in each image. The image that contain objects that the user wants are copied into another folder for use in the later stages of this project. The reason for this is to reduce the time used for searching the entire Visual Genome dataset when enquiries are made in the graph creation, Machine Learning classification, and graph sampling stages. There is more elaboration in section 5.1.

4.1.2 Graph Creation and Visualization

We use the filtered JSON data that represents images which contains image information such as the predicate (relationship between two objects), object names and id and subject id (the id of the second objects int the relationship) to make another JSON file that represents a scene graph which has more information describing the objects, and their new relationships with other objects. These relationships are defined in the graph function. This function uses the Networkx python library to represent objects as vertices and relationships between vertices as edges as seen in Figure 3.3 b. This can be called a graph of network. Example of relationships and objects:

Relationship list $\longrightarrow$ [16099: chair at desk, 16100: computer ON desk, 16101: box under desk, 16102: building seen from window, 16103: mug ON desk, 16104: box under desk, 16105: cpu below desk, 16106: mouse ON desk, 16107: bowl ON table, 16108: pot has flowers]

Objects list $\longrightarrow$ [it, building, window, table, pot, flowers, clothing, legs, cabinet, scene, chair, computer, box, sun, mug, wall, mouse, bowl, plant, plant, mouse pad, cup, desk, monitor, floor, mouse, plate]

The outcome of algorithm 1 is to produce a directed graph (a graph consisting of a set of vertices connected by edges in which the edges have their corresponding path) where all the vertices are connected (all each vertex is connected to at least one other vertex). To make this possible, all vertex are connected initially and only edges that exist already and edges that have a smaller weight are left after solving the minimum spanning tree (Kruskal's algorithm) with a time complexity of $O(E \log V)$, where E is an edge and V is a vertex [9]. The graphs created here are extremely important to the rest of this research project. It will be used on a machine leaning algorithm, it will be sampled and used to manipulate (with the help of networkx built-in functions and classes) and to create the desired image. In order to work with the graph we have to see it using the visualising function. To visualise the scene network I used

the networkx draw function as well as the plot function to display both the image and its scene graphs separately.

Algorithm 2: Graph Generation

```
Input: Scene graph data
Output: A connected networkx Digraph
Initialise width and height of image
Initialise scene network type and metadata
for objects in scene graph relationships do
    Find distance between objects
    Add the distance as weight to the scene network
end
for objects in scene graph do
    Add scaled object attribute to scene network
    Add object region box to scene network
end
Solve minimum spanning tree
```

4.1.3 Virtual User

Create virtual users which include more specific scene graph relationship for rating the goodness of the graph. In order to search for a more specific image a virtual user will look at the scene graphs and rate them, like a person in real life would. It is just a function that effectively takes a network and gives it a rating, we have used a rating of 0 or 1 to express the image overall fitness. This is necessary for labelling images according to the user's preference; it acts as the label required for any machine learning algorithms to better classify images. See algorithm 3.

Algorithm 3: Virtual User

```
Input: Scene graph
Output: Rating of 1 or 0
Select vertices
Select edges
Select list of wanted object ans subject ids
for object id in wanted object ids do
    for subject id in wanted subject ids do
        if edge exist between subject and object then
            Ouput 1
        end
        else
            Output 0
        end
    end
end
```

4.1.4 EDeN and Machine Learning Algorithm

EDeN is a decomposite kernel centred on the Pairwise Distance Kernel (NSPDK) neighbourhood subgraph, which induces a comprehensive explanation of a graph element. This in effect makes for robust machine learning algorithms (e.g. quick stochastic gradient descent). This enables the efficiency of supervised and unsupervised learning activities[10]. The most important function of EDeN for the purpose of machine learning is vectorized. Its job is to take a graph and return a compressed sparse matrix of the graph and its features. This allows for classification clustering as well as further machine learning techniques. The theory behind EDeN (created by Dr Fabrizio Costa) is explained in "Learning an efficient constructive sampler for graphs" research paper. The Machine Learning algorithm used is Adaboost which is fully described in section 5.2.2. The reasons for this decision can be seen in sections 5 and 7.

4.1.5 Sample Graphs

The list of graphs that satisfy the virtual user preference is returned to the sample graph function. This function then takes a randomly selected graph from that list of graphs and makes samples of it. It

imitates the action of a user removing elements from the image based on their preference.

Algorithm 4: Sample Graph Generation

```
Input: A randomly selected scene graph
Output: A list of sample graphs
Initialise sample list
Select leaves in the graph
for leaf in leaves do
 | Create sample ⟶ Copy original input graph
end
for vertex in sample vertices do
 | Find shortest path from vertex to leaf
 | Remove the path from the sample graph
 | Append new sample into sample list
end
```

In order to make a sample we iterate over every vertex in the selected graph and remove all vertices in its path to the nearest leaf vertex. This means the sample will have random vertex missing at random is instances covering all possible combination added onto a list. This list is later fed to the Adaboost Machine Learning algorithm to classify which sample are good according to the user's preference again. At the end of it a list of good samples are returned back.

4.1.6 Final Graph and Image

Select a random correct sample graph and show the final image using image inpainting. We used Opencv's inpainting algorithm called INPAINT_TELEA. This algorithm is inspired by the paper "An Image Inpainting Technique Based on the Fast Marching Method" written by Alexandru Telea in 2004 [11]. This is focused on the accelerated marketing strategy. Take a region to be inpainted in the picture. The algorithm begins from this region's borders and reaches the area within its region, which first fills all areas inside the region. A small population is expected to be painted around the pixel of the boundary box. This pixel is substituted by a weighted normalised average of all established neighbourhood pixels. Weight selection is a significant question. The pixels close the centre, next to the usual border and those on the boundary contours are granted greater weighting. Using the Fast Marching Method to switch to the closer pixel after a pixel has been added [11].

To make inpainting possible we need to construct a mask that is the same size as the input image, where the region to be inpainted are non-zero pixels. Therefore, we decided to make the boundary box of removed objects i.e removed vertices as the mask. After the process of inpainting we see a new image with the removed objects missing from the original image.

4.2 Critical Decisions

The first critical decision we made on this project will be the choice of using Python programming language for developing the system. Python is a fairly used and supported programming language with multiple libraries that enable graph representation to be possible. The Library used in this project is called Networkx. It is used to create, evaluate and represent networks (graphs). It offers classes covering many forms of network and application of several other network science algorithms. NetworkX provides multiple functionalities and is fairly simple to use and is suitable for studying network technology , testing on small to medium sized networks and it is well documented [12].

The second critical decision we made was to download and use the local data. Visual Genome offers users the choice to use the API functions to access the data directly from their server. Initially we used this option but over the course of using it we noticed that it took a long time to retrieve the data which would overall increase the time of computation for the project. In order to rectify this, we decided to switch to downloading the local data and using it for the project. It is a lot faster while processing. This is also what influenced our decision to filter the Visual Genome dataset to make our own.

The third critical decision was to focus on selected graph (in this project is it selected at random) from the final filtered graphs to keep and remove from that graph instead of replacing objects in the graph. This make the production of a new image less complicated which was good given the limited time we had towards the end of the project life cycle due to Covid-19 and group project deadline.

The fourth critical decision made was to focus on the final sample graph as the system output rather than the image after applying inpainting. This is because some inpainting result remove the desired elements from the image whilst the graph shows that it is still present in the image. This is because the boundary boxes overlap as a result paints over other boundary boxes that belong to an other element in the image.

5 Testing/Experimentation

To decide how the algorithm performs there is the need to accurately test the system components and compare the results to other algorithms. This section describes the data used to test the system and how the experimental procedure was conducted. The evaluation section of this document has a large part of the experimental design describing the various numerical measures that were used to evaluate the performance of a machine learning algorithm.

5.1 Filtered Visual Genome dataset

Two issues were considered to determine what data set to be used for this project:

- How many other algorithms will the results be matched against?
- How long does this dataset take for the algorithm to train?

It was found that the use of the Visual Genome dataset to train a machine learning algorithm is possible in a published paper about Image Generation from Scene Graphs [5]. However the size of the dataset makes it less manageable by our filtering and machine learning algorithms.

Filtered dataset - a subset of the Visual Genome data set that fulfils the user's main preference because it is not a big dataset, and it only contains data of interest for the user which make it the natural choice. This data set as the filtered data set. see appendix A.

The filtered dataset is any image in the Visual dataset that contains element that correlates with the user's wishes. These elements are equivalent to objects that make up an image. Without the filtered Visual Genome data set or a similar collection, individual researchers developing an interactive image generator would have to use their own data sets, with different segmentation and object detection techniques and documents for image representation (i.e. json files). This would make it very difficult to compare results and hence make it very difficult to make significant contribution in the field of Interactive image generation. Therefore the use of filtered Visual Genome collections allows for proper assessment on algorithm performance in comparison to others.

5.2 Machine Learning Algorithms

We carried out three machine learning related tests in this research project:

- Find the best suited Machine Learning algorithm of the filtered dataset with the aid of the virtual user and EDeN to get the dataset labels and to create an explicit visual structure representation for graphs respectively.
- Measure the classification performance for a virtual user using the top three Machine learning algorithms chosen to find the best one (there was not much difference).
- Test how accurate each virtual user's preference is and how good the sample graphs produced are.

5.2.1 Machine Learning Classifier Comparison

In this section we will investigate how ensemble of trees as well as other classification algorithm perform on the classification task of "filtering images according to user preference" and the importance of the parameters. This will be done in the following stages:

1. What the user preference is for the experiment.
2. Comparing performance with major classification algorithm based on the user preference and select the top three.
3. Choose the best Machine Learning classification algorithm from the top three via further testing.

We train and test the datasets with 17 different classification models and get performance results for each model. These results are based on the user preference "a woman and an elephant". This means the goal of these 17 classification models is to get the images that suit this preference. For each classification model, the recommended parameter values are used for optimal classification.

5.2.2 Testing Top Three Algorithms

We have selected the three ensemble methods of classification following the previous test. The purpose of ensemble methods is to merge predictions made by a few base estimators (An algorithm for choosing the "true," or most likely more reliable, real-world data model [13].) with a particular learning algorithm to boost generalisation / robustness with a single estimator. We used the two types of ensemble methods:

- Averaging methods - Several estimators are designed independently and their predictions are averaged. In general, since the variance of the cumulative estimator is minimized, it is typically higher than just a single base estimator. Examples: Random Forest, Decision Tree.
- Boosting methods: Direct estimators are sequentially designed and the cumulative estimator bias is aimed to reduce. The goal is to put together many weak models for a powerful ensemble. Example: Adaboost.

Decision Tree - In this part we train a decision tree on the filtered data. we will use the implementation of the classification tree in sklearn, which allows us to play with the parameters (read the documentation in the test jupyter notebook). Especially on the max_depth, min_samples_split and min_samples_leaf parameters. In this test we (See section 7.1.2 .) :

- Draw a graph of the mean of the classification accuracy depending on tree depth - use train/test split to get these estimates.
- Calculate the confusion matrix for the tree with the best performance.

Random Forests - In this part, we use Random Forest,while having a sklearn trees as weak learners. We will use the function provided by sklearn: sklearn.ensemble.RandomForestClassifier. Once we have this, we can provide a graph of the accuracy curves for different tree depths and forest size (See Figure 7.1.2):

- The first graph has the forest size as the x-axis, classification error as the y-axis.
- The second graph has tree depth (eg, for 1,2,5 and 10) as the x-axis, classification error as the y-axis.
- As before, the confusion matrix for the forest with best performance is provided.

Boosting - In this last section we use AdaBooost via sklearn tree stumps (trees with a single non-leaf node) as weak learners. We used the function provided by sklearn: sklearn.ensemble. AdaBoostClassifier which supports multi-class problems. Once done, we plot a similar curve like before, with boosting rounds as x-axis, classification error as y-axis and one curve for tree depth of 1,2,5 and 10. See section 7.1.2.

5.2.3 Testing Machine Learning Algorithm Quality

Quality assessment is a key role in machine learning, and a AUC-ROC curve is dependable when grappling with a classification problem. Where the output of the multiple classification problem needs to be tested or visualised, we use the curve AUC (Area Under The Curve) ROC (Receiver Operating Features). This is one of the most significant appraisal metrics to test the efficiency of any classification pattern. This is also referred to as AUROC (Area Under the Receiver Operating Characteristics) [14]. We will be testing the classifier accuracy on each virtual user preference using AUC-ROC and Confusion metrics (See section 7.1.3.)

5.3 Experiment Design

A significant aspect of every Machine Learning project is the evaluation of the machine learning algorithm. When calculated by a metric for accuracy score, the models should give satisfactory results, but could offer bad results when assessed by other metrics such as the Confusion Matrix or some other metrics. Sometimes, we use accuracy of classification to calculate the efficiency of the model, but it is not sufficient to actually test our model. We will cover various kinds of assessment methods used in this project including:

- Classification Accuracy
- Confusion Matrix
- Area under Curve
- F1 Score

5.3.1 Classification Accuracy

This is the proportion of the correct number of estimates to the overall number of input samples. It only functions effectively if every class has the same sample size. Consider for example that our training sets contain 91% Class 1 and 9% Class 2 samples. Our model can then easily obtain 91% accuracy rate by merely anticipating each class 1 training sample. If a test collection of 60% Class 1 and 40% Class 2 samples is checked for the same experiment, then the accuracy of the test will drop to 60%. Classification accuracy is good, but it offers a false impression that high accuracy is achieved. It calculated by this formula [15] :

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

5.3.2 Confusion Matrix

Confusion matrix is an output matrix that defines the entire model efficiency. For example, if there is a sample of two classes like and dislike or 1 and 0; there is a classifier for a certain input sample which predicts a class. These are the following results when testing the model on 100 samples.

n = 100	Predicted: Like	Predicted: Dislike
Actual: Like	65	5
Actual: Dislike	20	10

There are 4 important terms:

- True Positives : Cases where we predicted Like and the output was also Like.
- True Negatives : Cases where we predicted Dislike and the output was Dislike.
- False Positives : Cases where we predicted Like and the output was Dislike.
- False Negatives : Cases where we predicted Dislike and the output was Like.

To measure the accuracy of the matrix, take the diagonal mean of the values [15]:

$$Accuracy = \frac{TruePositive\ +\ TrueNegative}{Total\ number\ of\ samples}$$

$$Accuracy = \frac{65\ +\ 10}{100} \approx 0.75$$

5.3.3 Area under Curve

Area Under Curve (AUC) is among the most commonly employed assessment metrics. It is used for the topic of binary classification. AUC is proportional to the probability that a randomly selected positive example will be ranked higher than the randomly selected negative one. There are two fundamental terms to understand before describing AUC:

1. True Positive Rate (Sensitivity) - True Positive Rate refers in relation to all positive data points to the proportion of positive data points correctly regarded as positive. It is calculated by:

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

   Where TP is True Positive and FN is False Negative [15].

2. False Positive Rate - For all negative data points, the False Positive rate refers to the ratio of negative data points that are erroneously considered as positive. This is calculated by:

   $$FPR = \frac{FP}{TN\ +\ FP}$$

   Where TN is True Negative and FP is False Positive [15].

AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points ranging from 0 to 1. The bigger the value, the better the accuracy of the model.

5.3.4 F1 Score

F1 Score is used to measure a test's accuracy. To achieve this F1 Score tries to find the balance between precision (how many instances it accurately categorises) and recall (A fair amount of cases are not missing). The following formulas shows how to calculate precision and recall:

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

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

Where TP is True Positive, FP is False Positive and FN is False Negative [15].

Therefore the F1 Score is calculated by [15]:

$$F1 = 2 * \frac{1}{\frac{1}{precision} \ + \ \frac{1}{recall}}$$

6 Product and fitness for purpose

The results clearly show that we have successfully used machine learning algorithm to develop the interactive image generation. The quality of the final image is not the best so better inpainting techniques would be required to offer a competitive solution to the problem of interactive image generation. However, we must give credit to Dennis Harrop, who was an inspiration for the creation of a fully connected scene graph and its visualisation that plays a significant part of this projects' success.
Here we identify the initial requirements and comment on whether they have been achieved or not, and justify departing from these requirements. We mention that this is very challenging to achieve because it is a research project and so many of the specifications are very vague to enable creativity in the development.

- Develop segmentation techniques to automatically identify relevant elements in images - This was achieved (see section 4)
- Convert images to networks of connected elements - This was achieved (see section 4)
- Employ graph grammars - This was achieved (see section 4)
- Employ graph kernel machine learning techniques - This has been the main focus of this report to induce the generative rules and inform an optimization algorithm on which actions to perform to generate a novel image. This has been somewhat achieved (see section 4) However perhaps some care should be taken in our definition of novel. We note that perhaps we have departed slightly from the original requirements in that we have focused more on removing selected object from the image than adding a new objects or replacing objects in an image.

The purpose of this project was to generate a novel combination of objects and relationship to make a novel image. This can be seen as removing elements of the image to make it different from the original hence making it new or replacing elements in the image to make it new. Those are two main ways of making a novel image. However this project is focused on keeping already existing elements of the image, and removing other elements seen as unimportant to the user. This does not mean that all the elements not specified by the user were removed. What the system returned was a collection of reasonable images (images that had what the user specified and some other elements). The collection of images include those unwanted elements removed respectively at random. For example, a user wants an image with a woman and an elephant but there are trees, a man and other element of the image available. The system will return one without the man but with the trees and other element including the woman and the elephant, another image will be returned without the man but with the tree and other element. This produces images with different element combinations and the user theoretically decides the image they want from that collection (since there is no user the chosen image it selected at random).

Consider the research problem: **Can graph theory and machine learning generate novel images?** We believe that we have answered this question as we have found that a good way of representing images into scene graphs using Visual Genome image representation (object and relationship analysis) and Networkx (for graph theory). We have used EDeN to make sparse matrix data for Machine Learning algorithm classification (for classifying images into clusters i.e. filtering images) and then representing the

final result into a novel image using Computer Vision technique (Inpainting). This does not necessarily mean that there are no other means of developing an interactive image generation approach. This is an area for future research.

7 Results and Evaluation

This section shows the result of our test and describes the different numerical measures that we use to assess the performance of a machine learning algorithm.

7.1 Results

7.1.1 Machine Learning Classifier Comparison

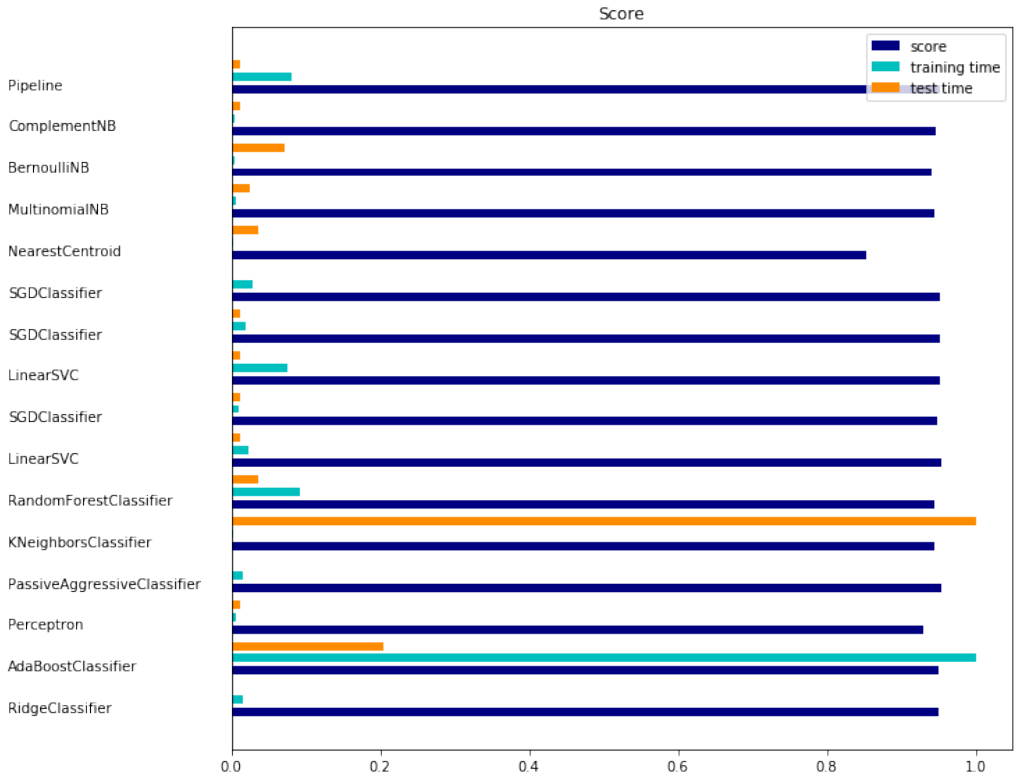
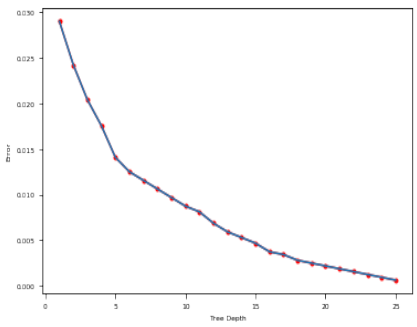
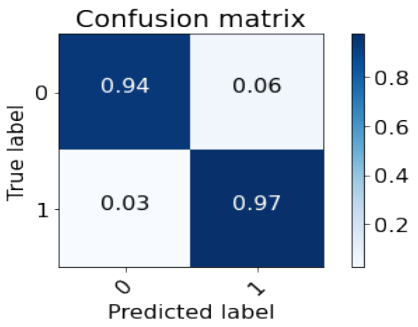


Figure 7.1: These images are from the Machine Learning Classifier Comparison test.

7.1.2 Testing Top Three Algorithms

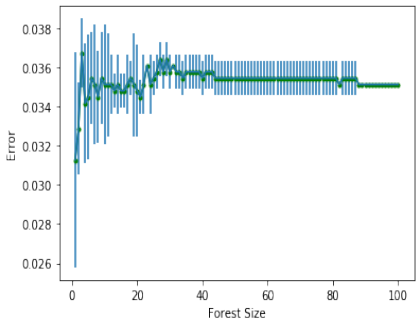


(a) A graph for the mean of the classification accuracy depending on tree depth

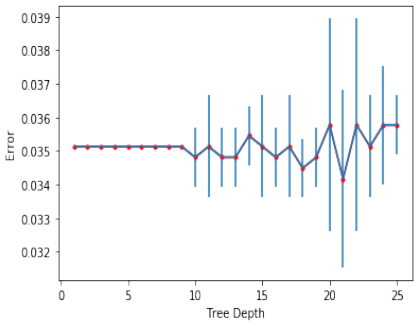


(b) Calculation of confusion matrix for the tree with the best performance.

Figure 7.2: These images are the result of Decision Tree testing.

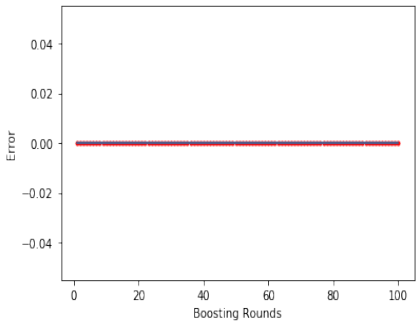


(a) A graph of forest size as the x-axis and classification error as the y-axis.

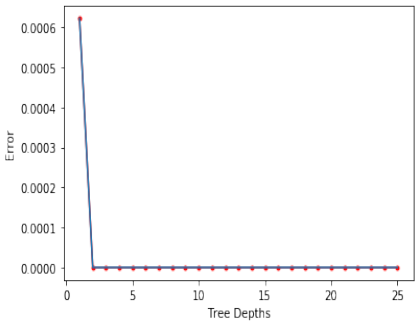


(b) A graph of tree depth (eg, for 1 to 25) as the x-axis and classification error as the y-axis.

Figure 7.3: These images are the result of Random Forests testing.



(a) Plot a similar curve like before, with boosting rounds as x-axis, classification error as y-axis and one curve for tree depth of 1,2,5 and 10.



(b) Calculation of confusion matrix for the tree with the best performance.

Figure 7.4: Results of Adaboost testing.

Table 7.1: A summary table of the best performance along with the estimated processing time obtained by each algorithm in my experiments.

	Decision Tree	Random Forest	Boosted Trees
Best Perfromance (Lowest Error)	0.000622	0.035776	0.000000
Estimated Processing Time (s)	1.4	2.5	1e+02

7.1.3 Testing Machine Learning Algorithm Quality on virtual users

Virtual User 1 - This user wants an image that has an elephant in it.

The following represent accuracy scores :
Recall Baseline: 1.0 Test: 1.0 Train: 1.0
Precision Baseline: 1.0 Test: 1.0 Train: 1.0
Roc Baseline: 0.5 Test: 1.0 Train: 1.0

Table 7.2: Accuracy score: 1.00

	Ground Truth Result					Test Result			
	precision	recall	f1-score	support		precision	recall	f1-score	support
Dislike	1.00	1.00	1.00	4		1.00	1.00	1.00	2
Like	1.00	1.00	1.00	1497		1.00	1.00	1.00	642
accuracy			1.00	1501				1.00	644
macro avg	1.00	1.00	1.00	1501		1.00	1.00	1.00	644
weighted avg	1.00	1.00	1.00	1501		1.00	1.00	1.00	644

Virtual User 2 - This user want an image that has at least two elephants in it.
The following represent accuracy scores :
Recall Baseline: 1.0 Test: 0.84 Train: 0.8
Precision Baseline: 0.55 Test: 0.84 Train: 0.84
Roc Baseline: 0.5 Test: 0.88 Train: 0.88

Table 7.3: Accuracy score: 0.8229813664596274

	Ground Truth Result					Test Result			
	precision	recall	f1-score	support		precision	recall	f1-score	support
Dislike	0.77	0.81	0.79	672		0.80	0.80	0.80	288
Like	0.84	0.80	0.82	829		0.84	0.84	0.84	356
Accuracy			0.81	1501				0.82	644
Macro avg	0.81	0.81	0.81	1501		0.82	0.82	0.82	644
Weighted avg	0.81	0.81	0.81	1501		0.82	0.82	0.82	644

Virtual User 3 - This user wants an image that has a man and an elephant in it.
The following represent accuracy scores :
Recall Baseline: 1.0 Test: 0.81 Train: 0.81
Precision Baseline: 0.18 Test: 0.71 Train: 0.73
Roc Baseline: 0.5 Test: 0.95 Train: 0.95

Table 7.4: Accuracy score: 0.90527950310559

	Ground Truth Result					Test Result			
	precision	recall	f1-score	support		precision	recall	f1-score	support
Dislike	0.96	0.93	0.95	1229		0.96	0.93	0.94	528
Like	0.73	0.81	0.77	272		0.71	0.81	0.76	116
Accuracy			0.91	1501				0.91	644
Macro avg	0.84	0.87	0.86	1501		0.83	0.87	0.85	644
Weighted avg	0.92	0.91	0.91	1501		0.91	0.91	0.91	644

Virtual User 4 - This user wants an image that has an elephant on a road.
The following represent accuracy scores :
Recall Baseline: 1.0 Test: 0.18 Train: 0.33
Precision Baseline: 0.03 Test: 0.44 Train: 0.81
Roc Baseline: 0.5 Test: 0.98 Train: 0.99

Table 7.5: Accuracy score: 0.9642857142857143

	Ground Truth Result					Test Result			
	precision	recall	f1-score	support		precision	recall	f1-score	support
Dislike	0.98	1.00	0.99	1449		0.97	0.99	0.98	622
Like	0.81	0.33	0.47	52		0.44	0.18	0.26	22
Accuracy			0.97	1501				0.96	644
Macro avg	0.89	0.66	0.73	1501		0.71	0.59	0.62	644
Weighted avg	0.97	0.97	0.97	1501		0.95	0.96	0.96	644

Virtual User 5- Note: This user wants an image that has an elephant and a woman in it.
The following represent accuracy scores :
Recall Baseline: 1.0 Test: 0.69 Train: 0.82
Precision Baseline: 0.07 Test: 0.57 Train: 0.63
Roc Baseline: 0.5 Test: 0.94 Train: 0.98

Table 7.6: Accuracy score: 0.9456521739130435

	Ground Truth Result					Test Result			
	precision	recall	f1-score	support		precision	recall	f1-score	support
Dislike	0.99	0.97	0.98	1403		0.98	0.96	0.97	602
Like	0.63	0.82	0.71	98		0.57	0.69	0.62	42
Accuracy			0.96	1501				0.95	644
Macro avg	0.81	0.89	0.85	1501		0.77	0.83	0.80	644
Weighted avg	0.96	0.96	0.96	1501		0.95	0.95	0.95	644

The algorithm performs particularly well with resulting accuracy scores all in 0.9 and 0.8. This is enough proof, it shows that the classifier works well with the user's preference. This indicates that the images produced by this model are very likely to meet the user's liking. See Appendix B for the confusion matrix result and the AUC-ROC graph.

7.1.4 Final system output example

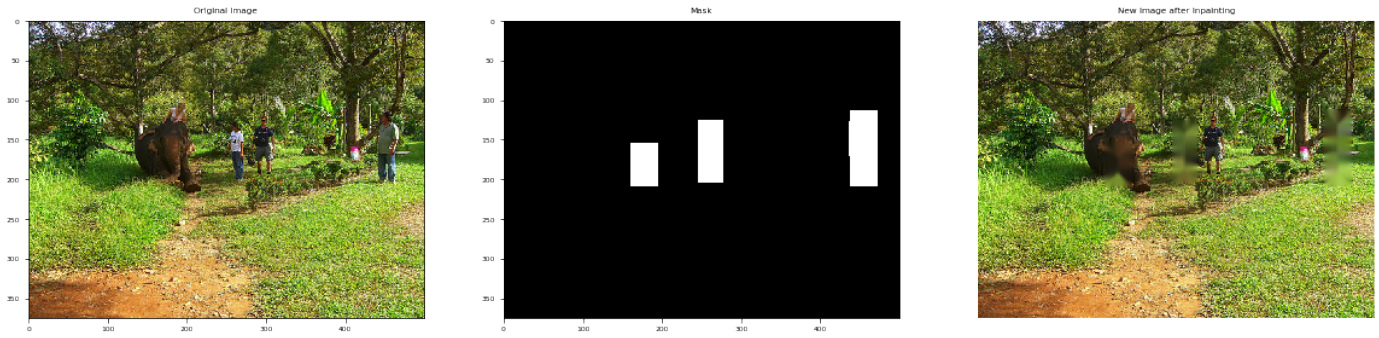
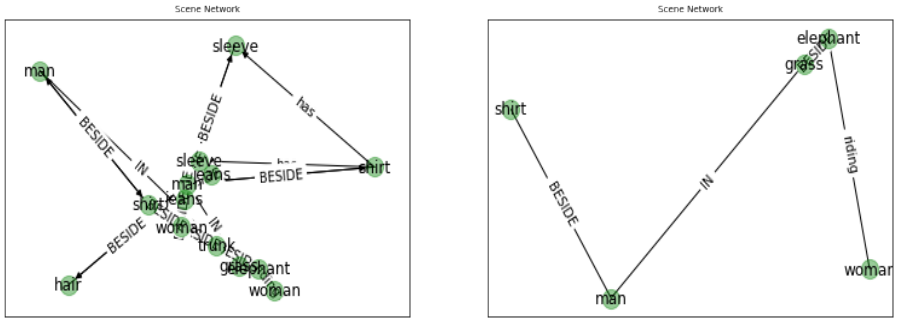


Figure 7.5: The image on the left is the original image, the one in the middle is the mask for the inpainting and the one on the right is the final image



(a) A graph of the mean of the classification accuracy depending on tree depth

(b) Calculate the confusion matrix for the tree with the best performance.

Figure 7.6: These images are result of original graph and sample graph.

The final output is a collection of the sample graphs, the original image, mask and new image (after inpainting). You will notice in some of the image in appendix do not have the desired elements in them because other bounding box of the removed element/objects overlaps the elephant and woman elements, so in the sample graph of those images the woman and elephant are present blurred on the image its self is absent. In addition the inpainting does not always produce the result we expected as explained in section 7.1.4.

7.2 Evaluation

7.2.1 Machine Learning Classifier Comparison

As seen in Figure B.10, most classifiers have at least around 0.85 accuracy scores. This is a good score for a classification task. According to [16], most professionals have an understanding that high precision scores (from 0.75 to 0.90) are good and over 0.90 are excellent. The precision of the grade 0.90 or even of the score 0.99 may be insignificant in the case of an imbalanced classification problems. The result are valid as there are no signs of class distribution imbalance (both classes of like and dislike are roughly equal).

7.2.2 Testing Top Three Algorithms

The experiment had a range of 1 to 25 tree depths, because after 10 12 tree depth the graph stayed linear (It doesn't change values) even up to 100 tree depth. To ensure the performance comparison was fair I made the overall tree depth the same for all the respective methods. According to the table of processing time and best performance (which was measured by the lowest error generated from the set tree depth range), Boosted trees method had the best performance and it took the lest estimated processing time. The Decision Tree has the worst performance compared to the other two but took the least estimated processing time. see Table 7.1.

7.2.3 Testing Machine Learning Algorithm Quality on virtual users

This test is done on a filtered elephant dataset (i.e all the images in this dataset had an elephant in it). The accuracy are quite impressive however the class distribution on virtual user 1,2,4 and 5 were disproportional in favour of the "dislike" class as seen in the tables (under the support column). The only case of equally distributed classes is in the virtual user 5 (see table 7.4) and the accuracy score is 0.82 which is good.

7.3 Critical Analysis

In this section, we continue our critical assessment of the final interactive image generation system, as well as analyse the project process and overall approach.

7.3.1 Limitations

Over the research and testing process we noticed that the size of the Visual Genome dataset was considerably large for the computer and for Google Colab (data analysis tool) to process, as it required a lot of

memory, causing slow processing and sometimes raises memory error messages. The two main segments of this research project affected by this were the creation of the filtered dataset and retrieving sparse matrix from the compressed sparse row matrix of vectorized scene graphs. As expressly stated in section 4.1.1 the Visual Genome dataset contains a large amount of data; when filtered through requires a lot of time.

Additionally, after graphs have been made from the filtered dataset a Sparse matrix is required- Matrices with mainly zero values are considered sparse, while matrices with non-zero values are considered dense.

Vast sparse matrices are popular in general and particularly in applied machine learning, such as data encoding that map clusters to counts, and also in entire machine learning sub-fields such as natural language processing [17]. It is computationally expensive to represent and work with sparse matrices. Problem that arose while dealing with spare matrices was in regards to space and time complexity [17]. Many large matrices need a large amount of memory, and those really large matrices used in this projects are sparse. This is where the compressed spares row matrix comes in play. EDeN (mentioned in section 4.1.4) exports a vectorized function that converts a list of scene graphs input to a data matrix output. The output format is a scipy Compressed Sparse Row matrix of dimension (2145, 65537) which implies that there are 2145 scene graph Instances and 65537 Features overall with an average of 30 features per instance. This martix is too large for any machine to display (see appendix C for more information). This becomes a problem when we tried to carry out k-fold cross validation.

Cross-validation is a remodelling technique used to validate a machine learning algorithm on a small data sample. This has a parameter named k which corresponds to the number of categories to be divided into for a specified data sample. The technique is sometimes referred to as cross-validation k-fold. By choosing a particular value for k, it may be included in the model comparison instead of k, with k=5 being a 5-fold cross-validation. In practical machine learning, cross-validation is mainly used to approximate a machine learning algorithm's capacity to learn unknown results[18]. In other words, to use a small segment to approximate how the algorithm is likely to behave and then used it to create assumptions regarding data not used during testing. This is a common approach as it is easy to explain and usually results in a less bias or less positive approximation of the ability of the algorithm relative to other approaches, such as an easy split train / test [18].

Another problem is the Visual Genome dataset object and relationship annotation. There are instances in the dataset object annotation where the object is wrong. For example; a scene graph that has a vertex of the letter "a" (this translate to "a" as an object). This can also be seen with the relationship annotation. For example ; edge between two objects namely; "man" and "jean", the relationship is "wearing a vest and". The problem lies within the edge label (relationship) because it has an object "vest" as part of the relationship.

In summary, our Machine Learning algorithm performs extremely well, but with its current settings and with a large dataset reflected in the Sparse matrix dimensions, there is no way of carrying out K-Fold evaluation. However, the classification result comes out fine with a simple train/test split but at a great computational cost and the Visual Genome dataset needs to be revised and improved.

7.3.2 Project Process And Approach

The result obtained in this project were good however there are still further improvements that could be made in some of the approaches.

First of all, the author remembers writing code for performing high level changes to an image (deleting and replacing elements in an image with the help of inpainting) without much focus on graph representation and machine learning algorithms. This was a huge mistake on his part as he realised the time dedicated to create an end product that was not in any way acceptable for the project proposal. This obviously led to panic and the author looking for easy, already available libraries on Github like sg2im to help compensate for the time wasted. However with further enquires from projects owner, supervisor and fellow students he was able to meet the requirement.

Secondly, due to the fact that the author had another project with a different group of people and the rise of Covid-19, he did not organise the code development process and other parts of the research as well as he should have done. However during the end of the project the author made significant progress

on testing and documentation. This involved choosing the most appropriate machine learning algorithm (after testing 17 algorithms) and tidying up loose ends in the code and report.

The part of the process that took a lot of time was processing through the Visual Genome dataset and filtering the wanted images out of it. During the development of this part of the system it took 3 days for this process to be completed and verified.

On the design and development of the system, there have been two major changes to the initial phases of the system:

1. Checking that the new graph make logical sense: we decided to make a sub-tree of the original scene graph containing objects that the user specifies and their relationship with a combination of other objects. With this in mind, those newly sub trees cannot be wrong since they are just sub-trees of the original graph and it is connected (there is no object in the sub-tree that does not share an edge with at least one other node/vertex/object).
2. Materialise the final chosen graph and make a final image as a result: Due to the effects of the Covid-19 and other unforeseen circumstances we were unable to effectively implement this part of the project.

8 Conclusion

To conclude this work, Visual Genome was used and therefore negating the need to develop or test segmentation techniques, filtered the dataset to create a new one that contains only the specified elements the user wants. Also Eden was used to create a Compressed Sparse Row Matrix to enable for machine learning classification. The model created by the machine learning algorithm Adaboost with a decision tree as the weak classifier. It also classifies a group of images pertaining to the users preference. An image is selected at random and samples were created from that image with elements removed allowing for a new combination of elements in the graph. Adaboost is used to rate the goodness of the new sample graph. This in turn returned a list of good graphs (on a structural and logical level) and a graph was chosen at random as the final image.

8.1 Future Work

It would be beneficial if further research on inpainting and making the final image look more natural compared to the result/ outcome of this project is carried out. Also further research on replacing nodes and making new images with elements from other images in new sequence should be encouraged. Another research could be to improve the quality of object cognition and annotation in the visual genome dataset for better quality graph representation and in turn better image description.

Additionally, related to machine learning in structured domains (i.e. domains where data is best encoded as graphs) and in particular in the problem of constructive machine learning or design tasks that a machine could learn is another area of research [1]. These future research ideas would contribute more to the Machine Learning and Computer Vision fields in computer science.

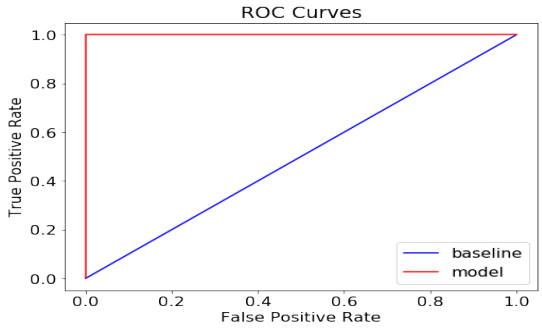
A Filtered Dataset (Elephant)

A.1 Elephant dataset

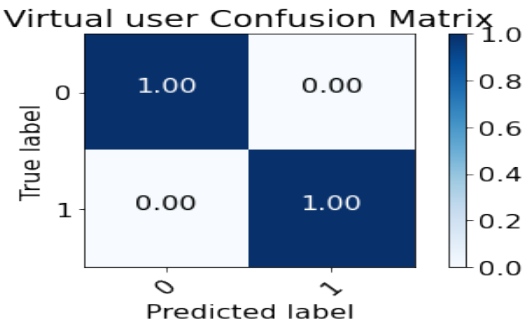
Figure A.1: This show the images in filtered elephant dataset.

B Other Results

B.1 Virtual User 1



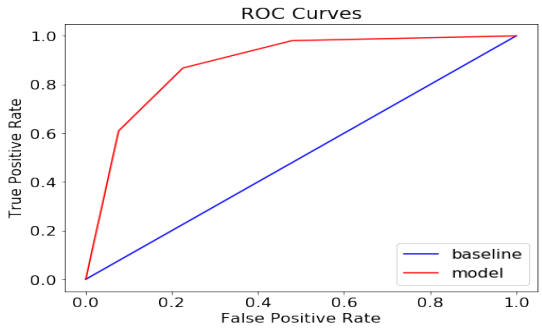
(a) We plot a ROC Curve



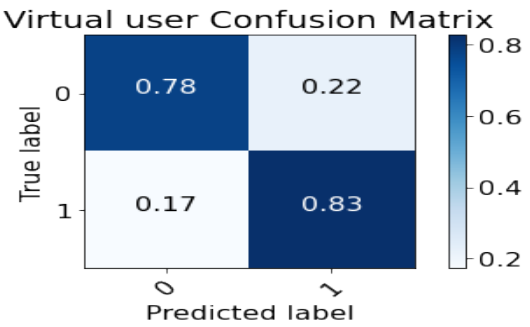
(b) Calculate the normalised confusion matrix for virtual user 1.

Figure B.1: Virtual user 1 - This user want an image that has an elephant in it.

B.2 Virtual User 2



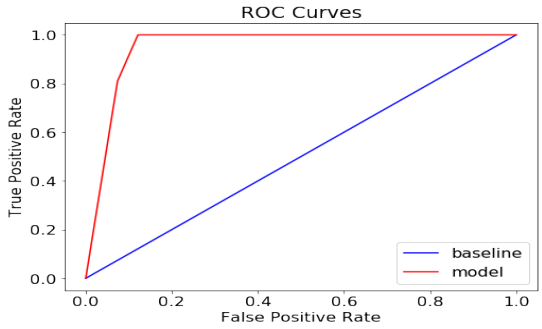
(a) We plot a ROC Curve



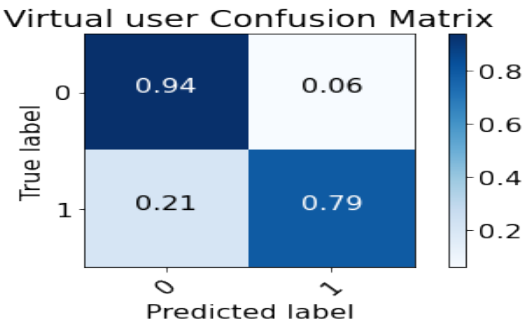
(b) Calculate the normalised confusion matrix for virtual user 2.

Figure B.2: Virtual user 2 - This user want an image that has at least two elephants in it.

B.3 Virtual User 3



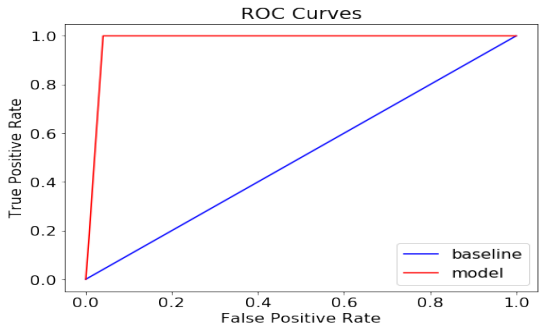
(a) We plot a ROC Curve



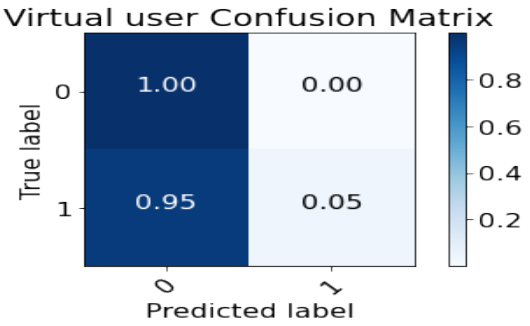
(b) Calculate the normalised confusion matrix for virtual user 3.

Figure B.3: Virtual user 3 - This user want an image that has a man and an elephant in it.

B.4 Virtual User 4



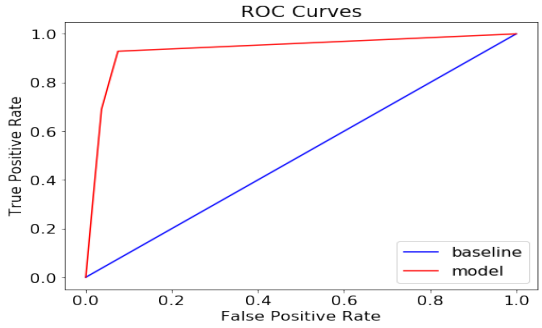
(a) We plot a ROC Curve



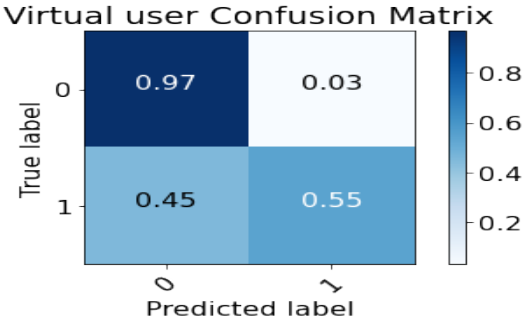
(b) Calculate the normalised confusion matrix for virtual user 4.

Figure B.4: Virtual user 4 - This user want an image that has an elephant on road.

B.5 Virtual User 5



(a) We plot a ROC Curve



(b) Calculate the normalised confusion matrix for virtual user 5.

Figure B.5: Virtual user 5 - This user want an image that has an elephant and a woman in it.

B.6 Final Results

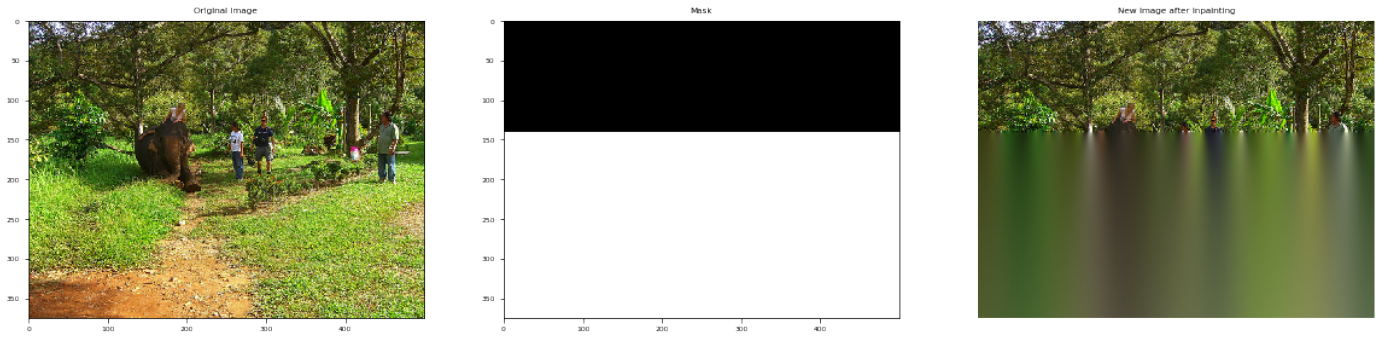
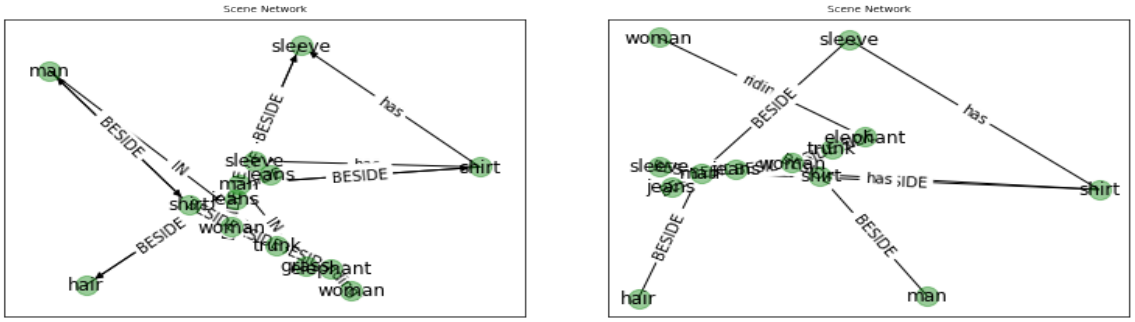


Figure B.6: The image on the left is the original image, the one in the middle is the mask for the inpainting and the one on the right is the final image



(a) Original Scene Graph

(b) Sample Graph

Figure B.7

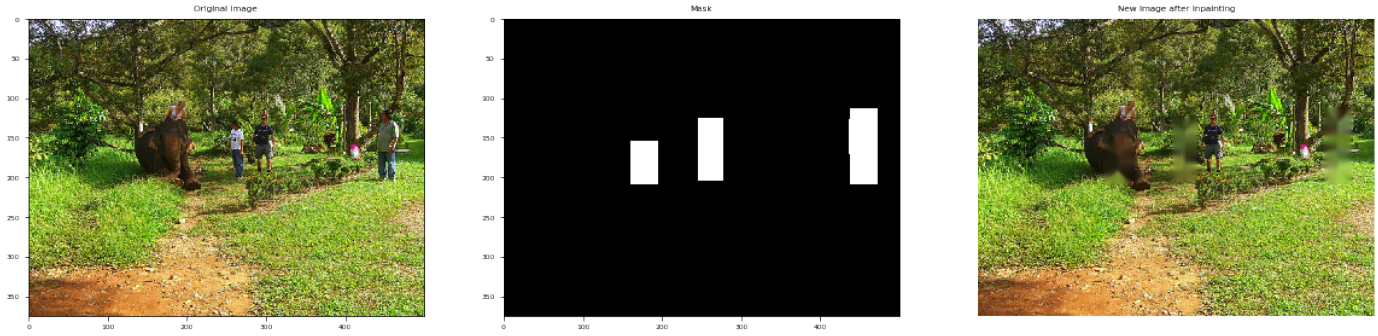
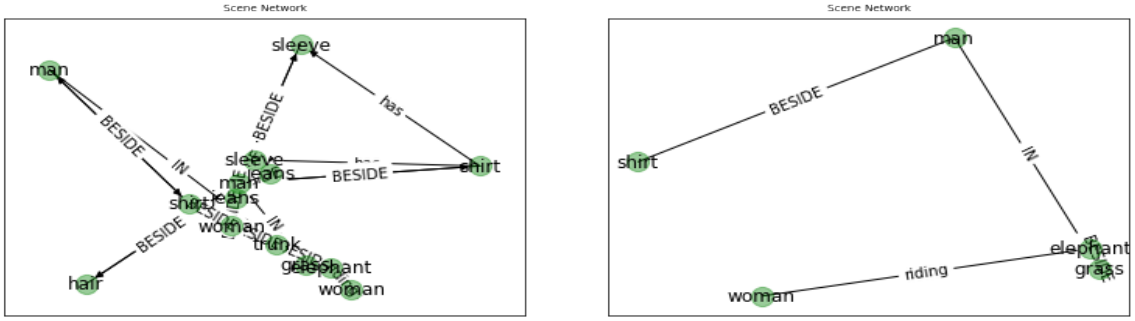


Figure B.8: The image on the left is the original image, the one in the middle is the mask for the inpainting and the one on the right is the final image



(a) Original Scene Graph

(b) Sample Graph

Figure B.9

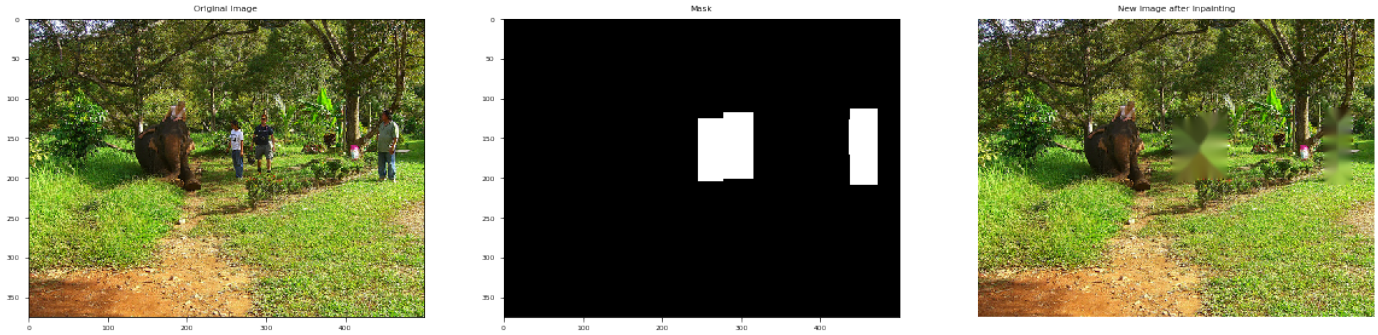
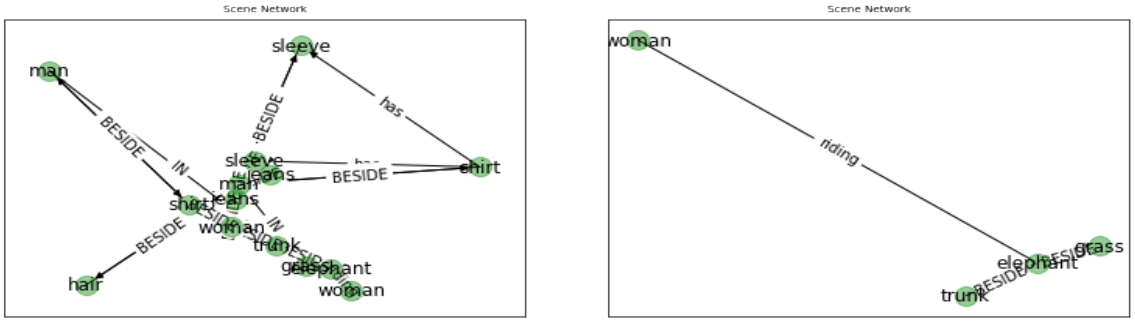


Figure B.10: The image on the left is the original image, the one in the middle is the mask for the inpainting and the one on the right is the final image



(a) Original Scene Graph

(b) Sample Graph

Figure B.11

C Sparse Row Matrix

C.1 Compressed Sparse Row Matrix

(0, 361) 0.03361463227264074
(0, 362) 0.03361463227264074
(0, 1433) 0.01680731613632037
(0, 2028) 0.13445852909056297
(0, 3829) 0.01680731613632037
(0, 5541) 0.050421948408961116
(0, 6042) 0.03361463227264074
(0, 6063) 0.03361463227264074
(0, 6106) 0.03361463227264074
(0, 6288) 0.03361463227264074
(0, 6442) 0.03361463227264074
(0, 6678) 0.08403658068160186
(0, 6746) 0.03361463227264074
(0, 7007) 0.1176512129542426
(0, 7712) 0.01680731613632037
(0, 8480) 0.06722926454528148
(0, 9540) 0.5042194840896111
(0, 10737) 0.03361463227264074
(0, 10882) 0.16807316136320372
(0, 11490) 0.03361463227264074
(0, 11842) 0.10084389681792223
(0, 11922) 0.01680731613632037
(0, 12036) 0.01680731613632037
(0, 12266) 0.03361463227264074
(0, 12637) 0.08403658068160186
: : : : :
(2144, 42889) 0.15018785229652756
(2144, 44130) 0.050062617432175854
(2144, 44712) 0.20025046972870342
(2144, 44738) 0.050062617432175854
(2144, 49811) 0.050062617432175854
(2144, 51178) 0.050062617432175854
(2144, 51514) 0.050062617432175854
(2144, 52085) 0.050062617432175854
(2144, 52245) 0.025031308716087927
(2144, 53363) 0.050062617432175854
(2144, 53403) 0.025031308716087927
(2144, 55297) 0.025031308716087927
(2144, 58071) 0.025031308716087927
(2144, 58730) 0.3003757045930551
(2144, 59549) 0.025031308716087927
(2144, 60108) 0.40050093945740683
(2144, 60602) 0.050062617432175854
(2144, 61410) 0.10012523486435171
(2144, 61666) 0.10012523486435171
(2144, 61921) 0.07509392614826378
(2144, 62018) 0.050062617432175854
(2144, 62896) 0.3003757045930551
(2144, 63682) 0.10012523486435171
(2144, 64069) 0.025031308716087927
(2144, 64228) 0.050062617432175854

C.2 Data

[[0.03361463, 0.03361463, 0.01680732, 0.13445853, 0.01680732, 0.05042195, 0.03361463, 0.03361463, 0.03361463, 0.03361463, 0.03361463, 0.03361463, 0.08403658, 0.03361463, 0.11765121, 0.01680732, 0.06722926, 0.50421948, 0.03361463, 0.16807316, 0.03361463, 0.1008439 , 0.01680732, 0.01680732, 0.03361463, 0.08403658, 0.01680732, 0.01680732, 0.03361463, 0.06722926, 0.03361463, 0.33614632, 0.06722926, 0.23530243, 0.03361463, 0.03361463, 0.1008439 , 0.30253169, 0.05042195, 0.20168779, 0.01680732, 0.03361463, 0.03361463, 0.13445853, 0.06722926, 0.01680732, 0.16807316, 0.01680732, 0.01680732, 0.16807316, 0.16807316, 0.03361463, 0.06722926, 0.1008439 , 0.03361463, 0.03361463, 0.1008439 , 0.05042195, 0.03361463, 0.01680732, 0.06722926, 0.03361463, 0.08403658, 0.01680732, 0.06722926, 0.15126585, 0.01680732, 0.01680732, 0.03361463, 0.03361463, 0.08403658, 0.05042195, 0.25210974, 0.03361463, 0.06722926, 0.06722926, 0.1008439 , 0.16807316, 0.01680732, 0.03361463, 0.03361463, 0.03361463, 0.01680732, 0.01680732, 0.01680732, 0.01680732, 0.01680732]

[0.10468478, 0.10468478, 0.26171196, 0.10468478, 0.41873914, 0.05234239, 0.10468478, 0.10468478, 0.05234239, 0.10468478, 0.10468478, 0.05234239, 0.10468478, 0.10468478, 0.10468478, 0.10468478, 0.10468478, 0.05234239, 0.10468478, 0.31405435, 0.05234239, 0.05234239, 0.05234239, 0.05234239, 0.15702718, 0.52342392, 0.10468478, 0.05234239, 0.10468478, 0.10468478, 0.05234239, 0.10468478, 0.05234239, 0.05234239, 0.05234239, 0.05234239, 0.05234239, 0.10468478, 0.20936957, 0.05234239, 0.05234239, 0.05234239, 0.10468478, 0.10468478, 0.05234239, 0.20936957]

[0.03115885, 0.0623177 , 0.0623177 , 0.03115885, 0.56085926, 0.0623177 , 0.03115885, 0.31158848, 0.0623177 , 0.12463539, 0.0623177 , 0.0623177 , 0.03115885, 0.0623177 , 0.03115885, 0.37390617, 0.03115885, 0.0623177 , 0.03115885, 0.31158848, 0.12463539, 0.03115885, 0.0623177 , 0.03115885, 0.0623177 , 0.0623177 , 0.0623177 , 0.18695309, 0.03115885, 0.0623177 , 0.12463539, 0.0623177 , 0.03115885, 0.03115885, 0.28042963, 0.03115885, 0.0623177 , 0.24927078, 0.0623177 , 0.03115885, 0.15579424, 0.03115885, 0.0623177 , 0.15579424]
...

[0.08899883, 0.08899883, 0.04449942, 0.04449942, 0.08899883, 0.08899883, 0.08899883, 0.17799766, 0.53399299, 0.35599533, 0.08899883, 0.04449942, 0.08899883, 0.08899883, 0.04449942, 0.08899883, 0.04449942, 0.04449942, 0.04449942, 0.04449942, 0.17799766, 0.08899883, 0.04449942, 0.08899883, 0.08899883, 0.35599533, 0.04449942, 0.17799766, 0.08899883, 0.08899883, 0.08899883, 0.04449942, 0.08899883, 0.08899883, 0.04449942, 0.04449942, 0.04449942, 0.04449942, 0.2669965 , 0.13349825, 0.04449942, 0.08899883, 0.08899883, 0.04449942, 0.08899883, 0.04449942, 0.08899883, 0.2669965 , 0.04449942, 0.04449942]

[0.04576288, 0.04576288, 0.09152575, 0.02288144, 0.04576288, 0.13728863, 0.04576288, 0.02288144, 0.68644314, 0.02288144, 0.04576288, 0.13728863, 0.02288144, 0.02288144, 0.02288144, 0.04576288, 0.02288144, 0.11440719, 0.04576288, 0.02288144, 0.04576288, 0.02288144, 0.04576288, 0.16017007, 0.02288144, 0.02288144, 0.04576288, 0.32034013, 0.09152575, 0.02288144, 0.02288144, 0.09152575, 0.06864431, 0.02288144, 0.04576288, 0.34322157, 0.1830515 , 0.09152575, 0.04576288, 0.04576288, 0.04576288, 0.13728863, 0.02288144, 0.04576288, 0.22881438, 0.06864431, 0.04576288, 0.1830515 , 0.04576288, 0.02288144, 0.06864431, 0.04576288]

[0.02503131, 0.05006262, 0.05006262, 0.15018785, 0.10012523, 0.05006262, 0.15018785, 0.02503131, 0.02503131, 0.15018785, 0.05006262, 0.05006262, 0.02503131, 0.10012523, 0.02503131, 0.10012523, 0.10012523, 0.05006262, 0.02503131, 0.02503131, 0.55068879, 0.10012523, 0.02503131, 0.05006262, 0.05006262, 0.05006262, 0.05006262, 0.2753444 , 0.05006262, 0.05006262, 0.15018785, 0.05006262, 0.20025047, 0.05006262, 0.05006262, 0.05006262, 0.05006262, 0.05006262, 0.02503131, 0.05006262, 0.02503131, 0.02503131, 0.02503131, 0.3003757 , 0.02503131, 0.40050094, 0.05006262, 0.10012523, 0.10012523, 0.07509393, 0.05006262, 0.3003757 , 0.10012523, 0.02503131, 0.05006262]]

Bibliography

[1] Dr Fabrizio Costa. Project proposals. http://emps.exeter.ac.uk/computer-science/staff/fc334/project_proposals#fHucti4rHtMAkeIw.99, 2019.

[2] Amy Hodler Mark Needham. How graph algorithms improve machine learning. https://www.oreilly.com/content/how-graph-algorithms-improve-machine-learning/, 2020.

[3] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. https://arxiv.org/abs/1609.03552, 2020.

[4] Gaurav Mittal, Shubham Agrawal, Anuva Agarwal, Sushant Mehta, and Tanya Marwah. Interactive image generation using scene graphs. https://arxiv.org/abs/1905.03743, 2020.

[5] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. https://arxiv.org/abs/1804.01622, 2020.

[6] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[7] Yuke Zhu Ranjay Krishna, Justin Johnson Oliver Groth, Kenji Hata, Joshua Kravitz, Stephanie Chen, Li-Jia Li Yannis Kalantidis, David A. Shamma, and Li Fei-Fei Michael S. Bernstein. Visual genome, connecting language and vision using crowd sourced dense image annotations.

[8] George A. Miller. Wordnet: a lexical database for english. comm, 1995.

[9] Minimum spanning tree tutorials & notes: Algorithms. https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/.

[10] Fabrizio Costa. Eden. http://fabriziocosta.github.io/EDeN/.

[11] Alexander Mordvintsev and K Abid. Image inpainting. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_photo/py_inpainting/py_inpainting.html.

[12] metadataController.pageTitle. https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789955316/1/ch01lvl1sec12/what-is-networkx.

[13] https://deepai.org/machine-learning-glossary-and-terms/estimator, 2020.

[14] Sarang Narkhede. Understanding auc-roc curve. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5, 2020.

[15] Aditya Mishra. Metrics to evaluate your machine learning algorithm. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234, May 2020.

[16] Jason Brownlee. Failure of classification accuracy for imbalanced class distributions. https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/#:~:text=Therefore,mostpractitionersdevelopanintuitionthatlarge,maybetrivialonanimbalancedclassificationproblem., Jan 2020.

[17] Jason Brownlee. A gentle introduction to sparse matrices for machine learning. https://machinelearningmastery.com/sparse-matrices-for-machine-learning/#:~:text=Large%20sparse%20matrices%20are%20common%20in%20general%20and,of%20machine%20learning%20such%20as%20natural%20language%20processing., 2020.

[18] Jason Brownlee. A gentle introduction to k-fold cross-validation. https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=k-Fold%20Cross-Validation.%20Cross-validation%20is%20a%20resampling%20procedure%20used,such%2C%20the%20procedure%20is%20often%20called%20k-fold%20cross-validation., 2020.