SEMANTIC IMAGE SYNTHESIS THROUGH SPADE

Project Report Submitted

In Partial Fulfillment of the

Requirements For the Degree Of

BACHELOR OF

ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

NISHAT TABASSUM (1604-16-733-009)
SYED SAHEEL AHMED (1604-16-733-041)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MUFFAKHAM JAH COLLEGE OF ENGINEERING & TECHNOLOGY
(Affiliated to Osmania University)
Mount Pleasant, 8-2-249, Road No. 3, Banjara Hills, Hyderabad-

34 2020

Date: 30 / 03 / 2020

CERTIFICATE

This is to certify that the project dissertation titled **"SEMANTIC IMAGE SYNTHESIS THROUGH SPADE"** being submitted by

1. NISHAT TABASSUM (1604-16-733-009)
2. SYED SAHEEL AHMED(1604-16-733-041)

in Partial Fulfillment of the requirements for the award of the degree of BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING in MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY, Hyderabad for the academic year 2019-20 is the bonafide work carried out by them. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signatures:

Internal Project Guide   Head CSED

(Mrs. Maniza Hijab)   (Dr. A.A. Moiz Qyser)

(Associate Professor

External Examiner

DECLARATION

This is to certify that the work reported in the major project entitled “SEMANTIC IMAGE SYNTHESIS*”*is a record of the bonafide work done by us inthe Department of Computer Science and Engineering, Muffakham Jah College of Engineering and Technology, Osmania University. The results embodied in this report are based on the project work done entirely by us and not copied from any other source.

1. Nishat Tabassum (1604-16-733-009)

2. Syed Saheel Ahmed (1604-16-733-041)

ACKNOWLEDGEMENT

Our hearts are filled with gratitude to the Almighty for empowering us with courage, wisdom and strength to complete this project successfully. We give him all the glory, honor and praise.

We thank our Parents for having sacrificed a lot in their lives to impart the best education to us and make us promising professionals for tomorrow.

We would like to express our sincere gratitude and indebtness to our project supervisor **Mrs. Maniza Hijab** for her valuable suggestions and interest throughout the course of thisproject.

We are happy to express our profound sense of gratitude and indebtedness to **Prof. Dr.Ahmed Abdul Moiz Qyser,** Head of the Computer Science and Engineering Department, forhis valuable and intellectual suggestions apart from educate guidance constant encouragement right throughout our work and making successful.

With a great sense of pleasure and privilege, we extend our gratitude to **Prof. Dr. SyedShabbeer Ahmed,** Associated Head of Computer Science and Engineering Department,project in-charge, offered valuable suggestions was a pre-requisite to carry out support in every step.

We are pleased to acknowledge our indebtedness to all those who devoted themselves directly or indirectly to make this project work a total success.

NISHAT TABASSUM
SYED SAHEEL AHMED

ABSTRACT

Image synthesis is the process of creating new images from some form of image description.Synthetic images are often used to verify the correctness of operators by applying them to known images.The operator output on such images is generally clean used in Education, CAD ... Previous methods (CRN, pix2pix) directly feed the semantic layout as input to the network, which is then processed through stacks of convolution, normalization, and nonlinearity layers. We show that this is suboptimal because the normalization layers tend to wash away semantic information. To address this issue,we propose Spatially-adaptive normalization, a simple but effective layer for synthesizing photorealistic images given an input semantic layout using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation.

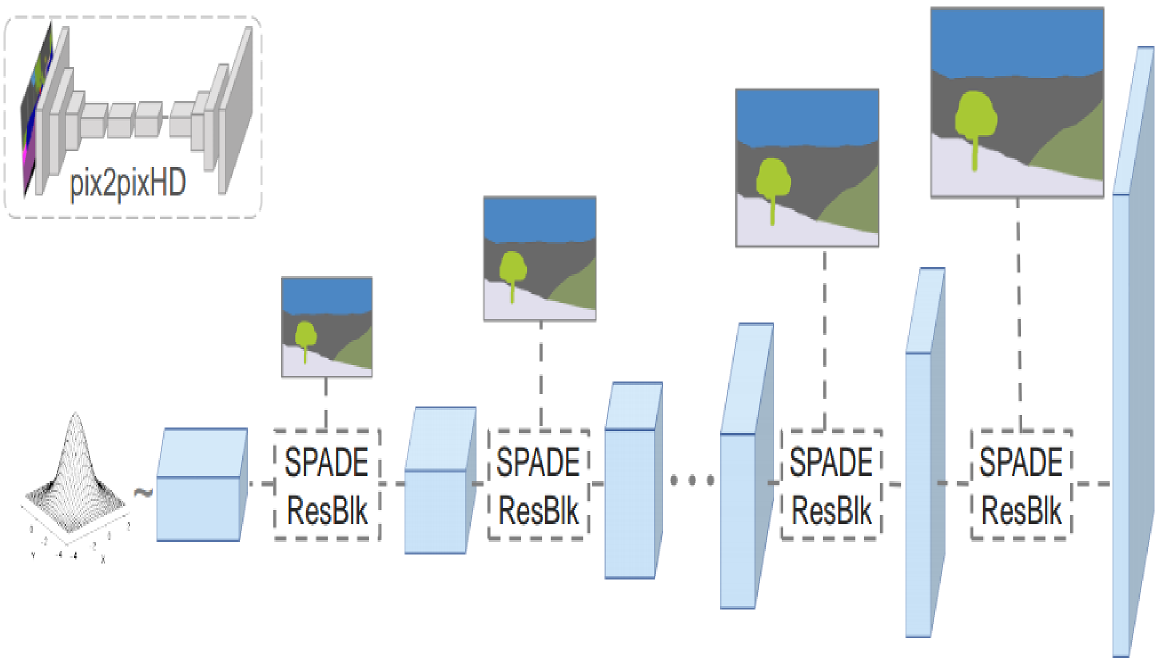
This approach produces an image with photographic appearance that conforms to the input layout. Unlike recent and contemporaneous work, our approach does not rely on adversarial training. We show that photographic images can be synthesized from semantic layouts by a single feedforward network with appropriate structure, trained end-to-end with a direct regression objective.

Both visual fidelity and alignment with input layouts were improved over several datasets when the above approach was used in contrast to existing methodology. Finally, our model allowsusers to easily control the style and content of synthesis results as well as create multi-modalresults. Extensive perceptual experiments on datasets of outdoor and indoor scenes demonstrate thatimages synthesized by the presented approach are considerably more realistic than alternative approaches. We hope that the simplicity of the presented approach can support follow-up work that will further advance realism and explore the applications of photographic image synthesis.

CONTENTS

1.INTRODUCTION

Image synthesis is the process of creating new images from some form of image description.Recently, generative adversarial networks (GAN) [6] have shown stunning results in generating photorealistic images of faces [16, 17] and simple objects [34, 1, 22]. However, generating photorealistic images for complex scenes with different types of objects and stuff remains a challenging problem. We consider semantic image synthesis, which aims at generating photorealistic images conditioned on semantic layouts. It has wide applications on controllable image synthesis and interactive image manipulation. State-of-the-art methods are mostly based on Generative Adversarial Networks (GAN).



Synthetic images are often used to verify the correctness of operators by applying them toknown images. They are also often used for teaching purposes, as the operator output on suchimages is generally `clean&#39;, whereas noise and uncontrollable pixel distributions in realimages make it harder to demonstrate unambiguous results. The images could bebinary, graylevel or color.

1.1 AREA OF OCCURRENCE

- Movies
- Games
- Virtual Reality
- Visualization (Scientific, Medical)
- Lighting Simulation (Interiors, Automobiles, …)
- Computer Aided Design (CAD)

Here the images thus produced are free from noise and are in high resolution which can be used in above applications.

1.2 IMPACT

This approach produces an image with photographic appearance that conforms to the input layout. Unlike recent and contemporaneous work, our approach does not rely on adversarial training. We show that photographic images can be synthesized from semantic layouts by a single feedforward network with appropriate structure, trained end-to-end with a direct regression objective.

we propose **Spatially-adaptive normalization**, a simple but effective layer for synthesizing photorealistic images given an input semantic layout using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation.

1.3TYPES OF IMAGES THAT ARE SYNTHESIZED

- ✓ ***Test Patterns****:*- Scenes with simple two dimensional geometric shapes.
- ✓ ***Image Noise****:*- Images containing random pixel values, usually generated from specific parametrized distributions.
- ✓ ***Computer Graphics****:*- Scenes or images based on geometric shape description. Often the models are 3D, but may also be 2D.

1.4 PROBLEM STATEMENT

The major areas where Image Synthesis is to be addressed are-

- Compression
- Degradation
- Recognition
- Visualization

In addition to above problems,-The need to use an external database of images ,stitching those images together and performing image processing is quite difficult.

1.5 OBJECTIVE

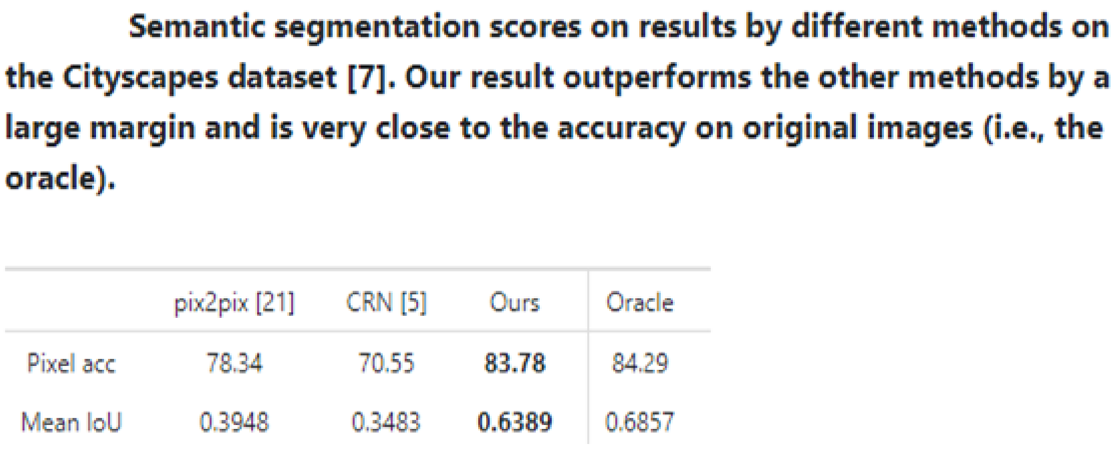
Semantic image synthesis aims at generating photorealistic images from semantic layouts. we propose, using the input layout for modulating the activations in normalization layers through a spatially- adaptive, learned transformation. We are able to maintain both visual fidelity and alignment with input layouts.
The performance metric parameters which justify for successful results are also calculated which were producing desired outcomes with (mIou) and pixel accuracy with higher rates compared to (Fid) value.
Finally, our model allows user control over both semantic and style as synthesizing images.

2.LITERATURE SURVEY

2.1 Existing Approaches:

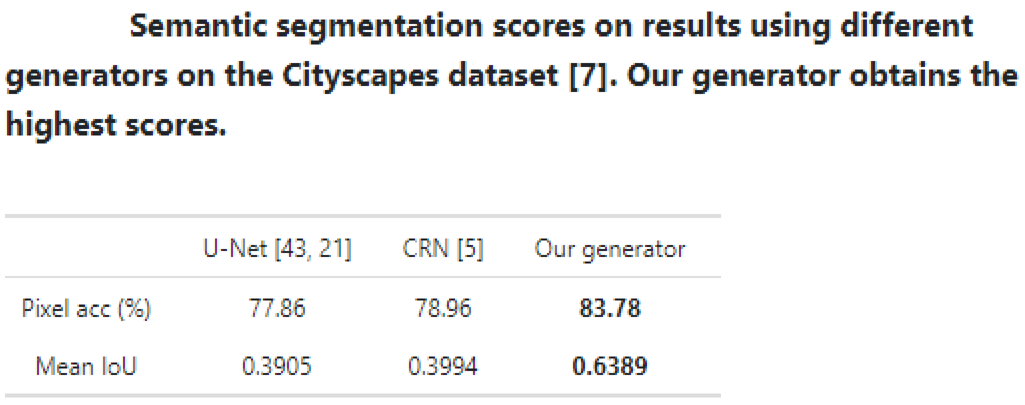
2.1.1 pix2pix:

- We investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping
- This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations.



Semantic segmentation scores on results by different methods on the Cityscapes dataset [7]. Our result outperforms the other methods by a large margin and is very close to the accuracy on original images (i.e., the oracle).

	pix2pix [21]	CRN [5]	Ours	Oracle
Pixel acc	78.34	70.55	**83.78**	84.29
Mean IoU	0.3948	0.3483	**0.6389**	0.6857



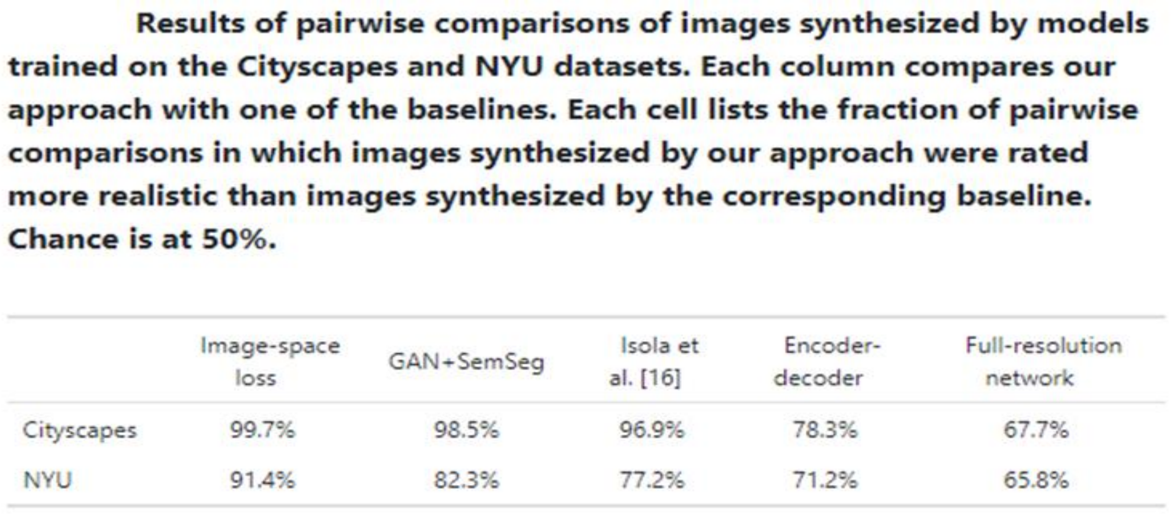
Semantic segmentation scores on results using different generators on the Cityscapes dataset [7]. Our generator obtains the highest scores.

	U-Net [43, 21]	CRN [5]	Our generator
Pixel acc (%)	77.86	78.96	**83.78**
Mean IoU	0.3905	0.3994	**0.6389**

Figure:1

2.1.2 CRN:

- Given a semantic label map, our approach produces an image with photographic appearance that conforms to the input layout. The approach thus functions as a rendering engine that takes a two-dimensional semantic specification of the scene and produces a corresponding photographic image.
- The presented approach scales seamlessly to high resolutions; we demonstrate this by synthesizing photographic images at 2-megapixel resolution, the full resolution of our training data.



Results of pairwise comparisons of images synthesized by models trained on the Cityscapes and NYU datasets. Each column compares our approach with one of the baselines. Each cell lists the fraction of pairwise comparisons in which images synthesized by our approach were rated more realistic than images synthesized by the corresponding baseline. Chance is at 50%.

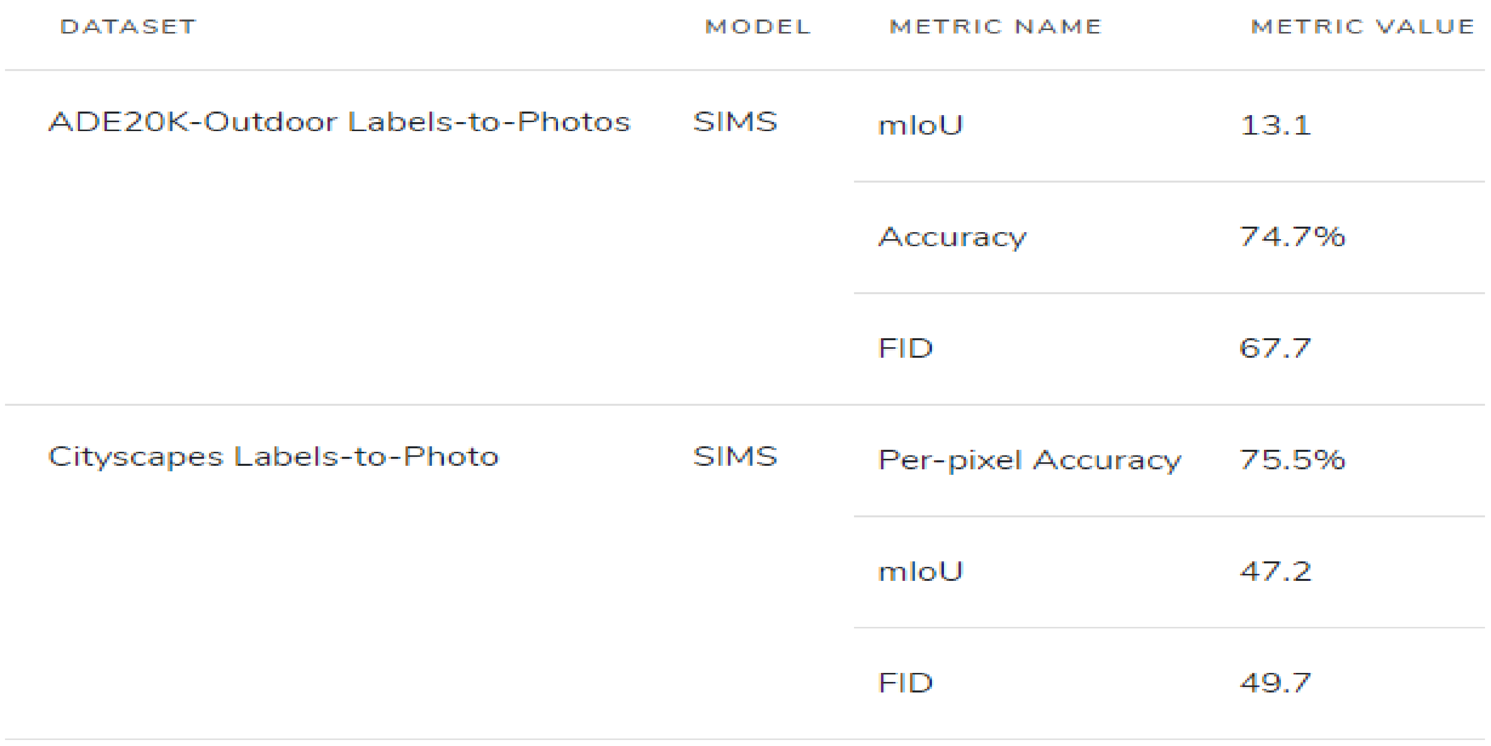
	Image-space loss	GAN+SemSeg	Isola et al. [16]	Encoder-decoder	Full-resolution network
Cityscapes	99.7%	98.5%	96.9%	78.3%	67.7%
NYU	91.4%	82.3%	77.2%	71.2%	65.8%

Figure:2

2.1.3 SIMS:

- The approach combines the complementary strengths of parametric and nonparametric techniques. The nonparametric component is a memory bank of image segments constructed from a training set of images.
- The synthesis is performed by a deep network that draws on the provided photographic material. Experiments on multiple semantic segmentation datasets show that the presented approach yields considerably more realistic images than recent purely parametric techniques.

The below table shows different results of performance metric parameters which were conducted on different Datasets by SIMS approach.



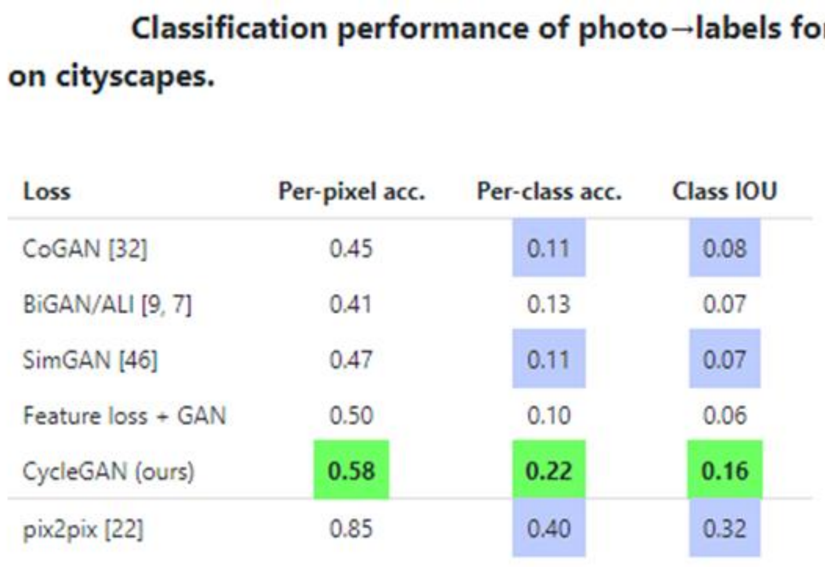
DATASET	MODEL	METRIC NAME	METRIC VALUE
ADE20K-Outdoor Labels-to-Photos	SIMS	mIoU	13.1
		Accuracy	74.7%
		FID	67.7
Cityscapes Labels-to-Photo	SIMS	Per-pixel Accuracy	75.5%
		mIoU	47.2
		FID	49.7

Figure:3

2.1.4 CycleGAN:

- Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs
- Our goal is to learn a mapping G:X→YG:X→Y such that the distribution of images from G(X)G(X) is indistinguishable from the distribution YY using an adversarial loss. Because this mapping is highly under-constrained, we couple it

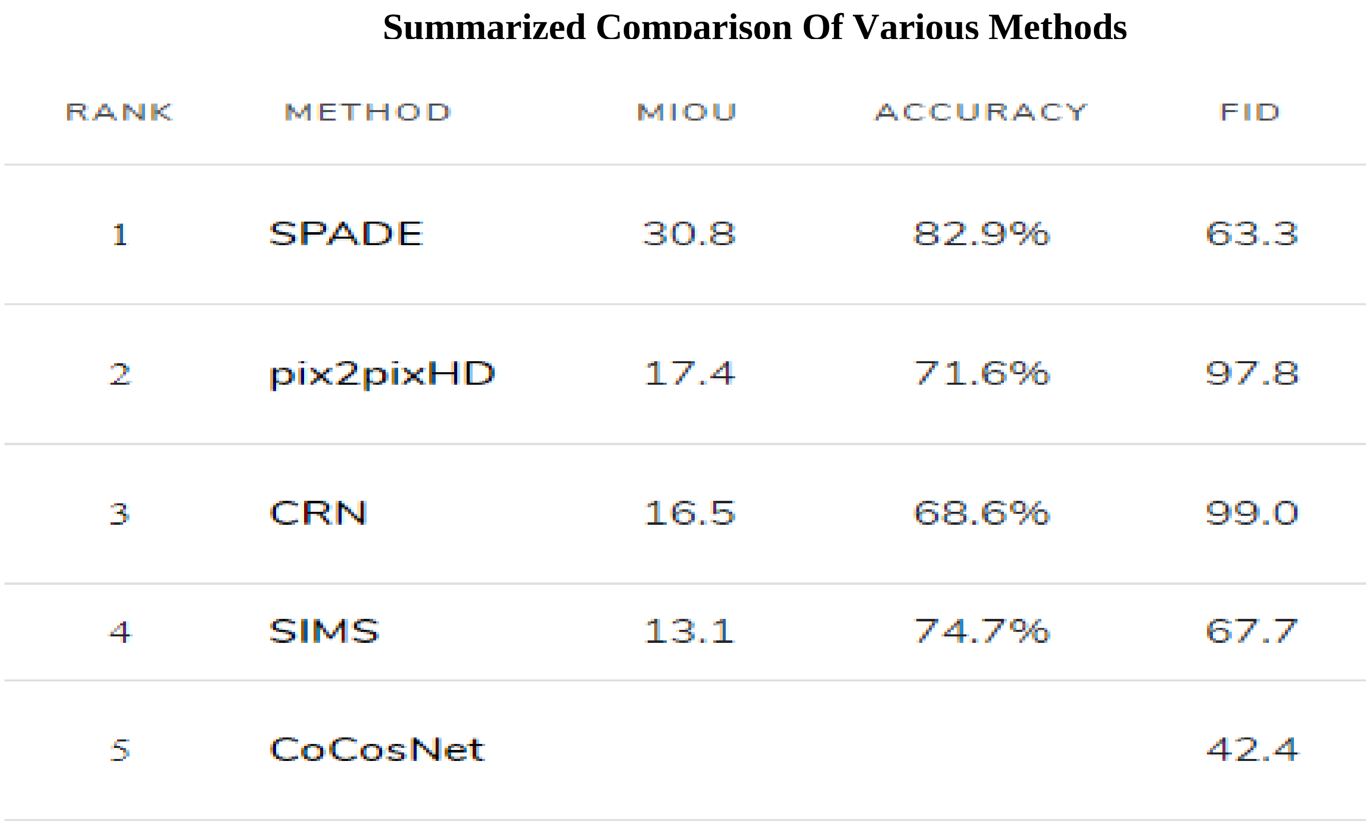
  with an inverse mapping F:Y→XF:Y→X and introduce a cycle consistency loss to push F(G(X))≈XF(G(X))≈X (and vice versa)



Classification performance of photo→labels for different methods on cityscapes.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.45	0.11	0.08
BiGAN/ALI [9, 7]	0.41	0.13	0.07
SimGAN [46]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [22]	0.85	0.40	0.32

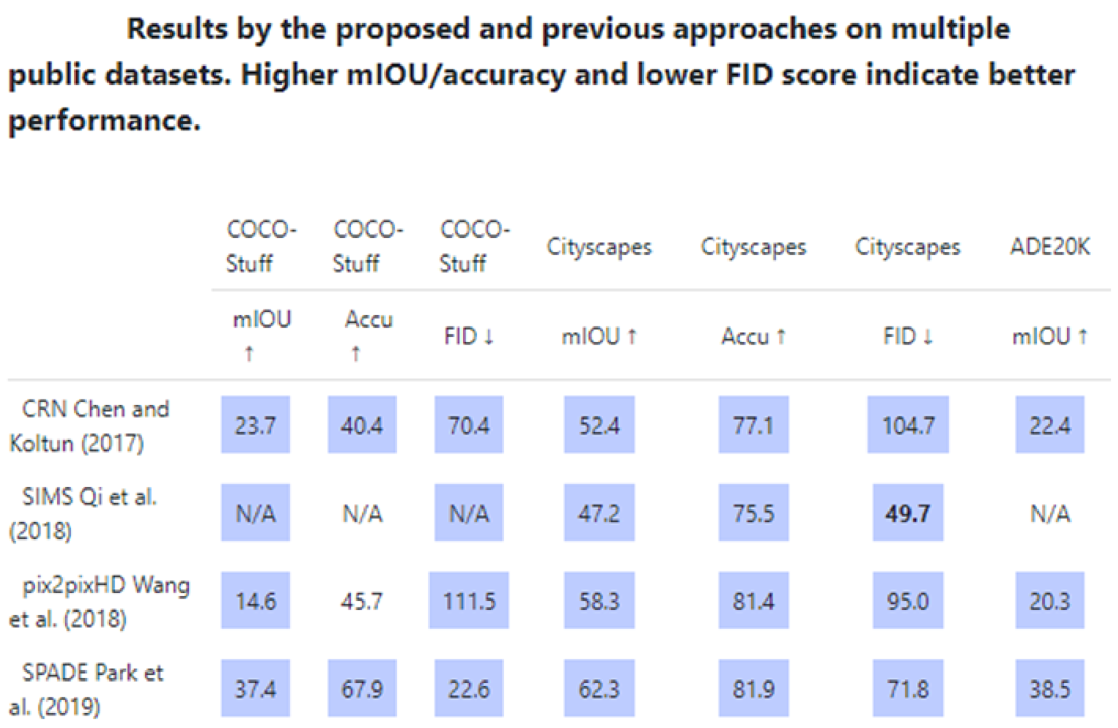
Figure:4



Summarized Comparison Of Various Methods

RANK	METHOD	MIOU	ACCURACY	FID
1	SPADE	30.8	82.9%	63.3
2	pix2pixHD	17.4	71.6%	97.8
3	CRN	16.5	68.6%	99.0
4	SIMS	13.1	74.7%	67.7
5	CoCosNet			42.4

Figure:5



Results by the proposed and previous approaches on multiple public datasets. Higher mIOU/accuracy and lower FID score indicate better performance.

	COCO-Stuff	COCO-Stuff	COCO-Stuff	Cityscapes	Cityscapes	Cityscapes	ADE20K
	mIOU ↑	Accu ↑	FID ↓	mIOU ↑	Accu ↑	FID ↓	mIOU ↑
CRN Chen and Koltun (2017)	23.7	40.4	70.4	52.4	77.1	104.7	22.4
SIMS Qi et al. (2018)	N/A	N/A	N/A	47.2	75.5	**49.7**	N/A
pix2pixHD Wang et al. (2018)	14.6	45.7	111.5	58.3	81.4	95.0	20.3
SPADE Park et al. (2019)	37.4	67.9	22.6	62.3	81.9	71.8	38.5

Figure:6

3.SYSTEM ANALYSIS

3.1. Problems with Existing System

Compression: A modern trend in image storage and transmission is to use digital techniques. Digitizing a television signal results in -100 megabits per second. But channel bandwidth is expensive. So for applications such as teleconferencing, one wants to use a channel of 64 kilobits per second.

Degradation: In enhancement, one aims to process images to improve their quality. An image may be of poor quality because its contrast is low, or it is noisy, or it is blurred. Noise-reduction algorithms typically involve local, averaging or smoothing which unfortunately, will blur the edges in the image. Adaptive methods have been investigated-e.g., smoothing less near the edges.

Recognition: Typically, a recognition system needs to classify an unknown input pattern into one of a set of pre-specified classes. The problem can become very difficult if the number of classes is very large or if members in the same class can look very different.

Visualization: Commonly, visualization is considered as a part of computer graphics. The main task is to generate images or image sequences based on three-dimensionalobject and scene models. A challenging problem is how to model dynamic scenes containing non rigid objects (as clothing, hair).

A GAN-based image synthesis model that can generate photo-realistic images given an input semantic layout. It is built on spatially-adaptive normalization, a simple but effective normalization layer. Previous methods directly feed the semantic layout as input to the deep network, which is then processed through stacks of convolution, normalization, and non-linearity layers. We show that this is sub-optimal as the normalization layers tend to "wash away" semantic information. To address the issue, we propose using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation. Our proposed method outperforms the previous methods by a large margin. Furthermore, the new method enables natural extension to control the style of the synthesized images. Given a style guide image, our style encoder network captures the style into a latent code, which our image generator network combines with the semantic layout via spatially-adaptive normalization to generate a photo-realistic image that respects both the style of the guide image and content of the semantic layout. Our method will enable people without drawing skills to effectively express their imagination. In the inference time is a simple convolutional neural network. It runs real-time on most modern GPU cards. It is one of the recent research efforts in advancing GANs for real-time image rendering.

3.2PROPOSED SYSTEM

SAN(Spatially Adaptive Normalization):

- ❖ **A simple but effective layer for synthesizing photorealistic images given an input semantic layout.**
- ❖ Previous methods directly feed the semantic layout as input to the deep network, which is then processed through stacks of convolution, normalization, and nonlinearity layers due to which semantic information is "washed away".
- ❖ To address the issue, we propose using the input layout for modulating the activations in normalization layers through a spatially adaptive, learned transformation.

Application:

- Generating Realistic Images

SAN MODEL

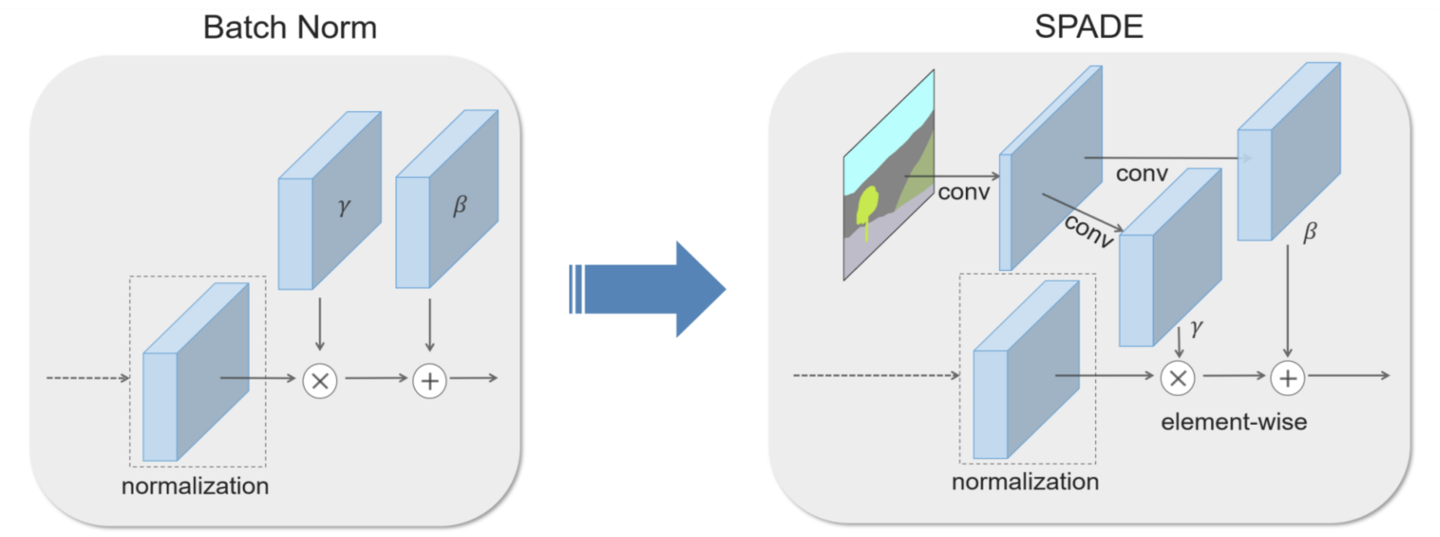


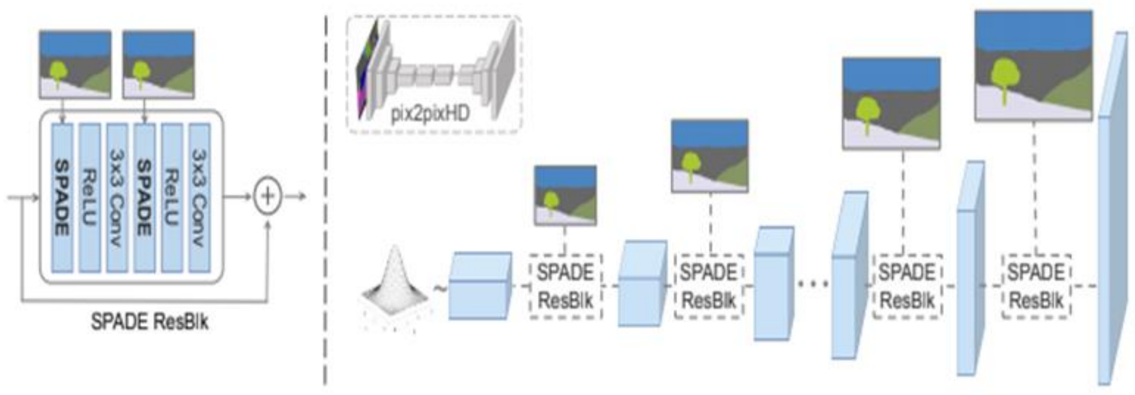
Figure:7

3.2.1METHODOLOGY:

SPADE.

- With SPADE, there is no need to feed the segmentation map to the first layer of the generator, since the learned modulation parameters have encoded enough information about the label layout.
- Therefore, we discard encoder part of the generator, which is commonly used in recent architectures.

SPADE outperforms existing methods on the COCO dataset which is more challenging than the Cityscapes dataset due to more diverse labels and scenes



In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations. *(left)* Structure of one residual block with SPADE. *(right)* The generator contains a series of SPADE residual blocks with upsampling layers. Our architecture achieves better performance with a smaller number of parameters by removing the downsampling layers of leading image-to-image translation networks (pix2pixHD [40]).

3.3 FEASIBILITY STUDY

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

The development of a computer-based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly.

Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

3.3.1 Types of Feasibility

- **Technical Feasibility**
- **Operational Feasibility**
- **Economical Feasibility**

3.3.1.1 Technical Feasibility

The application can be developed with the current equipments and has the technical capacity to hold the data required by the system.
This technology supports the modern trends of technology.
Easily accessible, more secure technologies.

3.3.1.1.1 Software and Platforms Used

NVIDIA GPU SYSTEM: The NVIDIAaccelerated computing platform gives modern data centers the power to accelerate deep learning, machine learning and high-performance computing (HPC) workloads.NVIDIA DGX™ solutions are made up of a powerful, fully integrated combination of innovative, GPU-optimized software, groundbreaking performance, and simplified management.
PYTHON IDE:IDE, is a very simple and sophisticated IDE developed primarily for beginners, and because of its simplicity, it is highly considered and recommended for educational purposes. It offers a variety of features
PYTORCH LIBRARY: PyTorch redesigns and implements Torch in Python while sharing the same core C libraries for the backend code. PyTorch developers tuned this back-end code to run Python efficiently. They also kept the GPU based hardware acceleration as well as the extensibility features that made Lua-based Torch.
CUDA:Compute **U**nified **D**evice **A**rchitecture. It is an extension of C programming, an API model for parallel computing created by Nvidia. Programs written using CUDA harness the power of GPU. Thus, increasing the computing performance.
TENSOR FLOW: TensorFlow provides a collection of workflows to develop and train models using Python, JavaScript, or Swift, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Technical feasibility on the existing system and to what extend it can support the proposed addition.
We can add new modules easily without affecting the Core Program. Most of parts are running in the server using the concept of stored procedures.

3.3.1.2 Operational Feasibility

This proposed system can easily be implemented, as this is based on Python coding and NVIDIA GPU system. The database created is with COCO datasets which is more secure and easy to handle. The resources that are required to implement/install are available. The personal of the organization already has enough exposure to computers. So, the project is operationally feasible and user friendly.

3.3.1.3 Economical Feasibility

If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. This system is more economically feasible and budget friendly which assess the brain capacity with quick & online test. So, it is economically a good project.

3.4 EFFORT AND COST ESTIMATION:

COCOMO (Constructive Cost Model) is a regression model based on LOC, i.e. number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best- documented models.

The key parameters which define the quality of any software products, which are also an outcome of the COCOMO are primarily Effort & Schedule:

- **Effort:** Amount of labour that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.
-

Different models of COCOMO have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.
3. **Embedded –** A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Software Requirements Specifications

INDEX

Software Requirements Specifications

1.Introduction

1.1 Purpose

In our proposed solution we focus on enhancing image quality to observe photorealistic images. This approach produces an image with photographic appearance that conforms to the input layout. Unlike recent and contemporaneous work, our approach does not rely on adversarial training. We show that photographic images can be synthesized from semantic layouts by a single feedforward network with appropriate structure, trained end-to-end with a direct regression objective.

1.2 Scope

In our proposed system, we propose **Spatially-adaptive normalization**, a simple but effective layer for synthesizing photorealistic images given an input semantic layout using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation.

2.Overall Description

2.1 Product Perspective

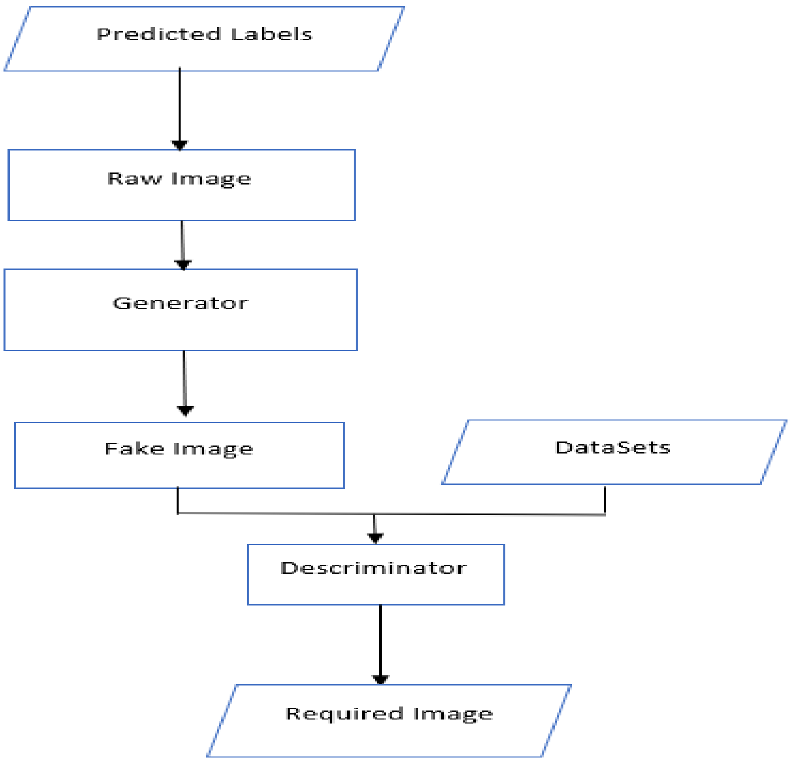


Fig 2.1 Architecture Description

2.2 Product Functions:

2.2.1 **Surveillance**

They play a significant role in applications such as traffic surveillance and security monitoring.

2.2.2 Medical diagnosis

Various medical imaging modalities can provide both anatomical information about the human body structure and functional information.

2.2.3 Earth-observation remote sensing

To improve the resolution of Landsat remote sensing images. satellites can ac- quire multi temporal or multi-view images

2.2.4 Astronomical observation

By improving the resolution of astronomical images, SR can help astronomers with the exploration of outer space.

2.2.5 Biometric information identification

Biometric recognition, including re- solution enhancement for faces fingerprints , and iris images . The resolution of biometric images is pivotal in the recognition and detection process

2.2.6 Education

By providing education with high resoluted images students can easily understand and show more interest in learning.

2.3 Operating Environment

NVIDIA GPU SYSTEM: The NVIDIAaccelerated computing platform gives modern data centers the power to accelerate deep learning, machine learning and high-performance computing (HPC) workloads.NVIDIA DGX™ solutions are made up of a powerful, fully integrated combination of innovative, GPU-optimized software, ground breaking performance, and simplified management.

PYTHON IDE:IDE, is a very simple and sophisticated IDE developed primarily for beginners, and because of its simplicity, it is highly considered and recommended for educational purposes. It offers a variety of features

PYTORCH LIBRARY: PyTorch redesigns and implements Torch in Python while sharing the same core C libraries for the backend code. PyTorch developers tuned this back-end code to run Python efficiently. They also kept the GPU based hardware acceleration as well as the extensibility features that made Lua-based Torch.

3.External Interface Requirement

3.1 User Interfaces

The input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation.

3.2 Software Interface

CUDA:Compute **U**nified **D**evice **A**rchitecture. It is an extension of C programming, an API model for parallel computing created by Nvidia. Programs written using CUDA harness the power of GPU. Thus, increasing the computing performance.

TENSOR FLOW: TensorFlow provides a collection of workflows to develop and train models using Python, JavaScript, or Swift, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.

NVIDIA GPU SYSTEM: The NVIDIAaccelerated computing platform gives modern data centers the power to accelerate deep learning, machine learning and high-performance computing (HPC) workloads.NVIDIA DGX™ solutions are made up of a powerful, fully integrated combination of innovative, GPU-optimized software, ground breaking performance, and simplified management.

WINDOWS(64 Bit)Operating System: In computer architecture, ***64-bit*** integers, memory addresses, or other data units are those that ... In 2003, ***64-bit*** CPUs were ***introduced*** to the (formerly 32-bit) mainstream personal computer market in the form of x86-64 processors and ... A ***64-bit*** register can hold any of 2^{64} (over 18 quintillion or $1.8\times\boldsymbol{10}^{19}$) different values.

4.SYSTEM FEATURES

4.1 Description and Priority

Pycharm IDE is too fast and provide too many functionalities like installing various packages for processing GPU tasks. CUDA is a version based for processing TensorFlow and Pytorch libraries of python. This version is combined together with NVIDIA GPU for segmenting high resolution images together.

4.2 Functional Requirements

REQ-1:The NVIDIAaccelerated computing platform gives modern data centers the power to accelerate deep learning, machine learning and high-performance computing (HPC) workloads.

REQ-2:CUDA:**C**ompute **U**nified **D**evice **A**rchitecture. It is an extension of C programming, an API model for parallel computing created by Nvidia. Programs written using CUDA harness the power of GPU. Thus, increasing the computing performance.

5.Other Non-functional Requirements

5.1 Performance Requirements

The software system to able to fulfill its purpose i.e, to produce the photorealistic images with given input semantic layout.It is able to produce high HD Resolution Images.

5.2 Software Quality Attributes

5.2.1 Usability

Our application can be viewed as a tool for producing a better quality images with less noise,High Hd resolution.

5.2.2 Efficiency

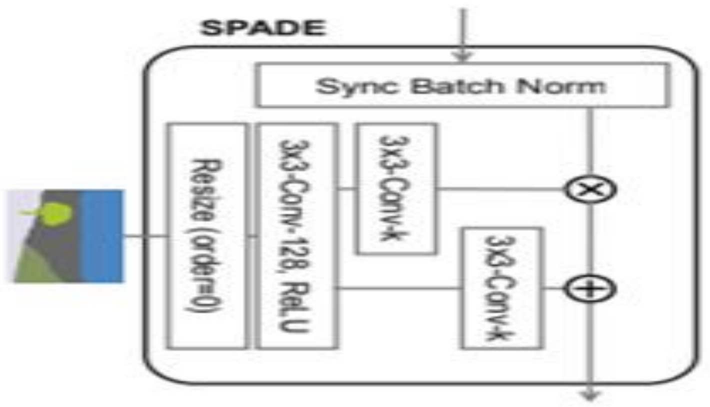
The software system to able to fulfill its purpose i.e, to produce the photorealistic images with given input semantic layout.It is able to produce high HD Resolution Images.

5.2.3 Robustness

Experiments on several challenging datasets demonstrate the advantage of the proposed method over existing approaches, regarding both visual fidelity and alignment with input layouts. Finally, our model allows user control over both semantic and style as synthesizing images.

4.SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

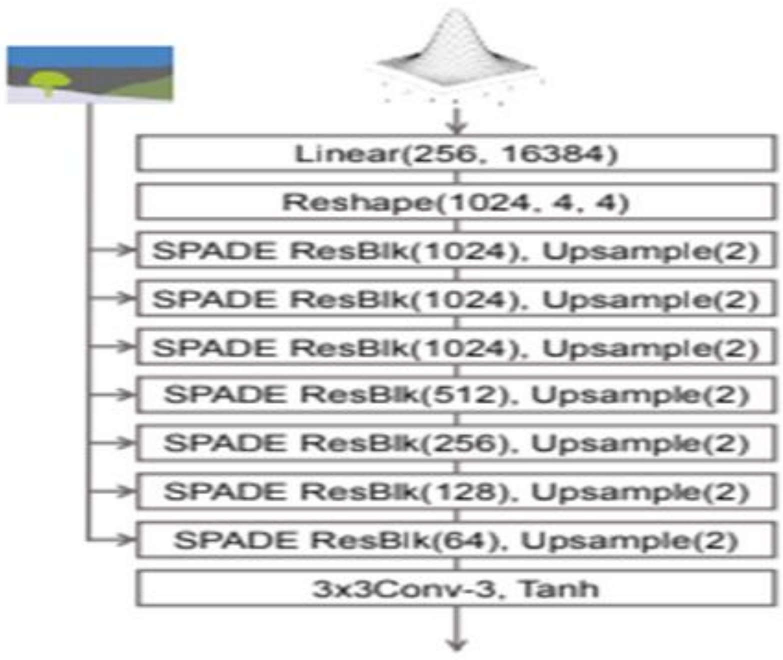


SPADE Design. The term 3x3-Conv-k denotes a 3-by-3 convolutional layer with k convolutional filters. The segmentation map is resized to match the resolution of the corresponding feature map using nearest-neighbor down-sampling.

Generator. The architecture of the generator consists of a series of the proposed SPADE ResBlks with nearest neigh bor upsampling. We train our network using 8 GPUs simultaneously and use the synchronized version of the batch normalization. We apply the spectral normalization [30] to all the convolutional layers in the generator.

Figure:10

Discriminator. The architecture of the discriminator follows the one used in the pix2pixHD method [40], which uses a multi-scale design with instance normalization (IN).The only difference is that we apply the spectral normalization to all the convolutional layers of the discriminator.


Linear(256, 16384)
Reshape(1024, 4, 4)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(512), Upsample(2)
SPADE ResBlk(256), Upsample(2)
SPADE ResBlk(128), Upsample(2)
SPADE ResBlk(64), Upsample(2)
3x3Conv-3, Tanh

SPADE Generator. Different from prior image generators [20, 40], the semantic segmentation mask is passed to the generator through the proposed SPADE ResBlks in Figure 11.

Image Encoder. The image encoder consists of 6 stride-2 convolutional layers followed by two linear layers to produce the mean and variance of the output distribution as shown in Figure 14.

Learning objective. We use the learning objective function in the pix2pixHD work [40] except that we replace its LS-GAN loss [28] term with the Hinge loss term [25, 30, 45].

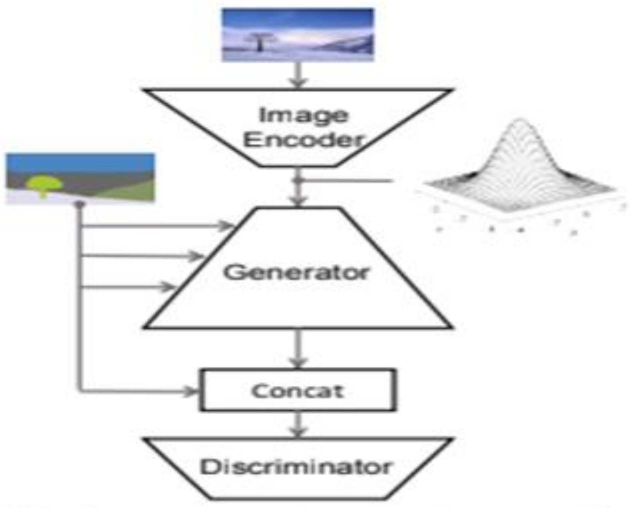
We use the same weighting among the loss terms in the objective function as that in the pix2pixHD work.When training the proposed framework with the image encoder for multi-modal synthesis and style-guided image synthesis, we include a KL

Divergence loss:

LKLD = DKL(q(z|x)||p(z))

where the prior distribution p(z) is a standard Gaussian distribution and the variational distribution q is fully determined by a mean vector and a variance vector.

Training details. We perform 200 epochs of training on the Cityscapes and ADE20K datasets, 100 epochs of training on the COCO-Stuff dataset, and 50 epochs of training on the Flickr Landscapes dataset. The batch size is 32. We initialize the network weights using Glorot i



: The image encoder encodes a real image to a latent representation for generating a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [22]. The generator also takes the segmentation mask of the input image as input via the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

5.1 IMPLEMENTATION

List Of Modules

- Semantic Palette
- Predictive Labels
- Tool-Bar

5.1.1 Semantic Palette

Semantic color palette is used to add extra functionality for the users design. This is used to give the enhancements to the predictive labels.

If a user wants to draw an imaginary segmentation object like clouds or mountains he can first select those labels then add the functionality from palette which gives the colour to the object i.e Blue colour to the clouds or sky and brown colour to mountains.

5.1.2 Predictive Labels

Predictive labels are given as input to the segmentation input layout which in-turn are processed by Generator of the model to produce an image which is sent to Image-Encoder part for segmenting the labels with image instances. This is finally compared to the original image by discriminator and result is produced. Thus predictive labels are used to produce images based on labels given in datasets. For example sky, mountains, water, sea are labels which incorporate with image instances.

5.1.3 Tool-Bar

Tool-Bar mainly consists of tools for sketching or designing users input as a predictive image. It consists of

Colour Bucket, Sketching Pen, Adding New Layout Button and Redraw .

This tools are user handy for any image to be produced.

5.2 Snippets

5.2.1 SPADE Block

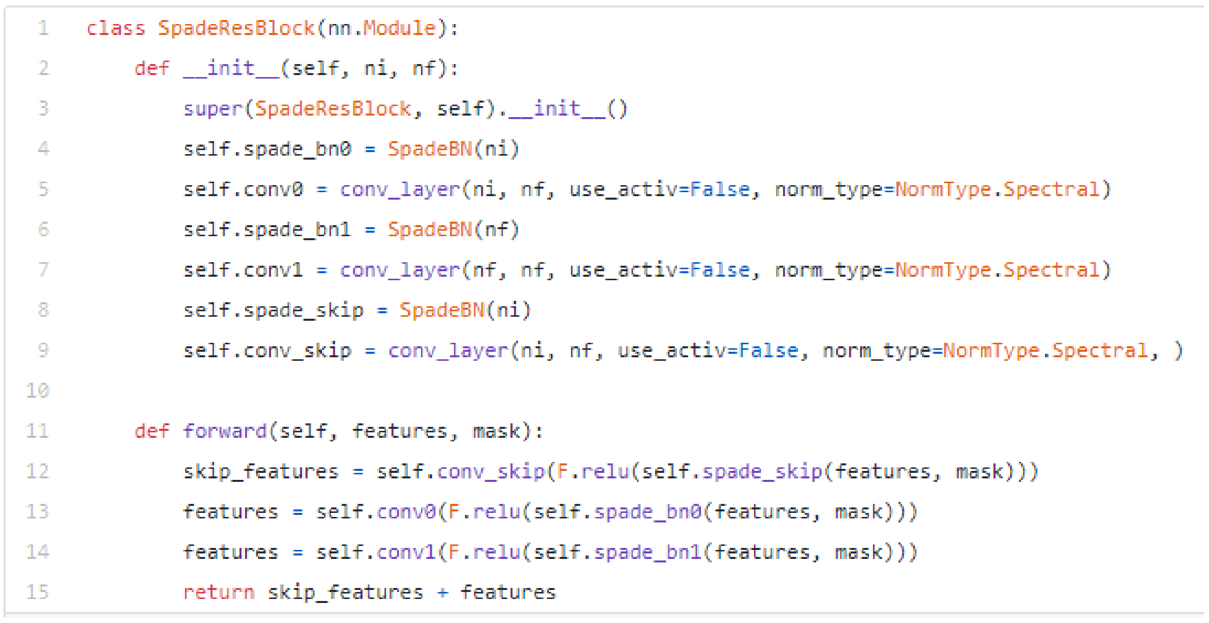
```python
# nn is imported from PyTorch
class SpadeBN(nn.Module):
    def __init__(self, nf):
        super(SpadeBN, self).__init__()

        self.bn = nn.BatchNorm2d(nf, affine=False)

        # conv_layer is a fastai function and NormType is a data type which tells
        # which type of Normalization layer to use.
        # Documentation at: https://docs.fast.ai/layers.html#conv_layer
        self.conv0 = conv_layer(1, 128, norm_type=NormType.Spectral)
        self.conv1 = conv_layer(128, nf, norm_type=NormType.Spectral)
        self.conv2 = conv_layer(128, nf, norm_type=NormType.Spectral)

    def forward(self, features, mask):
        size = features.size()[-2:]
        mask = F.interpolate(mask.float(), size=size)
        interim_conv = self.conv0(mask)
        gamma = self.conv1(interim_conv)
        beta = self.conv2(interim_conv)
        return (self.bn(features) * gamma) + beta
```

It accepts the segmentation mask and features. The segmentation mask is just the straight and simple long integer 2D mask, the paper advises to use embedding layer for the classes, but I decided to go simple, therefore the first convolutional layer number of input filters is 1. Then it resizes the mask in the size of the features. It does that because SPADE layer will be used at every layer, so it needs to know the size of the features so that the mask can be resized for the operation of the affine paramete. Take a look at when I initialize the BatchNorm2d layer I set affine to false to not use the default affine parameters.

Spectral Normalization is used in all the convolutional blocks in the paper to stabilize the GAN training. In PyTorch a new layer is implemented by inheriting from "*nn.Module*" and by implementing "*__init__*" and "*forward*" functions. The variable names *ni* and *nf* are used for number of input filters and number of output filter in convolutional layers respectively.

5.2.2 SPADE ResidualBlock

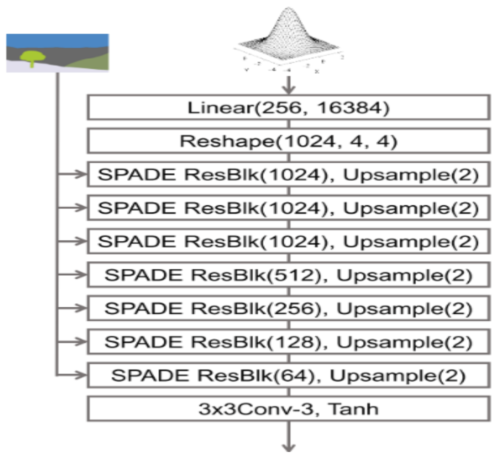
After implementing the SPADE block we can now use it in our SPADE ResBlk and its pretty straight forward.



```python
class SpadeResBlock(nn.Module):
    def __init__(self, ni, nf):
        super(SpadeResBlock, self).__init__()
        self.spade_bn0 = SpadeBN(ni)
        self.conv0 = conv_layer(ni, nf, use_activ=False, norm_type=NormType.Spectral)
        self.spade_bn1 = SpadeBN(nf)
        self.conv1 = conv_layer(nf, nf, use_activ=False, norm_type=NormType.Spectral)
        self.spade_skip = SpadeBN(ni)
        self.conv_skip = conv_layer(ni, nf, use_activ=False, norm_type=NormType.Spectral, )

    def forward(self, features, mask):
        skip_features = self.conv_skip(F.relu(self.spade_skip(features, mask)))
        features = self.conv0(F.relu(self.spade_bn0(features, mask)))
        features = self.conv1(F.relu(self.spade_bn1(features, mask)))
        return skip_features + features
```

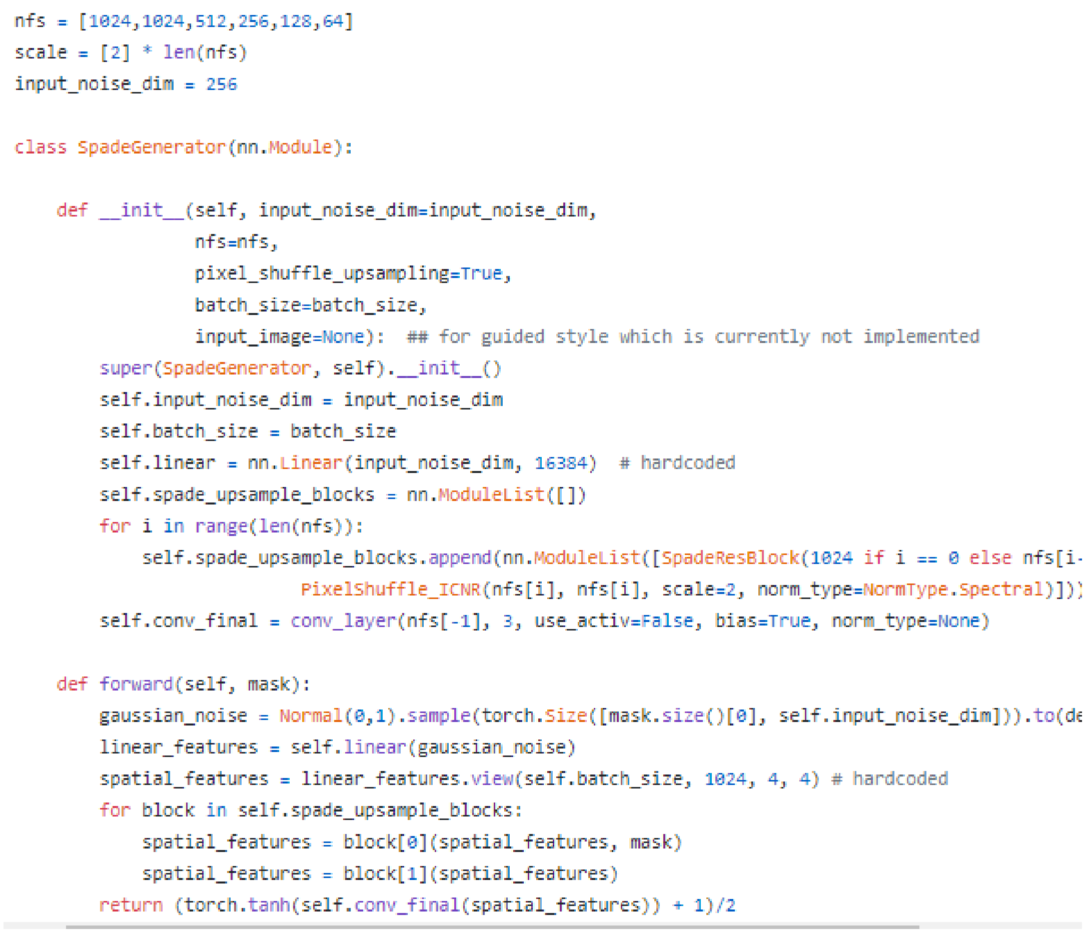
5.2.3 Generator

Now, we have got the basic block setup and now is the time to stack them up as shown in the architecture below.



Linear(256, 16384)
Reshape(1024, 4, 4)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(1024), Upsample(2)
SPADE ResBlk(512), Upsample(2)
SPADE ResBlk(256), Upsample(2)
SPADE ResBlk(128), Upsample(2)
SPADE ResBlk(64), Upsample(2)
3x3Conv-3, Tanh

I listed out the number of feature maps that individual layers will be throwing out and created the generator using a for loop.

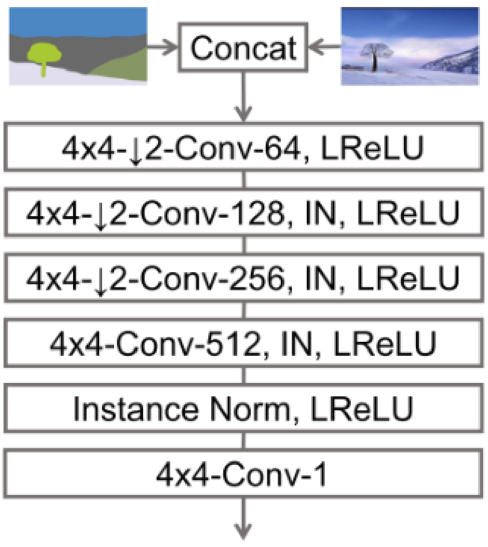


```
nfs = [1024,1024,512,256,128,64]
scale = [2] * len(nfs)
input_noise_dim = 256

class SpadeGenerator(nn.Module):

    def __init__(self, input_noise_dim=input_noise_dim,
                 nfs=nfs,
                 pixel_shuffle_upsampling=True,
                 batch_size=batch_size,
                 input_image=None):  ## for guided style which is currently not implemented
        super(SpadeGenerator, self).__init__()
        self.input_noise_dim = input_noise_dim
        self.batch_size = batch_size
        self.linear = nn.Linear(input_noise_dim, 16384)  # hardcoded
        self.spade_upsample_blocks = nn.ModuleList([])
        for i in range(len(nfs)):
            self.spade_upsample_blocks.append(nn.ModuleList([SpadeResBlock(1024 if i == 0 else nfs[i-
                              PixelShuffle_ICNR(nfs[i], nfs[i], scale=2, norm_type=NormType.Spectral)]))
        self.conv_final = conv_layer(nfs[-1], 3, use_activ=False, bias=True, norm_type=None)

    def forward(self, mask):
        gaussian_noise = Normal(0,1).sample(torch.Size([mask.size()[0], self.input_noise_dim])).to(de
        linear_features = self.linear(gaussian_noise)
        spatial_features = linear_features.view(self.batch_size, 1024, 4, 4) # hardcoded
        for block in self.spade_upsample_blocks:
            spatial_features = block[0](spatial_features, mask)
            spatial_features = block[1](spatial_features)
        return (torch.tanh(self.conv_final(spatial_features)) + 1)/2
```

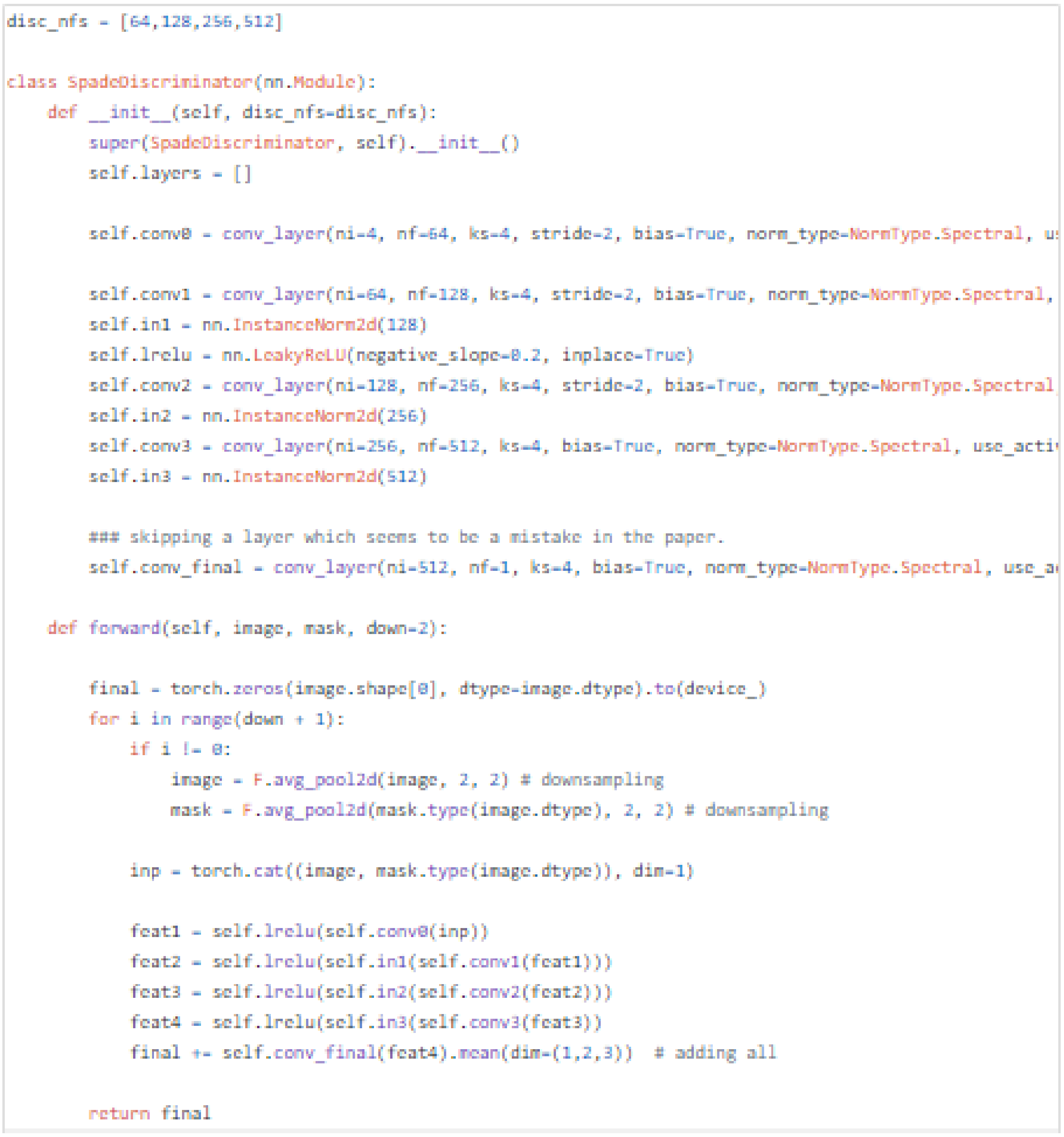
To keep things simple in the code, I use these tricks like initializing the parameters of the module with a global variable. You can see that the *nfs* variable contains the output of all the SPADE residual blocks which used in initializing the layers in the generator. In the end, I make the output from the *tanh* layer to be in the range 0–1, so that it is simpler to visualize.

5.2.4 Discriminator

The discriminator is a Multi-Scale, Patch Gan based Discriminator. Multi-Scale means that it classifies the given image at different scale. Patch Gan based discriminator the final output layer is convolutional and the spatial mean is taken. The output from multiple scale is just added to give the final output.

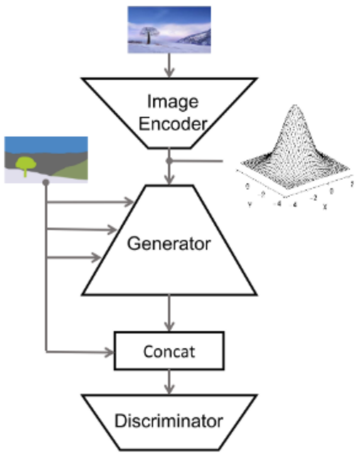


The discriminator takes both mask and the generated/real images at one time and output the activation. As this can be seen in the code the forward method in the discriminator is taking both mask and image as input.



```
disc_nfs = [64,128,256,512]

class SpadeDiscriminator(nn.Module):
    def __init__(self, disc_nfs=disc_nfs):
        super(SpadeDiscriminator, self).__init__()
        self.layers = []

        self.conv0 = conv_layer(ni=4, nf=64, ks=4, stride=2, bias=True, norm_type=NormType.Spectral, us

        self.conv1 = conv_layer(ni=64, nf=128, ks=4, stride=2, bias=True, norm_type=NormType.Spectral,
        self.in1 = nn.InstanceNorm2d(128)
        self.lrelu = nn.LeakyReLU(negative_slope=0.2, inplace=True)
        self.conv2 = conv_layer(ni=128, nf=256, ks=4, stride=2, bias=True, norm_type=NormType.Spectral
        self.in2 = nn.InstanceNorm2d(256)
        self.conv3 = conv_layer(ni=256, nf=512, ks=4, bias=True, norm_type=NormType.Spectral, use_acti
        self.in3 = nn.InstanceNorm2d(512)

        ### skipping a layer which seems to be a mistake in the paper.
        self.conv_final = conv_layer(ni=512, nf=1, ks=4, bias=True, norm_type=NormType.Spectral, use_a

    def forward(self, image, mask, down=2):

        final = torch.zeros(image.shape[0], dtype=image.dtype).to(device_)
        for i in range(down + 1):
            if i != 0:
                image = F.avg_pool2d(image, 2, 2) # downsampling
                mask = F.avg_pool2d(mask.type(image.dtype), 2, 2) # downsampling

            inp = torch.cat((image, mask.type(image.dtype)), dim=1)

            feat1 = self.lrelu(self.conv0(inp))
            feat2 = self.lrelu(self.in1(self.conv1(feat1)))
            feat3 = self.lrelu(self.in2(self.conv2(feat2)))
            feat4 = self.lrelu(self.in3(self.conv3(feat3))
            final += self.conv_final(feat4).mean(dim=(1,2,3))  # adding all

        return final
```

Now we have implemented the complete architecture of the model. The complete



architecture looks like this.

6.TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and /or a finished product.

Types of Testing

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests endure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Test Cases:

A test case is a set of conditions or variables under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Test Case: Input and results into an output image.

Test Objective: To check whether the given input is processed and results into an high resolute image based on labels given.

Test Description: The user confirms whether the input provided is valid and is segmented correctly.

Test Results: All the test cases mentioned above passed successfully. No defects encountered leading to the successful rendering of image as desired by user perspective.

Image Result:

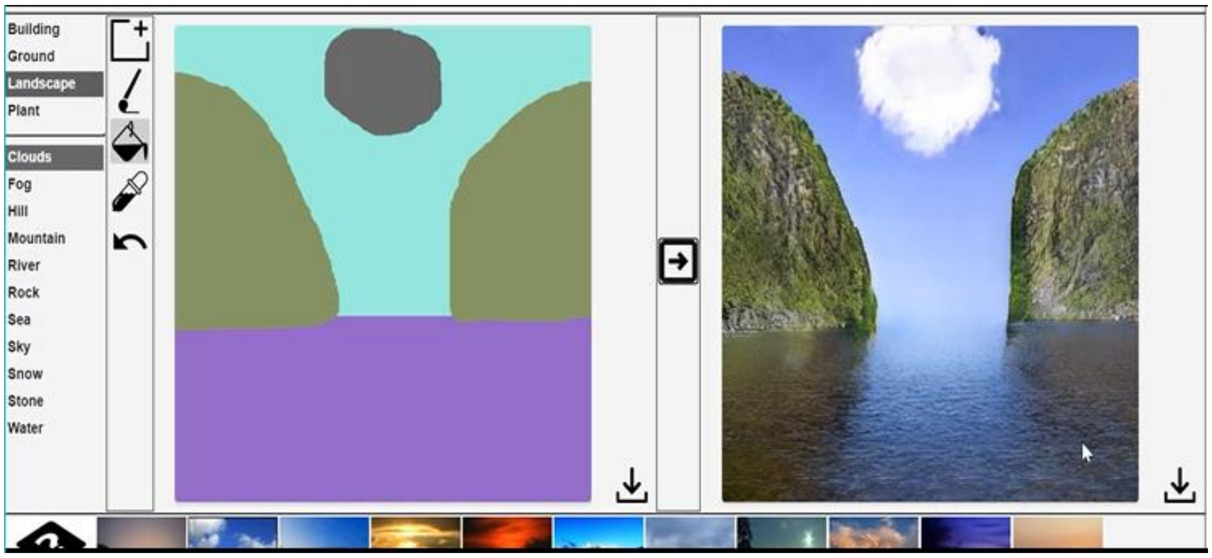


Fig-6

7.SCREENSHOTS

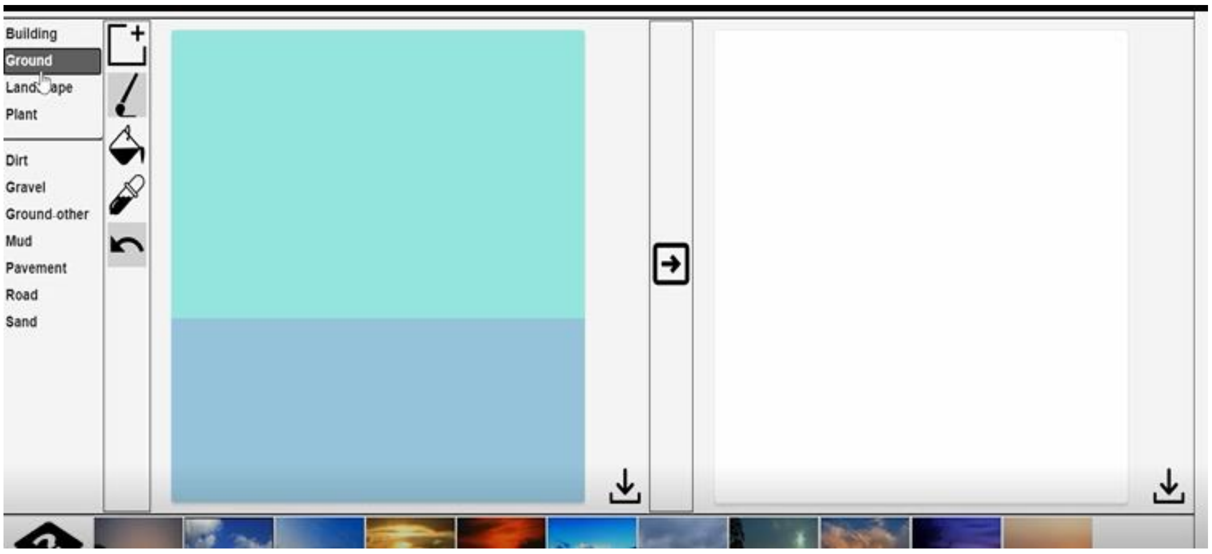


Fig-7.1 GUI Of Proposed Model

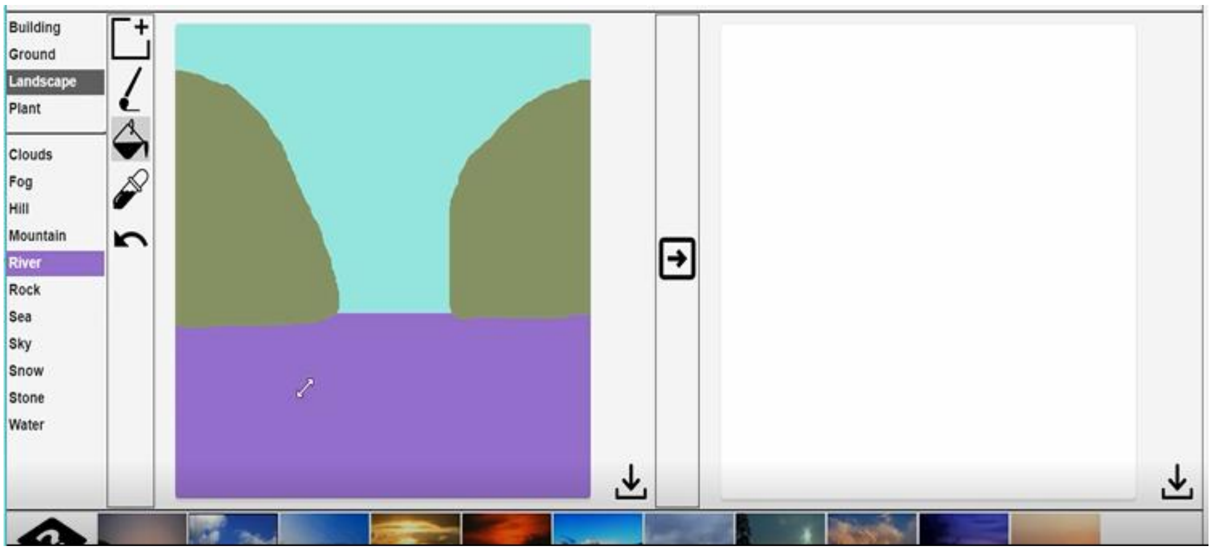


Fig-7.2 User Input

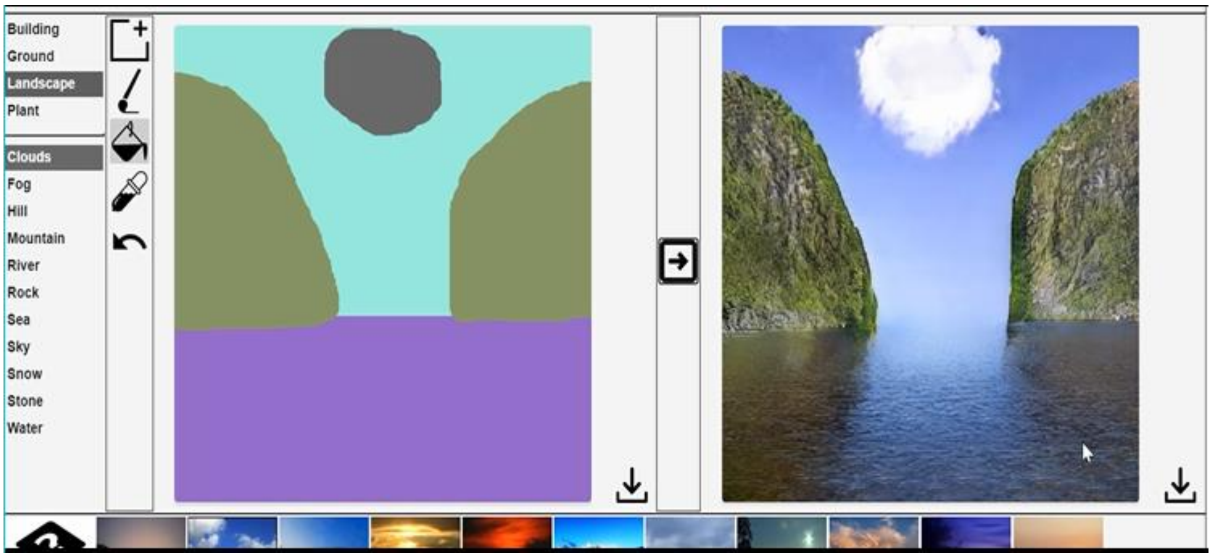


Fig-7.3 Processed Output

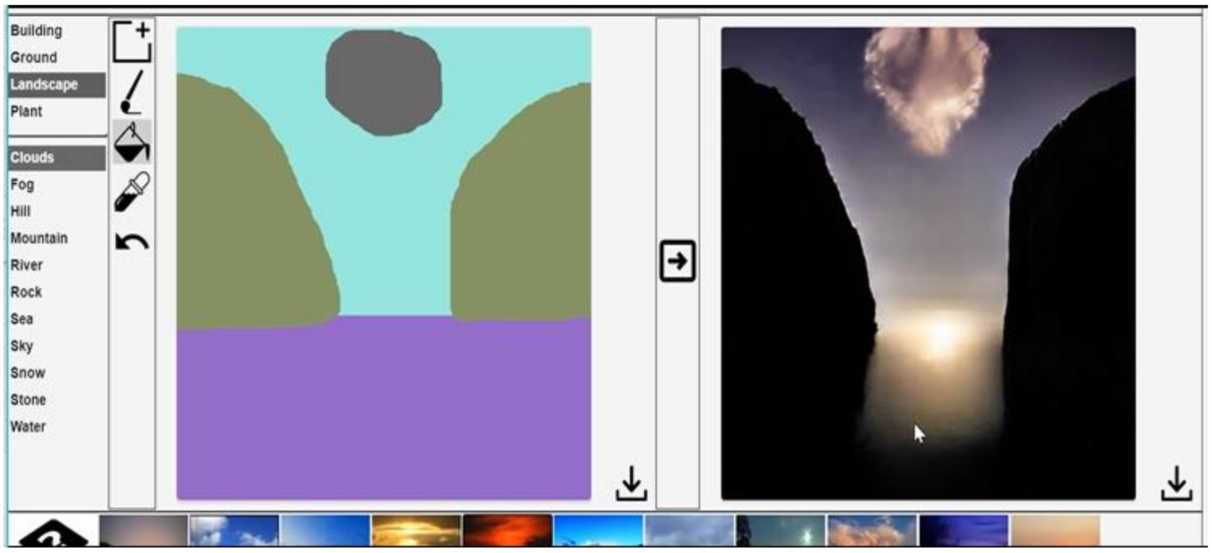


Fig-7.4 Image With Enhancement Effect

8.CONCLUSION

Producing Images with High resolution has been one of the difficult problem in present world. This problem has made an evolution to produce a model which can overcome such dilemma known as “Spatially Adaptive Normalization” In which important factors influencing the performance metrics have been analyzed. This parameters states that for an image to be accurate and resoluted (mIou and Pixel accuracy needs to be high) and FID to be lower. In the proposed application solution, the image have been rendered successfully based on users input. Thus the model we have proposed is free from errors and up to the desired results.

9.REFERENCES

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In International Conference on Machine Learning (ICML), 2017

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalizationarXiv preprint arXiv:1607.06450, 2016.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In ACM SIGGRAPH, 2009.

[4] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In International Conference on Learning Representations (ICLR) 2019. 1, 2

[5] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

APPENDIX I

RELEVANCE OF PROJECT TO POs and PSOs

PROJECT TITLE	CATEGORY
Semantic Image Synthesis Through SPADE	Application

ABSTRACT

Image synthesis is the process of creating new images from some form of image description.Synthetic images are often used to verify the correctness of operators by applying them to known images.The operator output on such images is generally clean used in Education, CAD ... Previous methods (CRN, pix2pix) directly feed the semantic layout as input to the network, which is then processed through stacks of convolution, normalization, and nonlinearity layers. We show that this is suboptimal because the normalization layers tend to wash away semantic information. To address this issue,we propose Spatially-adaptive normalization, a simple but effective layer for synthesizing photorealistic images given an input semantic layout using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation.

This approach produces an image with photographic appearance that conforms to the input layout. Unlike recent and contemporaneous work, our approach does not rely on adversarial training. We show that photographic images can be synthesized from semantic layouts by a single feedforward network with appropriate structure, trained end-to-end with a direct regression objective.

Both visual fidelity and alignment with input layouts were improved over several datasets when the above approach was used in contrast to existing methodology. Finally, our model allowsusers to easily control the style and content of synthesis results as well as create multi-modalresults. Extensive perceptual experiments on datasets of outdoor and indoor scenes demonstrate that images synthesized by the presented approach are considerably more realistic than alternative approaches.

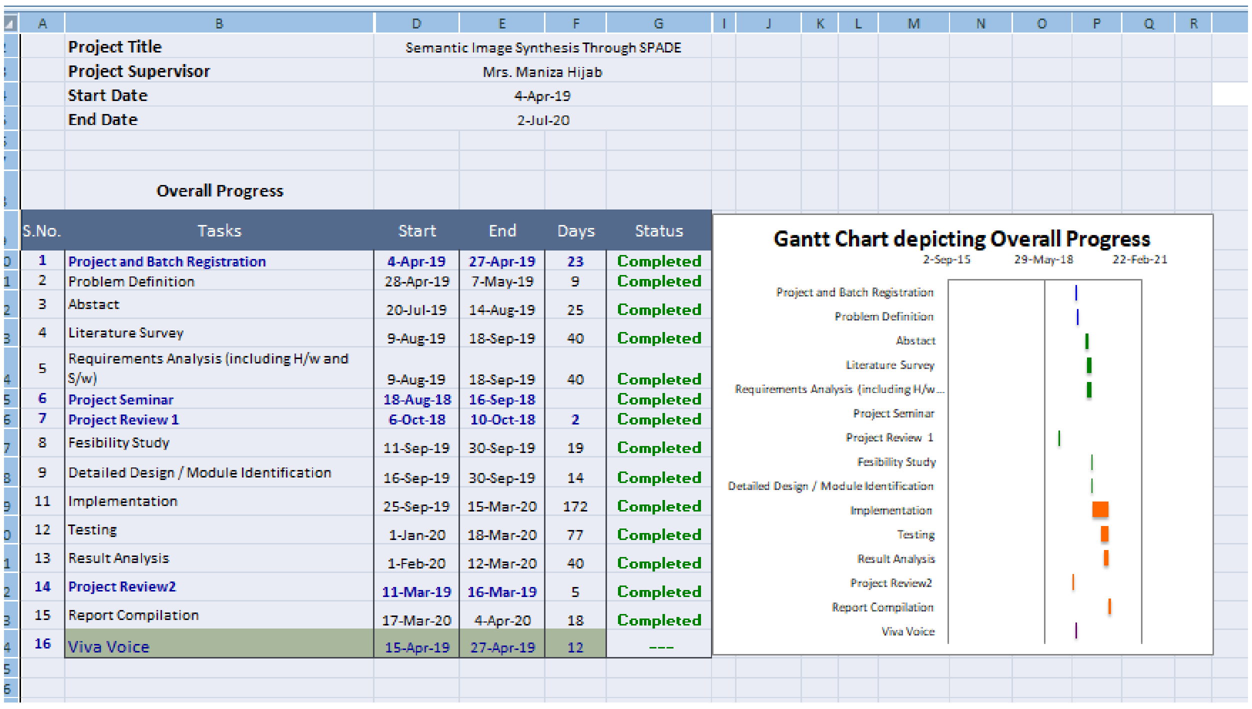
PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
3	**3**	**3**	**2**	**2**	**3**	**1**	**2**	**3**	**2**	**3**	**2**

Table [put table no] Mapping to PSOs: (Mapping Scale [1-3]: 1-Slight 2-Moderate 3-Strong)

PSO1	PSO2	PSO3
3	**2**	**3**

RELEVANCE DETAILS	
Implementation details The project is implemented using PYTHON Language, NVIDIA GPU, CUDA, PyCharm IDE	
PO and PSO Justification	
PO1	It is moderately mapped as the datasets have been created using the basic concepts of Operating System.
PO2	It is strongly mapped as the Segmentation of image instances is based on the review literature.
PO3	It is strongly mapped as images rendered are processed using various Convolution layers.
PO4	It is moderately mapped as various built in functions were used which were imported from Pytorch library.
PO5	It is strongly mapped as the PyCharm IDE and NVIDIA has been used to develop the application.
PO6	It is strongly mapped as contextual knowledge of the SPADE is difficult to understand.
PO7	It is weakly mapped as it serves the general purpose of Paint Application.
PO8	It is strongly mapped because the plagiarism tool can be used to check the authenticity and originality of the work done by the students
PO9	Projects are used to inculcate group work and to manage a team for promoting knowledge, conceptualization and delivering same with varied complexity, therefore the mapping is strong.
PO10	Demonstrate versatile and effective communication skills, both verbal and written with team members and present the product to the audience in comprehensive manner. Therefore, the mapping is moderate.
PO11	High Level knowledge and understanding of engineering principles along with ML algorithms are applied in the development of this project hence a strong mapping.
PO12	The mapping is moderate as projects are executed based on the self-learning or self-efforts put in by the group.
PSO1	It is strongly mapped as understanding of the principles and working of the hardware and software aspects of computer systems and the Machine Learning environment is required to decompose the system into phases and workflows.
PSO2	It is strongly mapped as the application the tools used for development, operation and maintenance are high.

Work Process Plan



Project Title	Semantic Image Synthesis Through SPADE
Project Supervisor	Mrs. Maniza Hijab
Start Date	4-Apr-19
End Date	2-Jul-20

Overall Progress

S.No.	Tasks	Start	End	Days	Status
1	Project and Batch Registration	4-Apr-19	27-Apr-19	23	Completed
2	Problem Definition	28-Apr-19	7-May-19	9	Completed
3	Abstract	20-Jul-19	14-Aug-19	25	Completed
4	Literature Survey	9-Aug-19	18-Sep-19	40	Completed
5	Requirements Analysis (including H/w and S/w)	9-Aug-19	18-Sep-19	40	Completed
6	Project Seminar	18-Aug-18	16-Sep-18		Completed
7	Project Review 1	6-Oct-18	10-Oct-18	2	Completed
8	Fesibility Study	11-Sep-19	30-Sep-19	19	Completed
9	Detailed Design / Module Identification	16-Sep-19	30-Sep-19	14	Completed
11	Implementation	25-Sep-19	15-Mar-20	172	Completed
12	Testing	1-Jan-20	18-Mar-20	77	Completed
13	Result Analysis	1-Feb-20	12-Mar-20	40	Completed
14	Project Review2	11-Mar-19	16-Mar-19	5	Completed
15	Report Compilation	17-Mar-20	4-Apr-20	18	Completed
16	Viva Voice	15-Apr-19	27-Apr-19	12	---