A PRELIMINARY PROJECT REPORT ON

DEPICTION OF A SCENARIO FROM ITS TEXT DESCRIPTION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

IN THE PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

CHETAN CHAKU B150058524

RITESH GHORSE B150058538

JUNAID MAGDUM B150058568

UNDER THE GUIDANCE OF

DR. A. M. BAGADE

DEPARTMENT OF INFORMATION TECHNOLOGY

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE .

PUNE - 411043

CERTIFICATE

This is to certify that the preliminary project report entitled

"DEPICTION OF A SCENARIO FROM ITS TEXT DESCRIPTION"

Submitted by

Chetan Chaku B150058524

Ritesh Ghorse B150058538

Junaid Magdum B150058568

is a bonafide work carried out by them under the supervision of Prof. Dr. A. M. Bagade and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University for the award of the Degree of Bachelor of Engineering (Information Technology)

This project report has not been earlier submitted to any other Institute or University for the award of any degree or diploma

Dr. A. M. Bagade	Dr. B. A. Sonkamble
Internal Guide	Head of Department
Department of Information Technology	Department of Information Technology

Dr. P. T. Kulkarni

External Examiner — Principal

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

Date: Pune.

Place:
Date:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards Dr.A.M.Bagade, Department of Information Technology, PICT PUNE, and our instructor for his invaluable support and guidance provided throughout the project. We consider ourselves very fortunate to have worked under his supervision.

We are also grateful to Prof. S.B.Deshmukh and Dr.S.C.Dharmadhikari for their continued guidance and support.

Thanking You,

Chetan Chaku
Ritesh Ghorse
Junaid Magdum

CONTENTS

ABSTRACT

Most of the scenic descriptions around the world today are in a textual format. While text specifications provide a general overview of the scenario, visual representation in the form of high resolution images give us a better understanding. A textual representation is prone to different interpretations by different people, however, a visual image of the scenario is precise in its meaning. The widespread availability of text descriptions must hence, be put to use for generation of high resolution images based on this text narrative. The recent emergence of Generative Adversarial Networks in the field of Computer Vision offers an unique opportunity to apply this concept to develop feature rich images from textual data. Such an application will allow any user to form a complete and clear perspective of a scenario from much readily available textual data.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Our first basic working model is related to the text descriptions of the objects. The architecture stated below has been constructed in such a way as to enable the development of a low resolution image based on the general outline of the object in the description. Finer details in the text will then be used to enhance this low resolution image into a high resolution image which will successfully encompass all the attributes present in the text description. Furthermore, the concept of conditional augmentation will be used to increase the accuracy of our model. Our model is able to generate an image from its text description. Basically, our model a is tool to convert text descriptions into corresponding images. It is achieved by implementing Generative Adversarial Networks which is capable of generating high quality images. The application is presented in the form of a web-application which can be accessed using any kind of device such as laptops, mobile phones, etc.

1.2 MOTIVATION

All different kinds of data description websites use text as their base format. This implies that a large amount of text is available for characterization of objects. While this text can be used to read about the object, a visual representation offers a clearer perspective on the outline of the object. That is, text descriptions are open to many interpretations. It creates ambiguity in understanding for different people. Therefore to interpret any object or situation significantly better, we can generate a high resolution image from text description. The emergence of GANs have made this aim feasible. GANs have the capability to generate the data from scratch with a very high accuracy. This makes GANs the go-to technology for attempting new and interesting machine learning applications. GANs typically is the architecture consisting of neural networks, more precisely, deep convolutional neural networks. Deep convolutional neural networks have shown great results in the field of computer vision tasks specifically. GANs are built on top of this architecture. Recent developments in this technology has shown amazing results in computer vision such as generation of celebrity images. This task involved training of generative adversarial network on the actual celebrity images. But when tested, it is capable to generate the images of a celebrity that does not even exist. Such is the potential of the generative adversarial networks. When applied to different fields such as medical imagery or creation of anime characters, generative adversarial networks can be used to the best of their potential.

CHAPTER 2
LITERATURE REVIEW

2.1 EXISTING METHODOLOGIES

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza et.al suggest a new framework for estimating generative models via an adversarial process. They study the earlier available approaches to generative models and propose the first ever generative adversarial networks that are based on the training of 2 models: G and D where G stands for generator and D for discriminator, both are multilayer perceptrons where G captures data distribution and D estimates whether a sample belongs to the training data rather than G. This competent approach is responsible for the substantially high accuracy obtained for developing generative models and thus, can be used for a variety of new applications.

Han Zhang, Tao Xu, Hongsheng Li et.al highlight the challenges faced by GANs in generating high quality images. To overcome this issue, the authors use two GANs in each stage of a development process to generate high resolution images. A system of advanced multi stage GAN architecture is also proposed for both generative and non-generative tasks, additionally, through the extensive experiments it is demonstrated that the proposed architecture significantly outperforms any other prevalent methods in generating photo-realistic images.

Han Zhang, Tao Xu, Hongsheng Li et.al in this paper, state the importance and need of developing detailed images from sample text descriptions. A detailed practical solution that involves the use of two GANs where, Stage-1 GAN sketches a primitive image and acts as an input to Stage-2 GAN which is responsible for the addition of necessary details has been implemented.

In the paper Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (DCGANs) by Alec Radford, Luke Metz, Soumith Chintala, the authors have proposed a new architecture for implementation of GANs using deep convolution networks. This new architecture mainly composes of convolution layers without max pooling or fully connected layers while at the same time using convolutional stride and transposed convolution for the downsampling and the upsampling of images.

In this paper, the authors i.e Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He, propose an Attentional Generative Adversarial Network (AttnGAN) that allows attention-driven, multi-stage refinement for fine-grained text-to-image generation. With a novel attentional generative network. The AttnGAN can synthesize fine-grained details at different subregions of the image by paying attentions to the relevant words in the natural language description. In addition, a deep attentional multimodal similarity model is proposed to compute a fine-grained image-text matching loss for training the generator.

2.2 PROPOSED METHODOLOGIES

Our first basic working model is related to the text descriptions of the objects. The architecture stated below has been constructed in such a way as to enable the development of a low resolution image based on the general outline of the object in the description. Finer details in the text will then be used to enhance this low resolution image into a high resolution image which will successfully encompass all the attributes present in the text description. Furthermore, the concept of conditional augmentation will be used to increase the accuracy of our model. The concept of conditional augmentation is nothing but the additional processing of the normal text embeddings. This is achieved by sampling the normal text embeddings with Gaussian distribution on two parameters - mean and diagonal covariance as the function of the text embeddings. These embeddings are feed to the architecture in two stages which is described in the architecture section ahead. Further, the two generative adversarial networks are constructed and are connected back to back like a stack architecture. The two networks are required because the accuracy of the images obtained in first stage is hazy and it is not able to detect all the features. Though, both the generative adversarial networks used in Stage-1 and Stage-2 seems similar in terms of architecture, they are trained on different features and the input to each one of them is different. Therefore, the image obtained at the end of Stage-2 is more clear and with comparatively more finer detail. The detailed architecture of the network is discussed in chapters ahead.

CHAPTER 3

REQUIREMENT SPECIFICATION AND ANALYSIS

3.1 PROBLEM DEFINITION

Generation of high resolution images representing a scenario or an object from its text description using Generative Adversarial Networks(GANs) which will help in visualizing different scenarios.

3.2 CONCEPT

The main concept of our project is to create images from its text description. Many times it is observed that people have the description of a particular object or scene in their mind but they cannot convey it to others while conversing. Our model will help to create images from the text description given by the user and generate an image accordingly. This image will be further available to the user to be downloaded from our chatbot interface to be used according to his/her own application. The image generation from its text description is achieved by converting the text description to its word embedding form and then by using Neural Network create and image by the concept of Generative Adversarial Network.

3.3 SCOPE

The model is capable of creating an image from its text description for the classes in MS-COCO dataset. Also, the model is trained sufficiently to detect basic features in the text descriptions such as grass field, beaches and many other background information. The dataset used for our particular application has many categories referenced in a particular image. This has led the model to identify bigger and broader categories better as compared to the finer details. Thus, descriptions that consist of general or comprehensive features such as the sky, beaches, fields, etc. are represented with better accuracy in our application, while the finer details of the object cannot be ascertained by the model due to the complexity of the scene description. This problem can be rectified with further training and a more compatible dataset.

3.4 OBJECTIVES

1. Generate a scenario described by the text description.
2. Help people download the images of objects they describe based on memory.
3. Provide better accuracy than rough sketches drawn from the description.
4. Remove ambiguity from description by converting them to photo realistic images.
5. Generate images for objects or scenarios which might not be generalized (ie) that might have some customizations based on the user's imagination.

3.5 PROJECT REQUIREMENTS

3.5.1 Datasets

1. MS-COCO 2014 Dataset
2. LSUN Dataset
3. CUB 2010-11 UCSD Dataset

3.5.2 Functional Requirements

1. Text descriptions must be written in english language.
2. Text descriptions should not contain any slang or typos.
3. Permissions for using the microphone of the device.
4. Text descriptions should contain more details about the main object or scene whose image is to be generated as compared to irrelevant details that will not contribute to information regarding the object.

3.5.3 Non-Functional Requirements

1. We do not own copyrights of the images.
2. It must be run on a dual-core processors.
3. Use of full-screen browser window should be avoided.

3.5.4 Hardware Requirements

1. VRAM: 8GB
2. Memory Bandwidth: 256 Gbs/second
3. Processing Power: 1920 cores@ 1683 MHz (~ 3,23 M CUDA Core Clocks)
4. GTX 1070 or higher
5. System Ram: 8 GB
6. CPU: Intel i7

3.5.5 Software Requirements

1. Tensorflow
2. Python
3. Flask
4. Javascript
5. JQuery
6. AJAX

3.6 PROJECT PLAN

3.6.1 Project Resources

1. Training of GAN model require high computational power with CUDA configured on GPU.
2. Server for deployment of the web application requires Flask installed on it.
3. Tensorflow libraries, especially tf.nn for creating the neural networks and tf.conv2D for designing the convolutional neural network.

3.6.2 Module Split-up

1. Text Embeddings Generation
2. Development of Stage 1 GAN - Discriminator
3. Development of Stage 1 GAN - Generator
4. Development of Stage 2 GAN
5. Development of Chatbot UI

3.6.3 Functional Decomposition

1. Text Embeddings Generation
    a. Input text description
    b. Generate word embeddings

2. Development of Stage 1 GAN - Discriminator
    a. Formation of Neural Network
    b. Optimization methods for weights and bias
    c. Input text embedding and Image
    d. Output - class label

3. Development of Stage 1 GAN - Generator
    a. Formation of Neural Network
    b. Optimization methods for weights and bias
    c. Input text embedding
    d. Output - Low Resolution Image

4. Development of Stage 2 GAN
    a. Formation of Neural Network
    b. Optimization methods for weights and bias
    c. Input text embedding
    d. Output - High Resolution Image

5. Development of Chatbot UI
    a. Front End Web Development
    b. Speech to Text Processing
    c. Rendering of generated image from model.

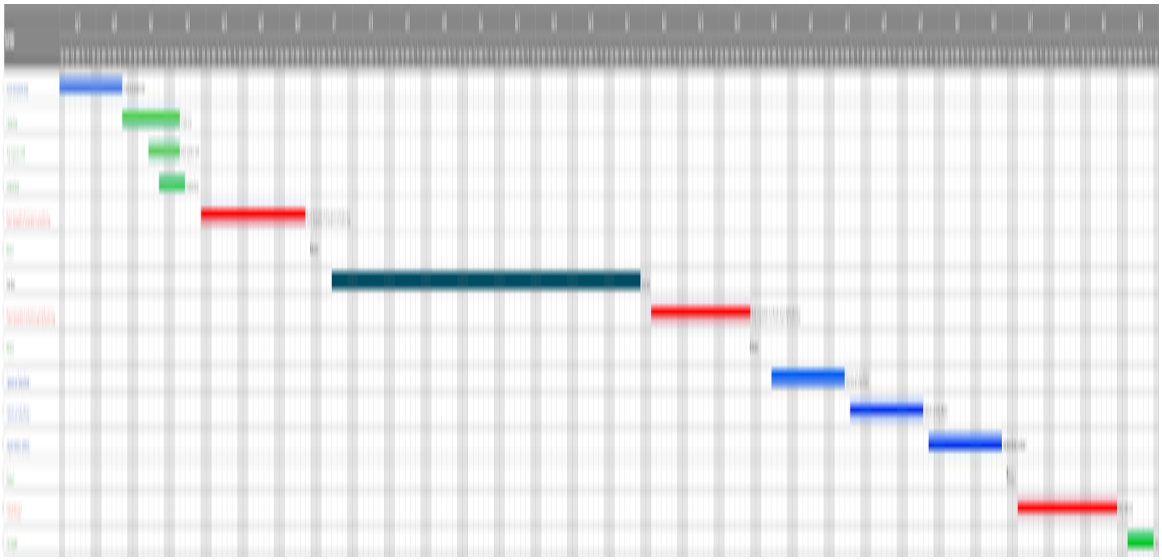
3.6.4 Project Team Role and Responsibilities

1. Project Team: The group responsible for conducting project activities. Project team members are fixed. The project team can contain internal staff as well as outside vendors

2. Client: The person or group requesting the project. The person or group for whom the project is being completed. The client is responsible for articulating the details of their project request: goals, objectives, specifications and outcomes. After the project is completed, the client becomes the owner of the product produced by the project.

3. Stakeholders: Stakeholders are often members of the project team. However, stakeholders should also be understood as any other person/group whose interests may be impacted by a project or its deliverables. Stakeholders should be identified during the initial stages of a project, should be consulted at various stages a project.

3.6.5 Project Plan 3.0



CHAPTER 4
SYSTEM ANALYSIS AND DESIGN

4.1 ARCHITECTURE

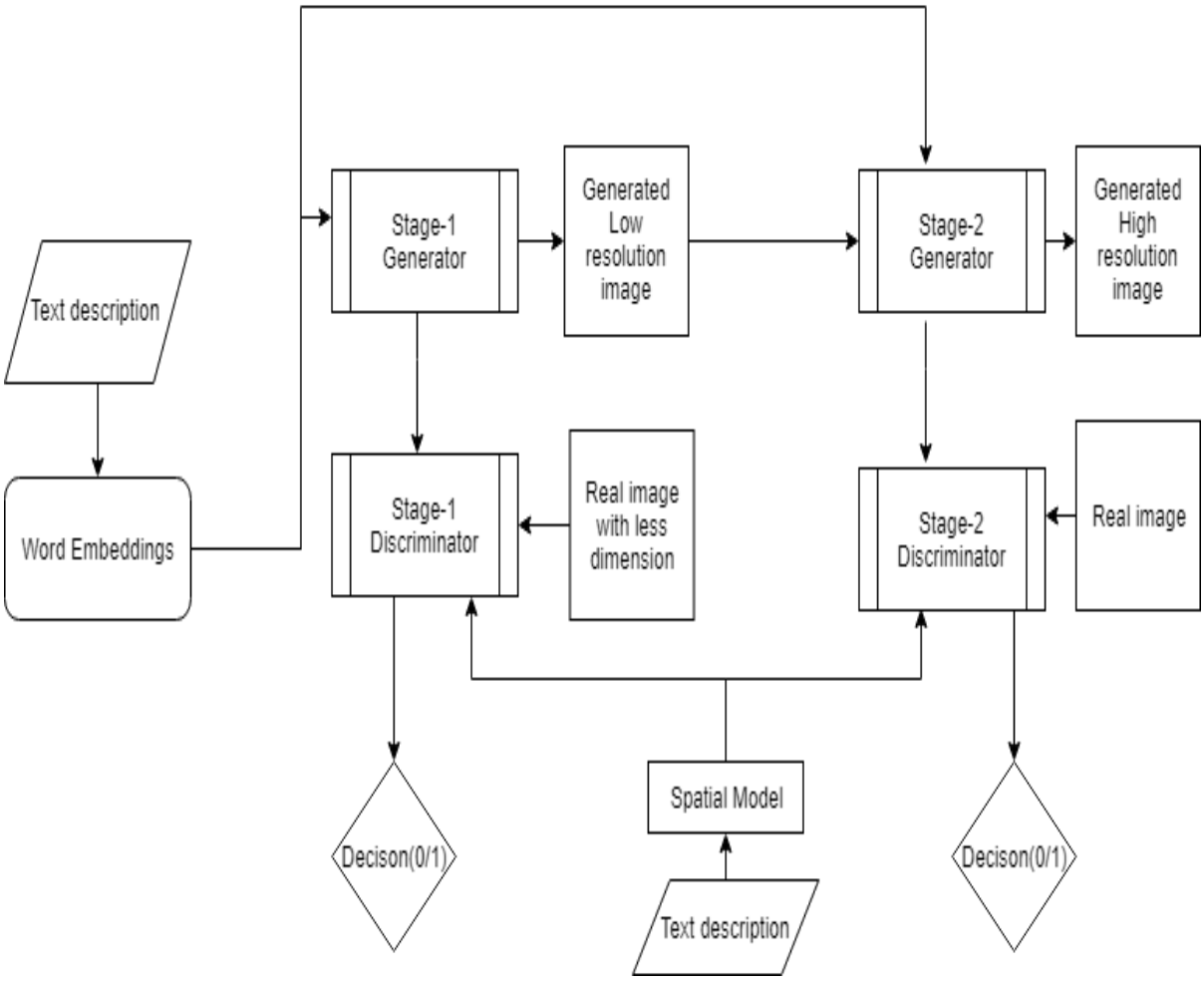


Fig. 1: Architecture Diagram

4.2 BEHAVIOURAL DIAGRAM

4.2.1 Use Case Diagram

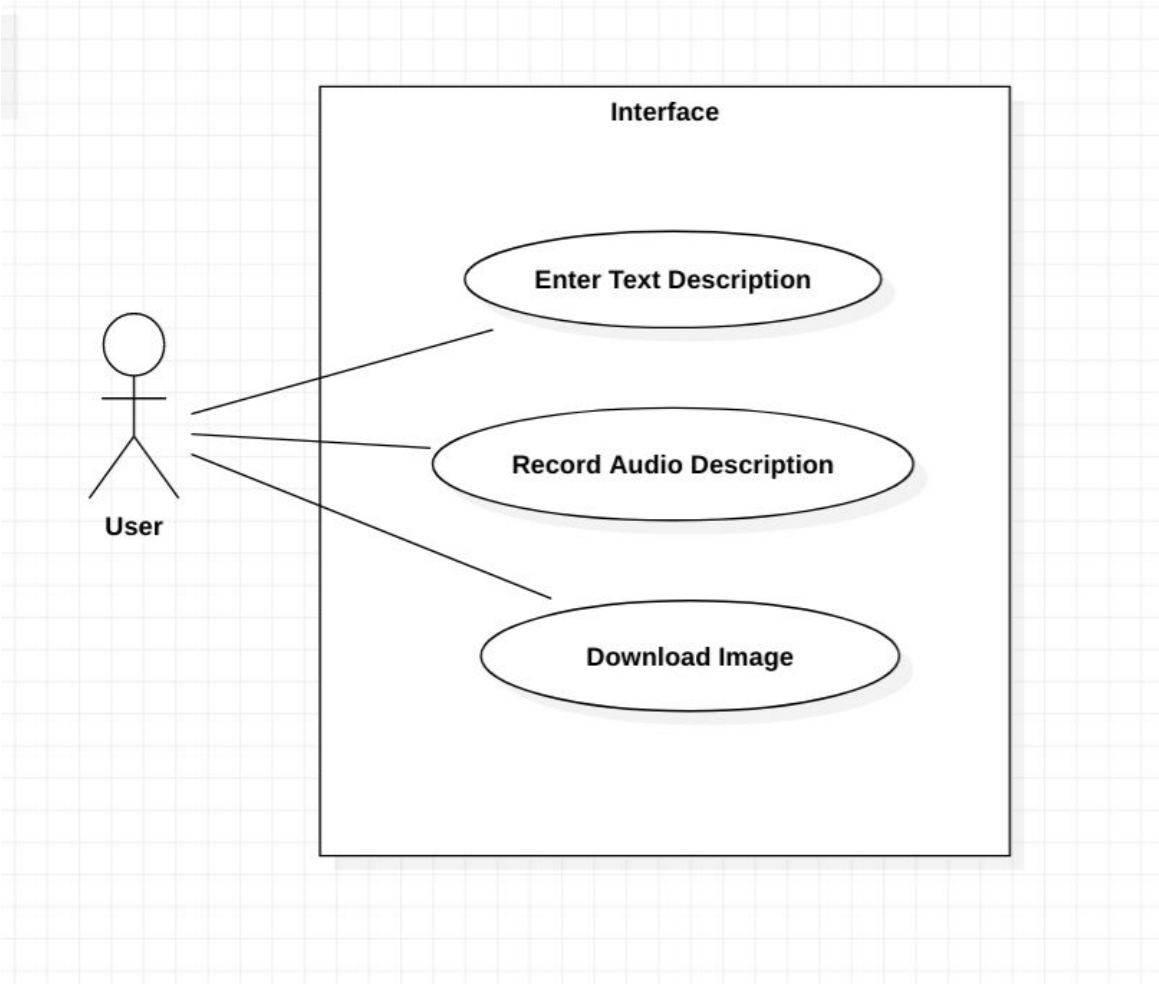


Fig. 2: Use Case Diagram

4.2.2 Activity Diagram

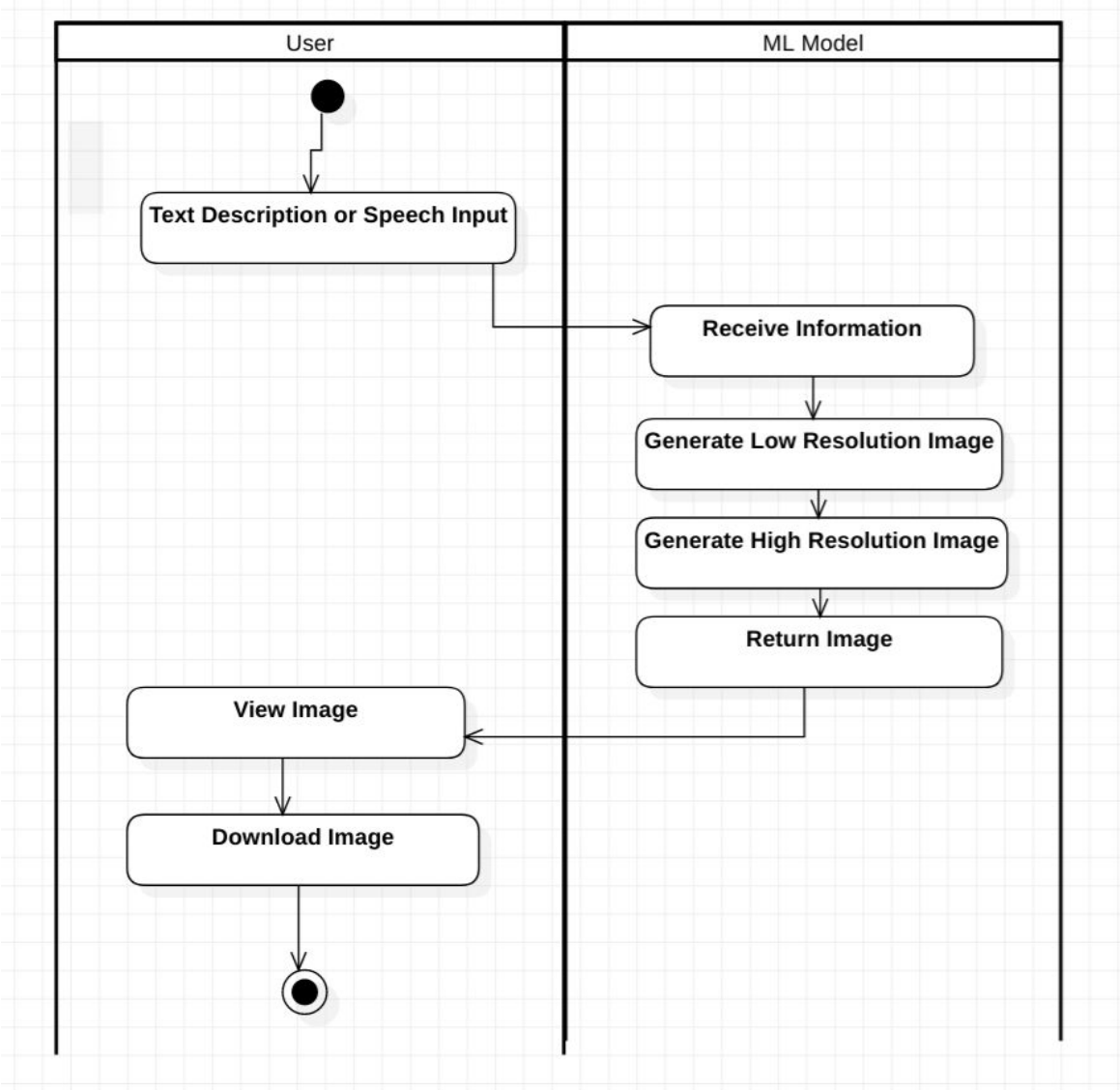


Fig. 3: Activity Diagram

4.2.3 Sequence Diagram

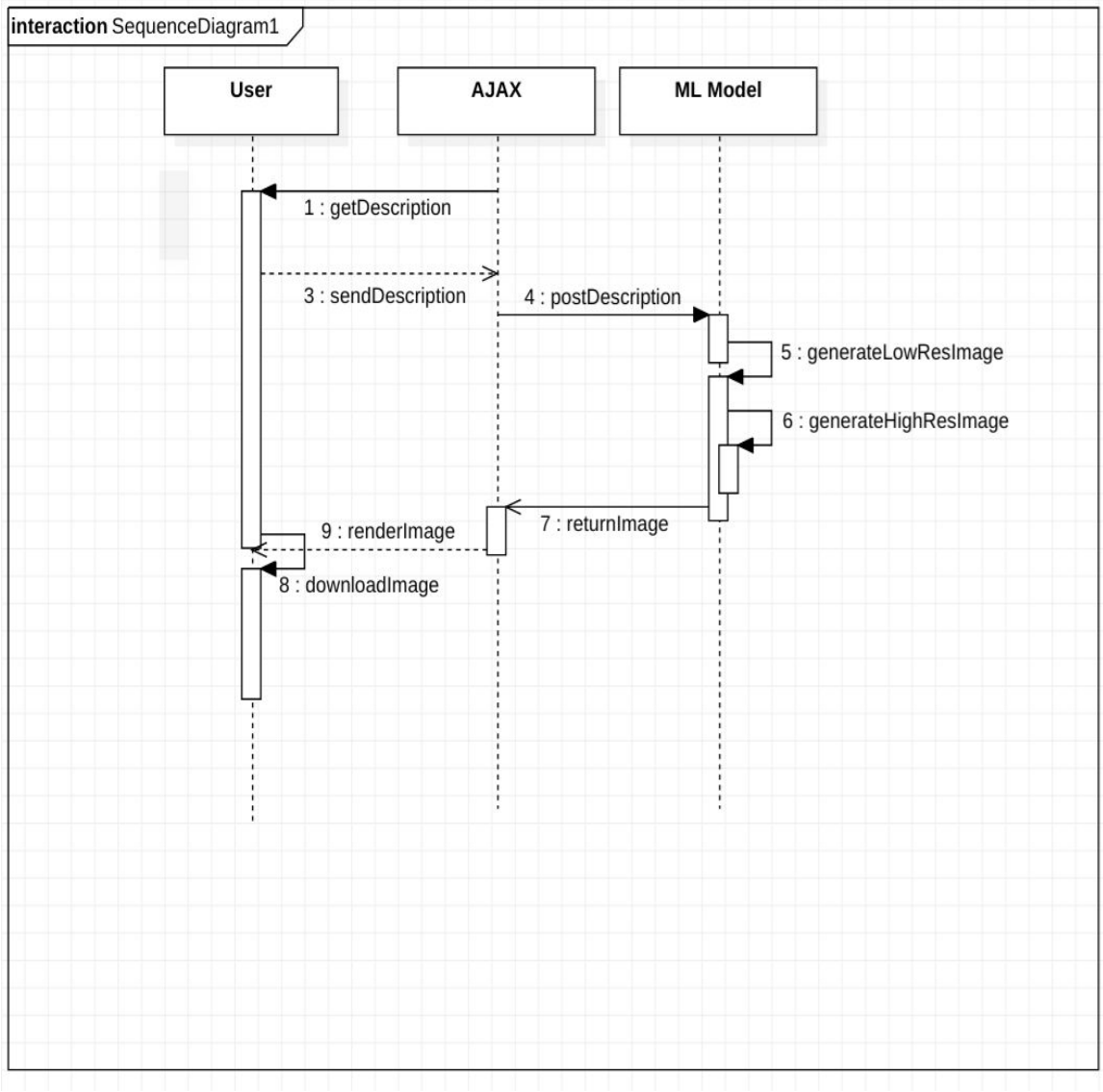


Fig. 4: Sequence Diagram

4.3 Algorithms and Methodologies

1. Generative Adversarial Networks

   Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other. GANs' potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is impressive – poignant even. Discriminative algorithms try to classify input data; that is, given the features of an instance of data, they predict a label or category to which that data belongs. However, generative algorithms do the opposite. Instead of predicting a label given certain features, they attempt to predict features given a certain label. Discriminative models learn the boundary between classes and Generative models model the distribution of individual classes.

   Here are the steps a GAN takes:

   - The generator takes in random numbers and returns an image.
   - This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.
   - The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

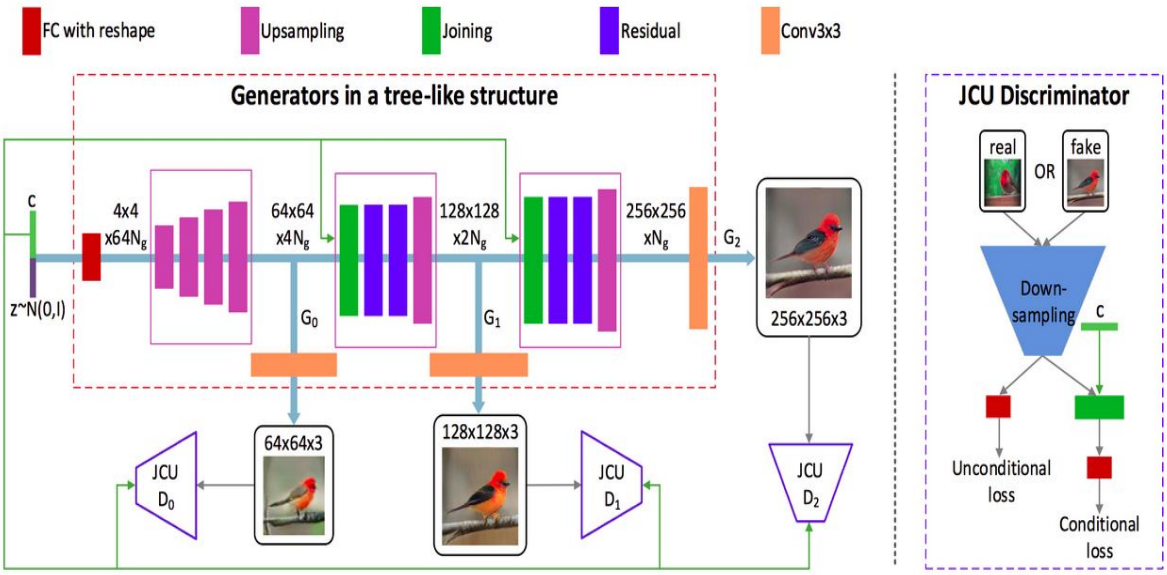


Fig. 5: GAN Architecture

2. Convolution Neural Networks

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images (e.g. name what they see), cluster them by similarity (photo search), and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data. Convolutional networks perform optical character recognition (OCR) to digitize text and make natural-language processing possible on analog and hand-written documents, where the images are symbols to be transcribed. CNNs can also be applied to sound when it is represented visually as a spectrogram. The efficacy of convolutional nets (ConvNets or CNNs) in image recognition is one of the main reasons why the world has woken up to the efficacy of deep learning. They are powering major advances in computer vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired.

3. Char-CNN-RNN Embeddings
    - It define a list a characters (i.e. m). For example, can use alphanumeric and some special characters. For my example, English characters (52), number (10), special characters (20) and one unknown character, UNK. Total 83 characters.
    - Transfer characters as 1-hot encoding and got a sequence for vectors. For unknown characters and blank characters, use all-zero vector to replace it. If exceeding predefined maximum length of characters (i.e. l), ignoring it. The output is 16 dimensions vector per every single character.
    - Using 3 1D CNN layers (configurable) to learn the sequence

4. ReLU

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. So it can be written as f(x)=max(0,x). It's surprising that such a simple function (and one composed of two linear pieces) can allow your model to account for non-linearities and interactions so well. But the ReLU function works great in most applications, and it is very widely used as a result.

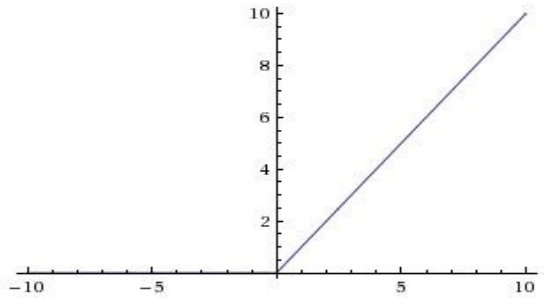


Fig. 6: ReLU

5. Batch Normalization

   We normalize the input layer by adjusting and scaling the activations. For example, when we have features from 0 to 1 and some from 1 to 1000, we should normalize them to speed up learning. If the input layer is benefiting from it, why not do the same thing also for the values in the hidden layers, that are changing all the time, and get 10 times or more improvement in the training speed.

   Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). Batch normalization allows each layer of a network to learn by itself a little bit more independently of other layers.We can use higher learning rates because batch normalization makes sure that there's no activation that's gone really high or really low. And by that, things that previously couldn't get to train, it will start to train.

   It reduces overfitting because it has a slight regularization effects. Similar to dropout, it adds some noise to each hidden layers activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information. However, we should not depend only on batch normalization for regularization; we should better use it together with dropout.

CHAPTER 5
IMPLEMENTATION

5.1 STAGES OF IMPLEMENTATION

5.1.1 Preparation Data

1. MS-COCO 2014 Dataset
2. Removing irrelevant objects from dataset
3. Removing or Completing any missing text descriptions
4. Removing incorrect data or noise

5.1.2 Processing

5.1.2.1 Implementation Part 1: Generation of word vectors

Input is given to the system in form of textual description of a scenario. The text description is converted into the word embedding form, which is the vector representation of the description as required by GANs. For word embeddings, char CNN-RNN model has been used. The word embeddings are further given as an input to the Generative Adversarial Networks(GANs) in two stages which will detect the features from text description and generate an appropriate image in two stages.

5.1.2.2 Implementation Part 2: Generation of a low-resolution image

The Stage-I GAN sketches the primitive shape and colors of a scene based on a given text description, yielding low-resolution images. The background of this Stage-1 image is generated from noise if no proper information is given on background. Also, some of the features might not be present in this low-resolution image. Discriminator in the Stage-1 image validates it if the image is newly

generated by the generator or it is identical to the one in training set with the help of spatial model and real image. For training this model i.e., Model 1, datasets of Birds, ImageNet, and LSUN datasets are used which contain labeled images. This Model 1 is capable of generating the images from new text description.

5.1.2.3 Implementation Part 3: Generation of high-resolution image

The Stage-II GAN takes a low-resolution image from the Stage-I result and the text description as inputs and generates high-resolution images with photo-realistic details. The missing features and the defects in the low resolution images are recovered in this stage. High-resolution image generated by the generator is again validated by the discriminator at stage-2 which helps to know if the generated image has the same features as in the training set with different values.

5.1.3 Integration with User Interface

5.1.3.1 Implementation Part 1: Extraction of text from speech input

The flask web application uses speech processing techniques in python to extract the relevant text from the speech input provided by the user. Appropriate text descriptions are generated with excellent accuracy by using the speech processing libraries in the python language.

5.1.3.2 Implementation Part 2: Passing of text extracted to GAN model

The text descriptions generated by the techniques are then passed to the model through the AJAX calls.

5.1.3.3 Implementation Part 3: Rendering of Generated Image back to Interface

The image that is generated by the model is then rendered as a result of the AJAX call too. The url of the image is obtained on success of the AJAX function resulting with the image being shown to the user in the same interface.

5.2 ELABORATE IMPLEMENTATION OF ISSUES/TECHNIQUES/ SOFTWARE TOOLS

1. Tensorflow

   TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. It helps in Easy model building, Robust ML production anywhere and Powerful experimentation for research. Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

2. PyTorch

   An open source deep learning platform that provides a seamless path from research prototyping to production deployment. Its key features include Hybrid Front-End, Distributed Training. A new hybrid front-end seamlessly transitions between eager mode and graph mode to provide both flexibility and speed. Scalable distributed training and performance optimization in research and production is enabled by the torch.distributed backend. Deep integration into Python allows popular libraries and packages to be used for easily writing neural network layers in Python. A rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.

3. Python

   Python is an interpreted, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations.

4. Anaconda

   Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

5. Flask

   Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

CHAPTER 6
RESULTS AND EVALUATION

6.1 EXPERIMENTS (ALGORITHMIC)

6.1.1 Detail Discussion Of Experiments Carried Out

We studied the sample datasets on which this application can be implemented. We have carried out this experiment on birds dataset (CUB_200_2011) since this dataset was easily available online and suitable for our deep learning model. We implemented Neural Networks for our training and testing model using PyTorch framework and Tensorflow for python 2.7. This experiment was performed on the NVIDIA GPU GTX 1080. We further used CNN-RNN model to convert the text description into the word embedding format as required by the deep learning model.

We further trained our model on an entirely new dataset of MS-COCO, officially provided by Microsoft. It consists of images with their respective annotations for more than 40 categories in all. Our architecture of Generative Adversarial Networks is currently trained to detect many features such as background information, details like car, horse, car, beach, grass field, and many other features.

We have wrapped our machine learning model inside a chatbot type voice assistant web-app. Flask is used to support the backend of the web-application and manage the server connectivity. Use of this model was appropriate for our application because the model is trained in python, and flask is a framework of python, hence importing the model was an easy task.

6.1.2 Results Of Experiments

We have discovered and collected the images and captioned them according to our application. We have obtained an exceptional accuracy on the birds dataset while a slightly less accuracy on the MS-COCO Dataset. During the image generation phase we encountered some errors which were affecting the accuracy of the model. We used hyperparameter tuning to optimize and improve the accuracy of the model. The image generated is generated in the two stages. Output of the every stage can seen separately. We further plan to fine tune the parameter and extend the application for general object. Even if the user enter a copied text description in english from a website, our model is capable to generate the corresponding image from it, which can be downloaded and can be used in further application as and when required.

6.2 Testing

6.2.1 White Box

6.2.1.1 Unit Testing

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within

that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

1. User Interface

   User interface for this web application is built using Flask framework, Javascript and normal front end technologies using HTML, CSS. We tried accessing this web application on different platforms with different features like screen size and the resolution.
   It worked fine for Galaxy S5, Pixel 2, iPhone 5, iPhone 6, iPhone X, iPad, iPad Pro.

2. Server Model

   The server is established using Flask application. To deploy this web application, the web server must have Flask module installed. Otherwise, it is not possible to deploy the app.

3. GAN model

   The GAN model is trained on the MS-COCO Dataset. Due to the large volume of the categories in the dataset, it is difficult to train the model with so many features and requires an high computation power. Thus, descriptions that consist of general or comprehensive features such as the sky, beaches, fields, etc. are represented with better accuracy in our application, while the finer details of the object cannot be ascertained by the model due to the complexity of the scene description. This problem can be rectified with further training and a more compatible dataset..

6.2.1.2 Integration Testing

Integration Testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing. Testing performed to expose defects in the interfaces and interaction between integrated components. Testing the integration of systems and packages; testing interfaces to external organizations.

1. Model Integration

   The deep learning model is integrated with user interface with the help of Flask server model. The Flask framework, being built on top of Python, provides library to import the deep learning model which is in Python itself. We have tested the model by sending the text descriptions from the web-app and the corresponding generated image is returned in the same user interface. This image is rendered as a reply to the user's description. Hence, while in most cases, chatbot interfaces have speech or textual replies, in this particular application, the reply will be in the form of an image.

6.2.2 Black Box

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories: Incorrect or missing functions, Interface errors, Errors in data structures or external database access, Behavior or performance errors, Initialization and termination errors.

6.2.2.1 Test Cases

1. Example-1

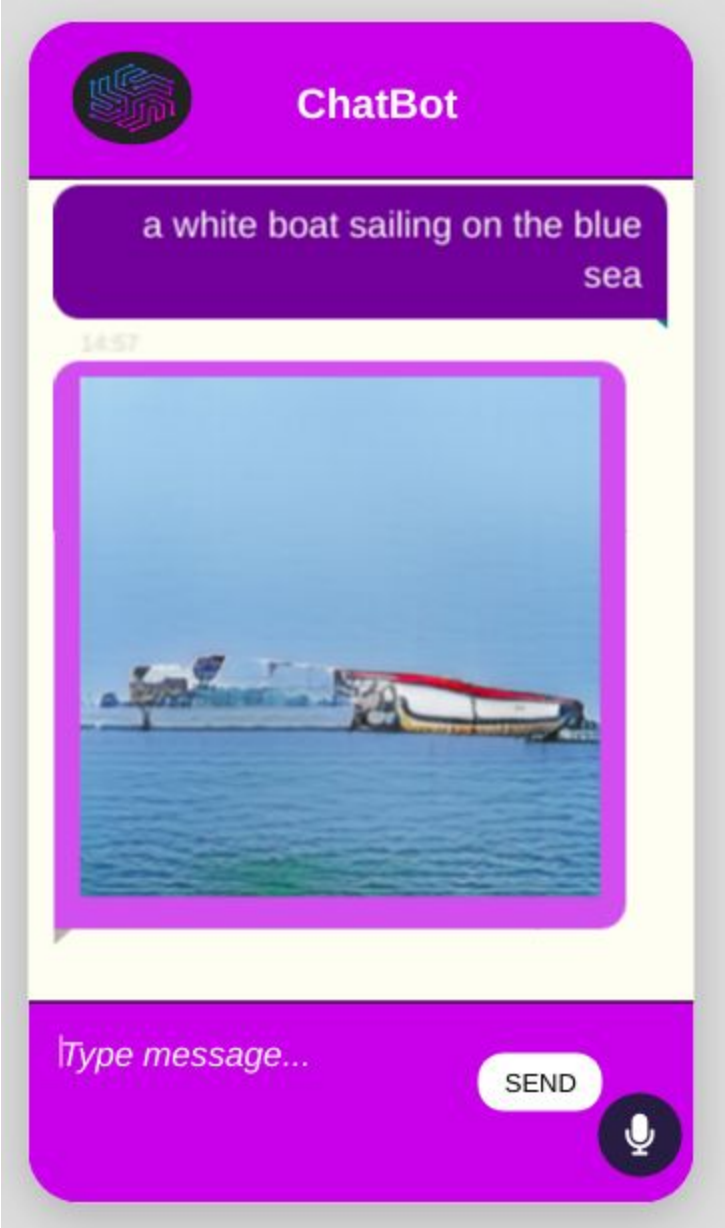


Fig. 7: Output Case 1

2. Example-2

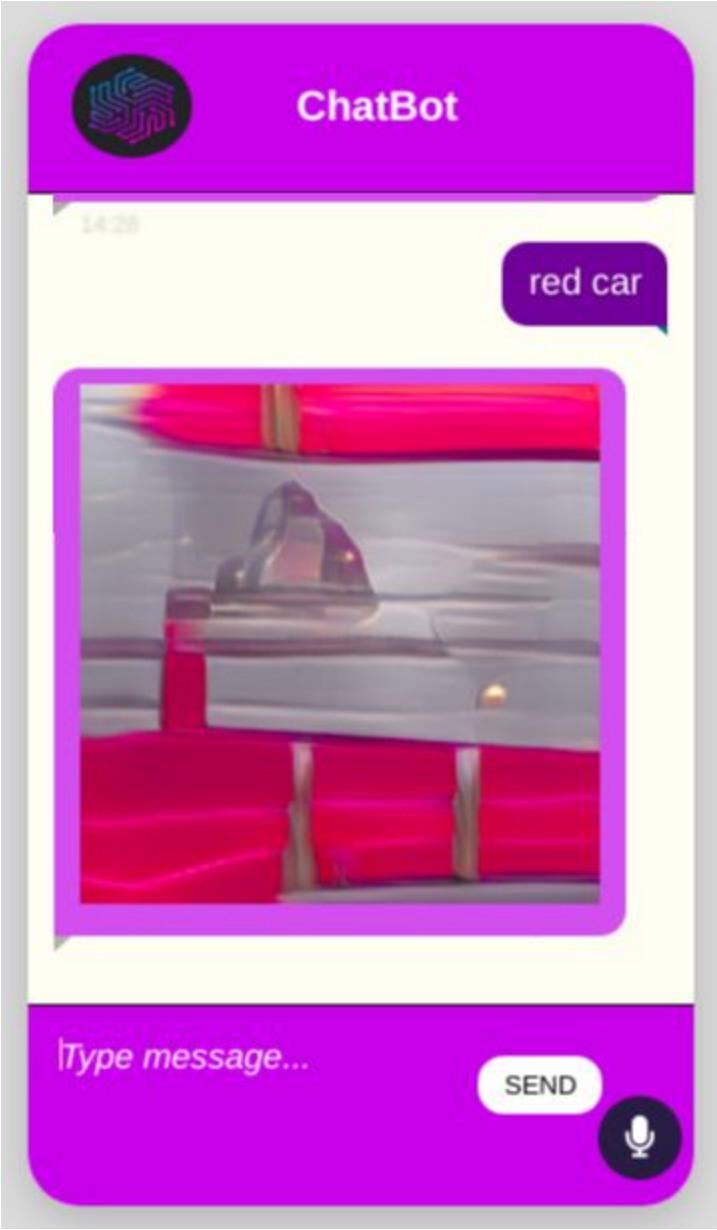


Fig. 8: Output Case 2

3. Example-3

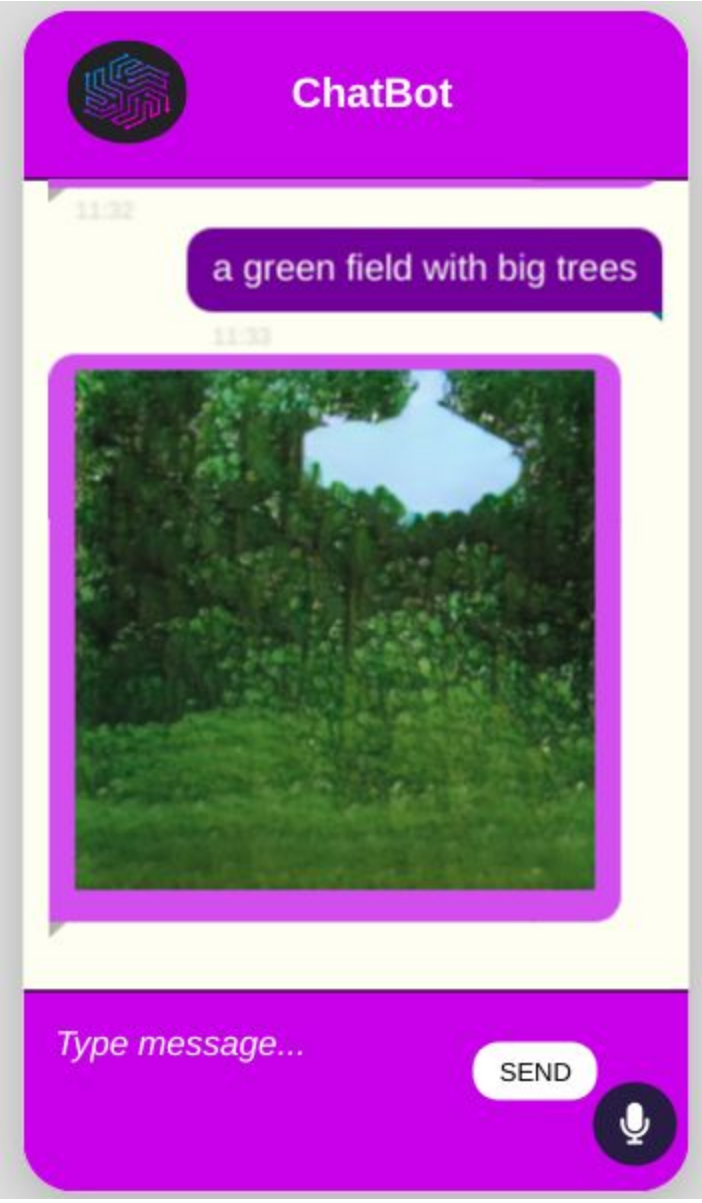


Fig. 9: Output Case 3

4. Example-4

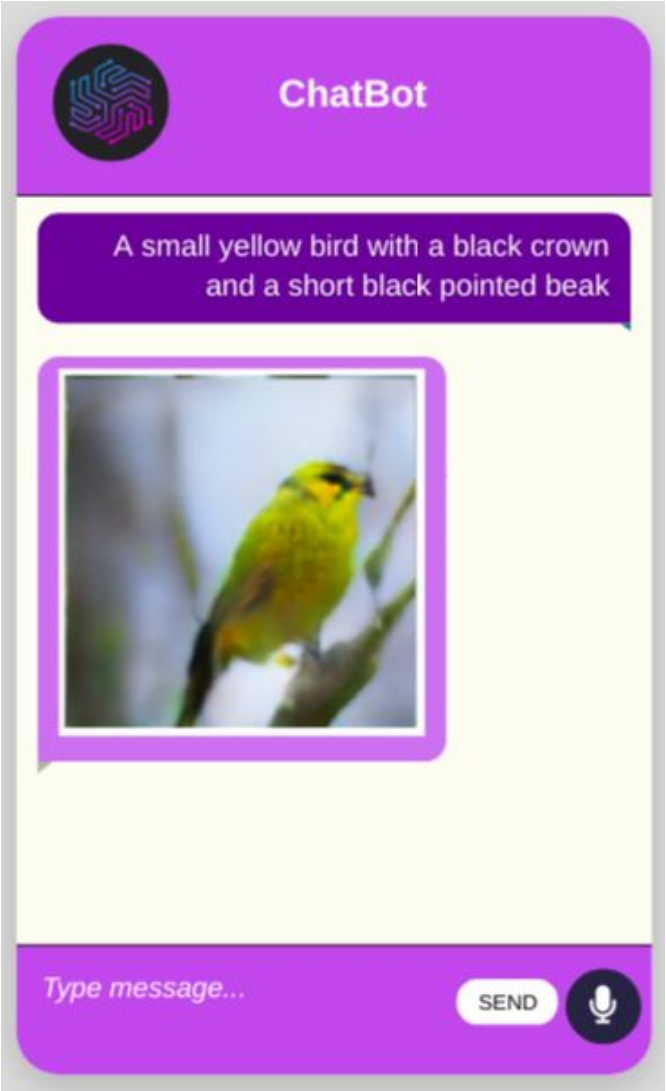


Fig. 10: Output Case 4

5. Example-5

Fig. 11: Output Case 5

6. Example-6

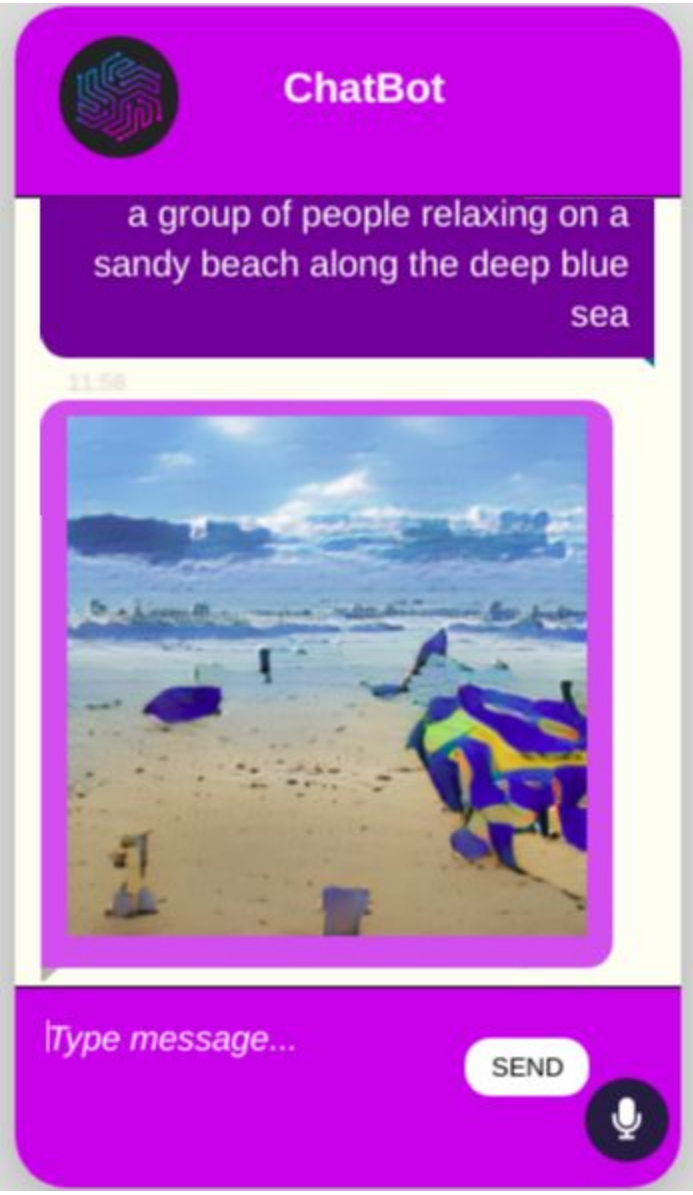


Fig. 12: Output Case 6

7. Example-7

Fig. 13: Output Case 7

8. Example-8

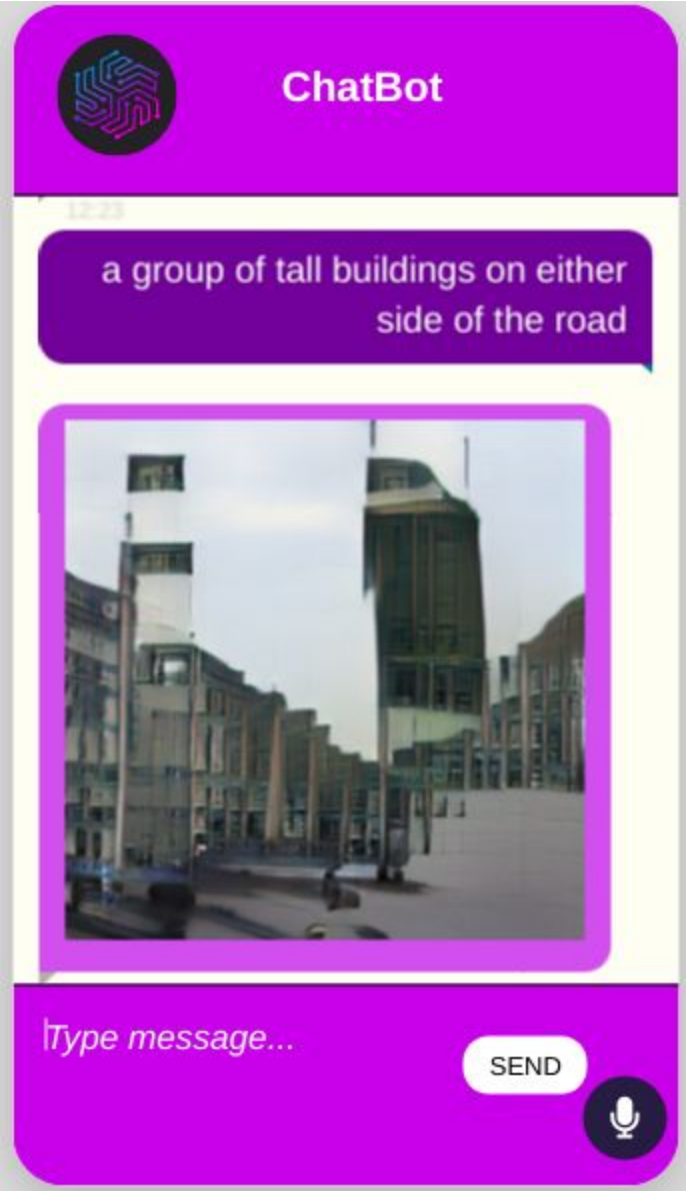


Fig. 14: Output Case 8

6.2.2.2 Summary of Black Box Testing

1. The CUB Dataset provides much more accurate captions for the training images as compared to MS-COCO dataset. Thus, the model trained on CUB dataset is able to detect the finer details in the text descriptions along with the more general features as well. As a result, the model trained on CUB Dataset generated high resolution images with excellent accuracy.
2. However, the model which is trained on the MS-COCO dataset, is less accurate in terms of detecting and generating the finer details of an scenario.
3. The MS-COCO dataset model, is however, able to generate the background image exceptionally well.
4. We have achieved an inception score of 19.89 for MS-COCO and 3.7 for CUB dataset.
5. The expected accuracy of these model were +4.36 for CUB Dataset and 25.89 for MS-COCO model.

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

Images are representation of objects or scenarios in art developed using a collection of pixels. They allow the viewer to visualize a precise scenario as compared to different interpretations from different people. Such a perception of any scenario can greatly amplify our understanding of the basic components in the scenario. The application developed as a part of this study is successfully able to generate high resolution images from text descriptions. The input to the model is a textual description, detailing the scenario and the output is an image based on this text. An application developed using the above technology and methodology is expected to outperform all existing apps in terms of usability and user experience and provide visually appealing results when it comes to the final objective of generation of high quality images from already available or real time generated textual data. Thus, the usage of text can be enhanced and made even more informative and visually coherent with this study.

7.2 LIMITATIONS

The application developed in this project has its dependency of an object dataset only, which also consists of text embeddings. Even though the project model is trained on a comprehensive dataset, with further research, it will be possible to gather data to build an all-inclusive application that goes beyond the range of limited objects. Hence, we aim to increase the scope of visualizing further classes of objects in the given application. There is also capacity to increase the accuracy of this product by utilizing better techniques and more optimized software and hardware resources.

REFERENCES

[1] Ian J. Goodfellow, Jean Pouget-Abadiey, Mehdi Mirza, Bing Xu, David Warde-Farley,Sherjil Ozairz, Aaron Courville, Yoshua Bengiox. Generative Adversarial Nets. In NIPS, 2014.

[2] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. arXiv:1612.03242, 2016.

[3] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Senior Member, IEEE, Xiaogang Wang, Member, IEEE, Xiaolei Huang, Member, IEEE, Dimitris N. Metaxas, Fellow, IEEE. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. arXiv:1710.10916, 2017.

[4] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran,Bernt Schiele, Honglak Lee. Generative Adversarial Text to Image Synthesis. arXiv:1605.05396, 2016.

[5] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks. arXiv:1711.10485, 2017.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016

[8] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked generative adversarial networks. In CVPR, 2017.

[9] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. In ICLR, 2017.

[10] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. In ICLR, 2017

[11] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. arXiv:1702.07983, 2017.

[12] I. P. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. In ICLR, 2017.

[13] J. Gauthier. Conditional generative adversarial networks for convolutional face generation.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In CVPR, 2017.

[15] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In ICLR, 2018.

[16] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In CVPR, 2017.

[17] E. Mansimov, E. Parisotto, L. J. Ba, and R. Salakhutdinov. Generating images from captions with attention. In ICLR, 2016

[18] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. In CVPR, 2017.

[19] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In ICML, 2017.

[20] S. Reed, Z. Akata, B. Schiele, and H. Lee. Learning deep representations of fine-grained visual descriptions. In CVPR, 2016.

[21] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text-to-image synthesis. In ICML, 2016.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015.

[23] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In NIPS, 2016.

[24] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In NIPS, 2016.

[25] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[26] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365, 2015.

[27] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In ICCV, 2017.