Journal of Innovation

March 2024 | 24th Edition

Object Management Group

Advancements in Synthetic Video Generation for Autonomous Driving

Generative Adversarial Neural Networks, Deep Learning, and Image Processing

Authors:

Manoj CR
Tata Consultancy Services
manoj.cr@tcs.com

Arunkrishna Thayyilravi
Tata Consultancy Services
arunkrishna.thayyilravi@tcs.com

Rajat Soni
Tata Consultancy Services
soni.rajat@tcs.com

Gokul Nair
Tata Consultancy Services
gokulg.nair@tcs.com

CONTENTS

1 Overview 4
2 Motivation 5
2.1 Survey 5
3 Proposals and Contributions 13
4 Methodology 14
4.1 Generative Adversarial Networks 14
4.2 Training of Generative Adversarial Networks 14
4.3 Objective Functions in Commonly used GANs 14
4.4 Generator Architecture 15
4.5 Discriminator Architecture 17
4.6 Objective Functions Used in Proposed GAN 18
4.7 Training and Validation 19
5 Results and Conclusions 20
5.1 Sample Frame Output 20
5.2 KPI (Key Performance Indicators) Evaluation 21
5.3 Conclusion 23
References 23
Acknowledgements 25

FIGURES

Figure 2-1: Different environmental conditions clockwise from top-left daylight, rain scene, sunlight, evening. [15] 6
Figure 2-2: Depth and segmentation output from CARLA simulator. [15] 6
Figure 2-3: Architecture for low resolution outputs. [5] 7
Figure 2-4: Architecture for high resolution outputs G1 corresponds to lower architecture network and G2 corresponds to higher resolution network. [5] 7
Figure 2-5: Architecture of label/flow embedding, image and segmentation used in world-consistent video-to-video synthesis. [4] 8
Figure 2-6: Architecture of generator used in world-consistent video-to-video synthesis. [4] 9
Figure 2-7: Architecture of multi SPADE residual block used in world-consistent video-to-video synthesis. [4] 9
Figure 2-8: Instance as well as label used for encoding. [12] 10
Figure 2-9: SPADE normalization. [13] 11

Figure 2-10: SEAN normalization. [3]......12
Figure 4-1: Multi SEAN-based image generator. 15
Figure 4-2: Multi SEAN residual block. 16
Figure 4-3: Architecture of label/flow embedding, image and segmentation. 17
Figure 4-4: Architecture of PatchGAN. 17
Figure 4-5: Overall system architecture. 19
Figure 5-1: Scenario generation outputs. 21

TABLES

Table 5-1: KPI Evaluation A. 22
Table 5-2: KPI Evaluation B. 22

1 Overview

Advanced driver assistance systems (ADAS) and their application in autonomous vehicles are evolving day by day. One of the main factors in focus is the availability of real-world or realistic datasets for training the deep-learning autonomous driving models. The quality, quantity and diversity of these datasets are important while training, testing, and validating deep learning models. Training any model corresponding to different conditions is a big challenge, as it is difficult to have a dataset that includes multiple geographies, weather conditions, surroundings, and road conditions.

According to the RAND Corporation report[1], autonomous vehicles must be driven billions of miles for reliability in terms of safety and security, so it requires almost 10 years to collect all the necessary datasets. This manual driving approach to collect datasets would also require a workforce and proper sensor configurations, which will eventually be expensive. Simulators like CARLA[2] can reduce the reality gap and generate huge trainable datasets, but the rendered scenarios appear unrealistic compared to real-world scenarios. Deep neural networks can help us generate realistic data specific to the ADAS applications under consideration.

Our proposed system is based on Generative Adversarial Networks (GANs), which use Semantic Region-Adaptive Normalization (SEAN)[3] based Image Generator. The article explores the possibilities of generating semantic inputs for GANs. It also explores pre-processing of sematic inputs and post-processing for generator outputs using Image processing. We propose an evaluation metric that can reduce human intervention. It also employs ideas from existing video synthesis solutions[4],[5].

The Video-to-video synthesis[6] method takes semantic labels as input and produces higher-quality photorealistic videos by considering previous frames and flow maps. It uses a cascaded approach to generate higher-quality outputs. Common issues of long-term temporal coherence has been addressed in our solution taking cues from world-consistent video-to-video synthesis[4]. This approach solves long-term temporal coherence problems by taking Spatially Adaptive Normalization (SPADE). Adding depth[7] and guidance image along with semantic[8] and flow inputs further refines output.

[1] *https://www.rand.org/pubs/research_reports/RR1478.html*
[2] *https://scholar.uwindsor.ca/etd/8305/*
[3] *https://arxiv.org/abs/1911.12861*
[4] *https://arxiv.org/abs/2007.08509*
[5] *https://arxiv.org/abs/1808.06601*
[6] *https://arxiv.org/abs/1910.12713*
[7] *https://arxiv.org/abs/2007.08854*
[8] *https://arxiv.org/abs/1911.10194*

Based on approaches from previous works, their advantages and limitations, we propose a novel approach which suggests fusion of Generative Adversarial Neural Network, Deep Learning, and Image processing. The proposed system aims to build an environment that will provide a one-stop solution for different AI (Artificial Intelligence) rendering, scenario generation and future video prediction. We also propose various possibilities[9] through which an end-user can generate realistic data from our system. For the testing of generated outputs, we have taken evaluation scores based on Fréchet Inception Distance (FID)[10] and Kanade-Lucas-Tomasi (KLT)[11] scores to reduce human intervention.

The organization of this report is as follows: Chapter 2 includes a survey, Chapter 3 includes proposals and contributions, Chapter 4 includes methodology, and Chapter 5 includes results and conclusions.

2 MOTIVATION

2.1 SURVEY

Research was conducted previously for image-to-image translation[12],[13],[14]. Video-to-video synthesis aspects have also been explored in the recent past [4]–[6], but have not been observed to be highly effective. Our work aims to explore the different possibilities from an end-user perspective of the application. It also aims to improve the performance of the latest work done in this area to develop a system that can help in the faster development of advanced driver assistance systems (ADAS).

K. K. Patel [2] explores the possibilities of reducing the reality gap in autonomous vehicles development, for which simulator-based environments are used to generate multiple scenarios of different environments. CARLA[15], an open-source simulation environment for autonomous driving research, aims to support the development, training, and validation of various autonomous driving modes.

[9] *https://arxiv.org/abs/1609.01326*
[10] *https://arxiv.org/abs/1706.08500*
[11] *https://ieeexplore.ieee.org/document/323794*
[12] *https://arxiv.org/abs/1711.11585*
[13] *https://arxiv.org/abs/1903.07291*
[14] *https://arxiv.org/abs/1611.07004*
[15] *https://arxiv.org/abs/1711.03938*

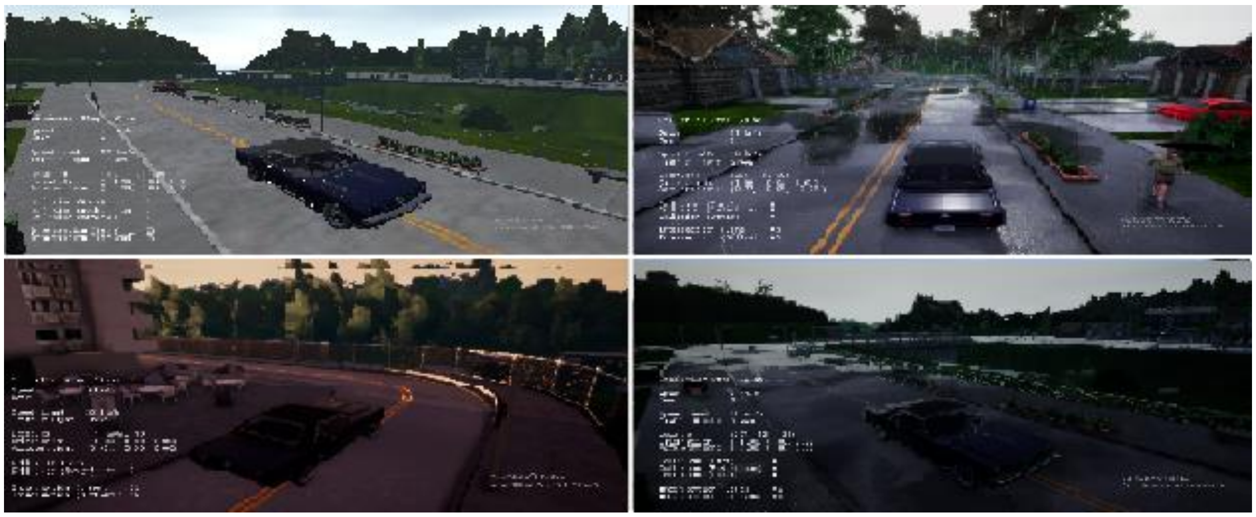


Figure 2-1: Different environmental conditions clockwise from top-left daylight, rain scene, sunlight, evening. [15]

CARLA provides different layouts and modes related to buildings, vehicles, and urban layouts, which is quite useful as it is an open-source platform without any cost. The simulation environment is user-friendly, and documentation integrating various sensors, environmental conditions, static and dynamic parameters control, scenario, and map generation are readily available at CARLA's official website and blogs [15].

Figure 2-1 shows scenes corresponding to environmental conditions e.g., daylight, rainy seasons, just after rain (sunny) and evening scenes. The limitations of generating datasets with this approach are that it tends to generate a not very realistic game-like environment. However, it does help in reducing the reality gap to an extent. One of the features we will use from CARLA is to generate inputs corresponding to our proposed system.

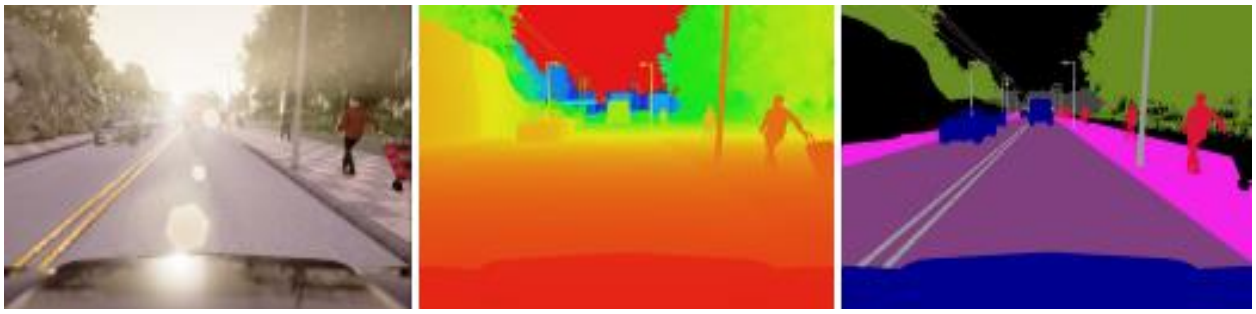


Figure 2-2: Depth and segmentation output from CARLA simulator. [15]

Figure 2-2 describes the different sensors (RGB (Red, Green, Blue), Depth) and their outputs corresponding to a scenario, which will help us create a user-centric approach to generate inputs to our proposed model.

Video-to-video synthesis proposed by T. C. Wang et al. [5] discusses an approach using Generative Adversarial Networks (GANs)[16]. The study employs sequential generator architecture to achieve spatial-temporal coherence. It converts input-segmented video to realistic video. For lower-resolution images, inputs are previously generated output frames and semantic images. The label maps are combined to form intermediate high-level features that are processed from various residual blocks, as shown in Figure 2-3. It applies similar residual-based processing for the previous images. Then, after combining intermediate layers, it is again processed by two different residual networks, which gives an output as the intermediate image, the mask, and the flow map.

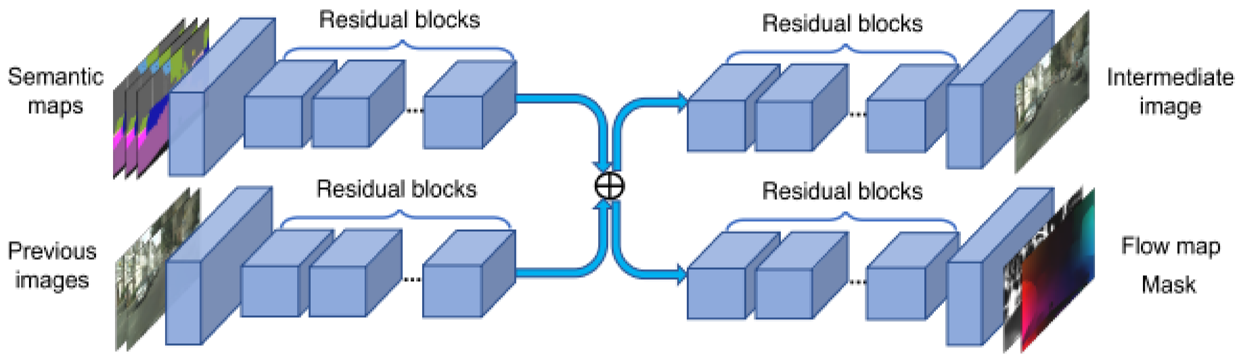


Figure 2-3: Architecture for low resolution outputs. [5]

As explained earlier, it uses similar configurations above the low-resolution network for higher-resolution results. It first down-samples the inputs and feeds them into the low-resolution network.

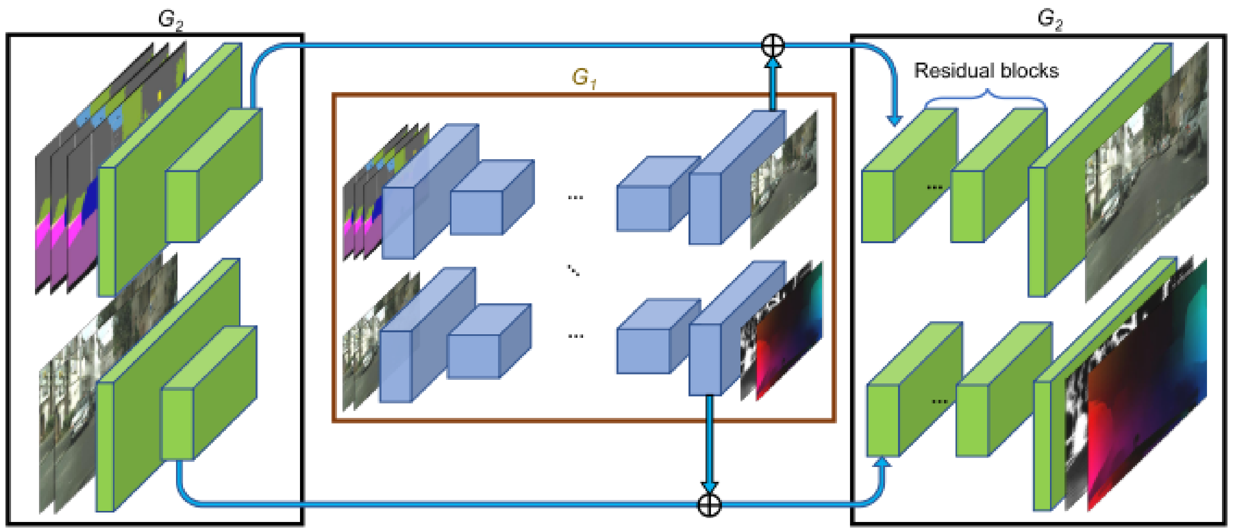


Figure 2-4: Architecture for high resolution outputs G1 corresponds to lower architecture network and G2 corresponds to higher resolution network. [5]

[16] *https://arxiv.org/abs/1406.2661*

Then, it integrates with the intermediate feature layer and extracts features from the last feature layer. These integrated features are then processed with various residual blocks to provide higher-resolution images. This approach was the first of its kind of detailed approach to video-to-video translation problems. The limitation of this approach is that it fails to solve issues like long-term temporal consistency, different AI rendering, and user perspective solutions.

World-Consistent Video-to-Video Synthesis proposed by A. Mallya, T. C. Wang et al. [4] discuss a new video synthesis framework that uses all previously generated frames. This is done by projecting structure from motion of all the previous frames to the existing frame. It uses a generator that corresponds to SPADE architecture and a discriminator that is like that of video-to-video synthesis [5].

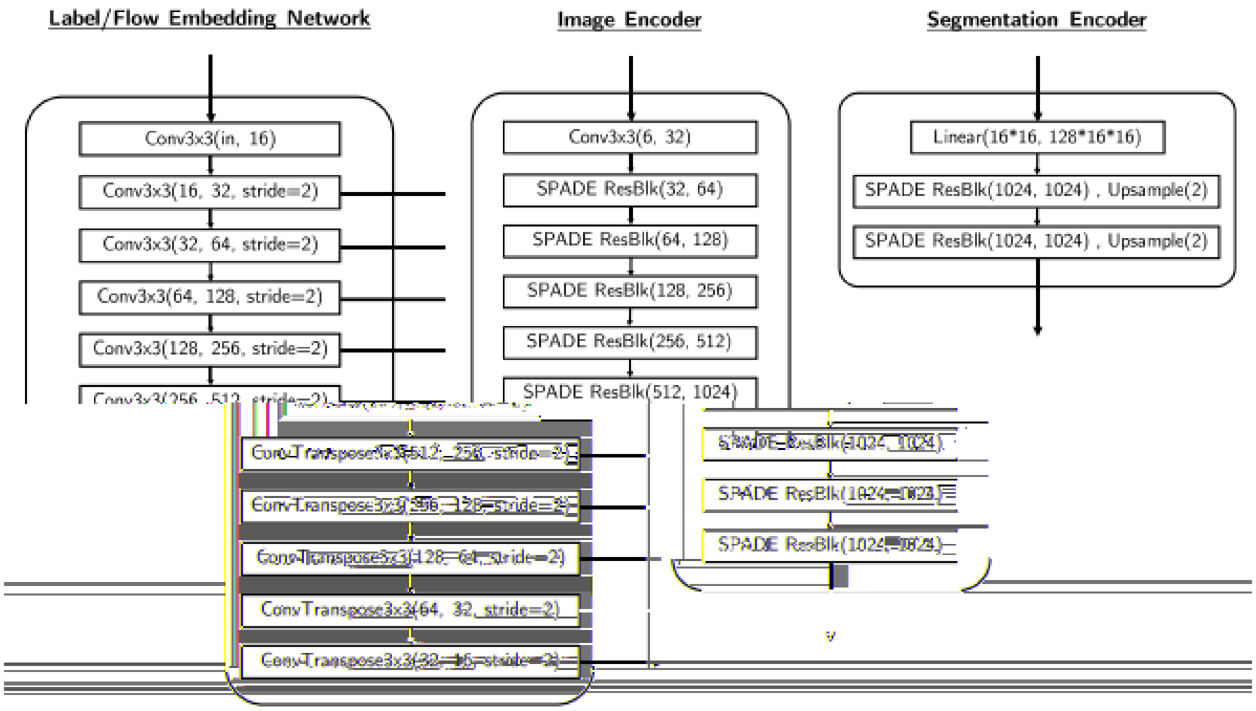

Label/Flow Embedding Network
Conv3x3(in, 16)
Conv3x3(16, 32, stride=2)
Conv3x3(32, 64, stride=2)
Conv3x3(64, 128, stride=2)
Conv3x3(128, 256, stride=2)
Conv3x3(256, 512, stride=2)
ConvTranspose3x3(512, 256, stride=2)
ConvTranspose3x3(256, 128, stride=2)
ConvTranspose3x3(128, 64, stride=2)
ConvTranspose3x3(64, 32, stride=2)
ConvTranspose3x3(32, 16, stride=2)
Image Encoder
Conv3x3(6, 32)
SPADE ResBlk(32, 64)
SPADE ResBlk(64, 128)
SPADE ResBlk(128, 256)
SPADE ResBlk(256, 512)
SPADE ResBlk(512, 1024)
SPADE ResBlk(1024, 1024)
SPADE ResBlk(1024, 1024)
SPADE ResBlk(1024, 1024)
Segmentation Encoder
Linear(16*16, 128*16*16)
SPADE ResBlk(1024, 1024), Upsample(2)
SPADE ResBlk(1024, 1024), Upsample(2)

Figure 2-5: Architecture of label/flow embedding, image and segmentation used in world-consistent video-to-video synthesis. [4]

The overall network architecture consists of Label Embedding, Flow Embedding, an Image Encoder, and an Image Generator, as given in Figure 2-5 and Figure 2-6, respectively. Label encoding uses an encoder-decoder style to embed input labels to distinctive features, which acts as one of the inputs to Multi SPADE in the Image generator. Flow Embedding deals with optical flow outputs of the previous frame, that is fed through the Multi SPADE block of Image generators. Image and Segmentation generators encode Image and segmented frames, respectively. The image encoder uses previously generated frames, while the segmentation encoder uses first-frame semantics. This helps to generate inputs for the Image Generator.

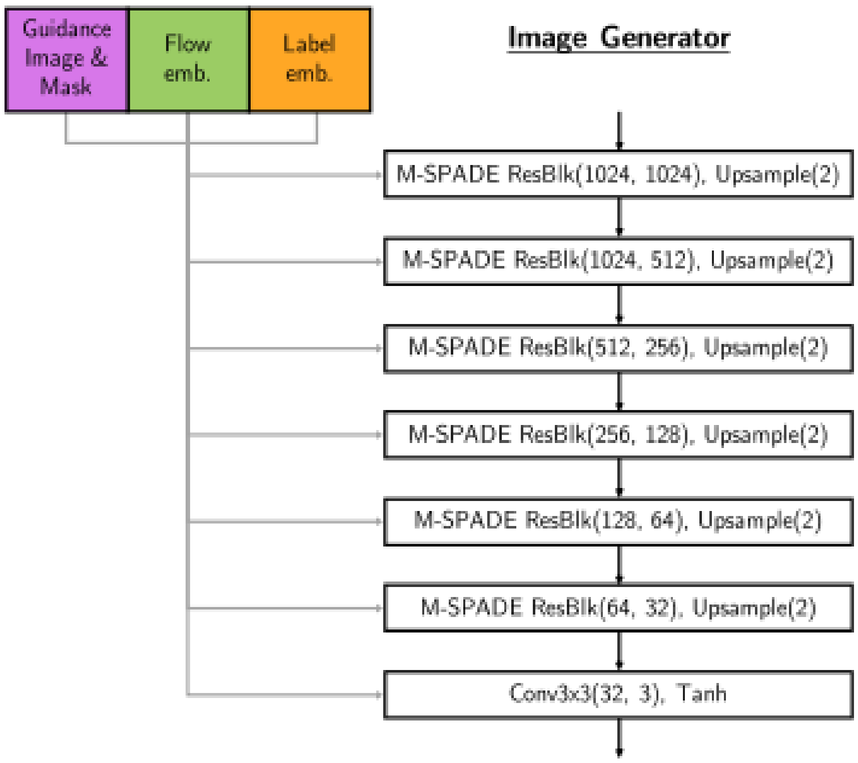

Guidance Image & Mask
Flow emb.
Label emb.
Image Generator
M-SPADE ResBlk(1024, 1024), Upsample(2)
M-SPADE ResBlk(1024, 512), Upsample(2)
M-SPADE ResBlk(512, 256), Upsample(2)
M-SPADE ResBlk(256, 128), Upsample(2)
M-SPADE ResBlk(128, 64), Upsample(2)
M-SPADE ResBlk(64, 32), Upsample(2)
Conv3x3(32, 3), Tanh

Figure 2-6: Architecture of generator used in world-consistent video-to-video synthesis. [4]

Image Generator, as shown in Figure 2-7, consists of Multi SPADE Residual blocks that input Guidance Images, Masks, Flow, and Label Embedding.

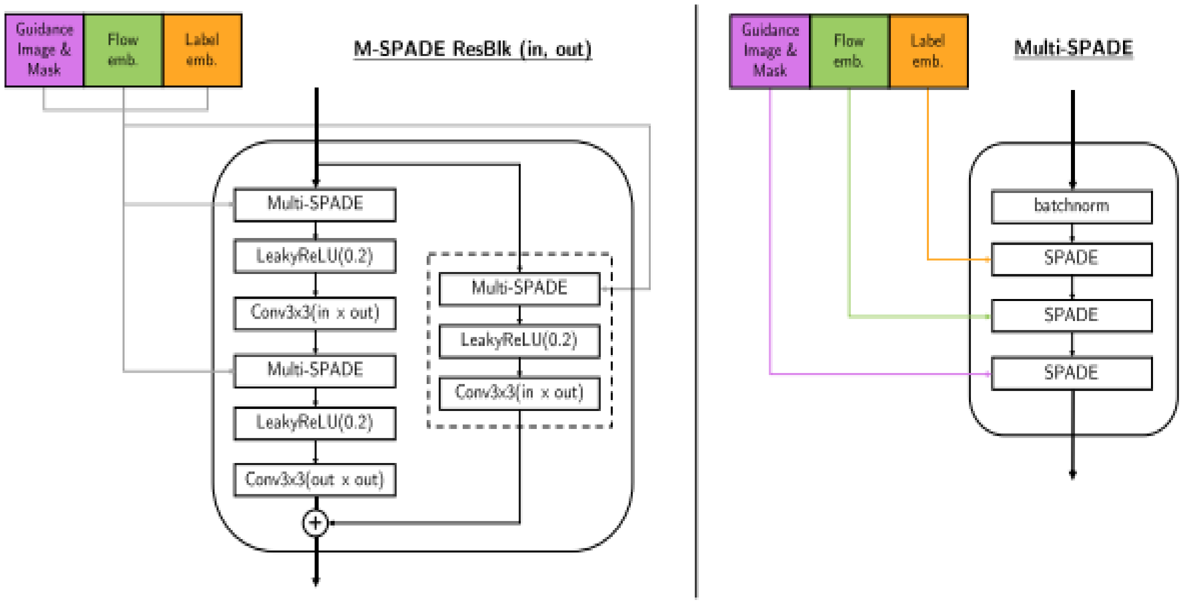


Figure 2-7: Architecture of multi SPADE residual block used in world-consistent video-to-video synthesis. [4]

The Multi SPADE residual block is explained in detail in Figure 2.7. Each block is subdivided into multiple SPADE Residual blocks (3 in this case).

A. Mallya et al. [4] significantly improves spatial and temporal consistency, but it lacks various user perspective applications, scenario generation and different AI rendering aspects.

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs proposed by T. C. Wang et al. [12] present an image-to-image translation method that uses a conditional GAN-based approach to generate high-resolution, realistic images from semantic labels. This study helps understand various scenario-changing possibilities, high-quality outputs, and an interactive environment, which would be helpful from the user's perspective. It uses instance and label mapping both as input to get sharper as well as easily distinguisable outputs.

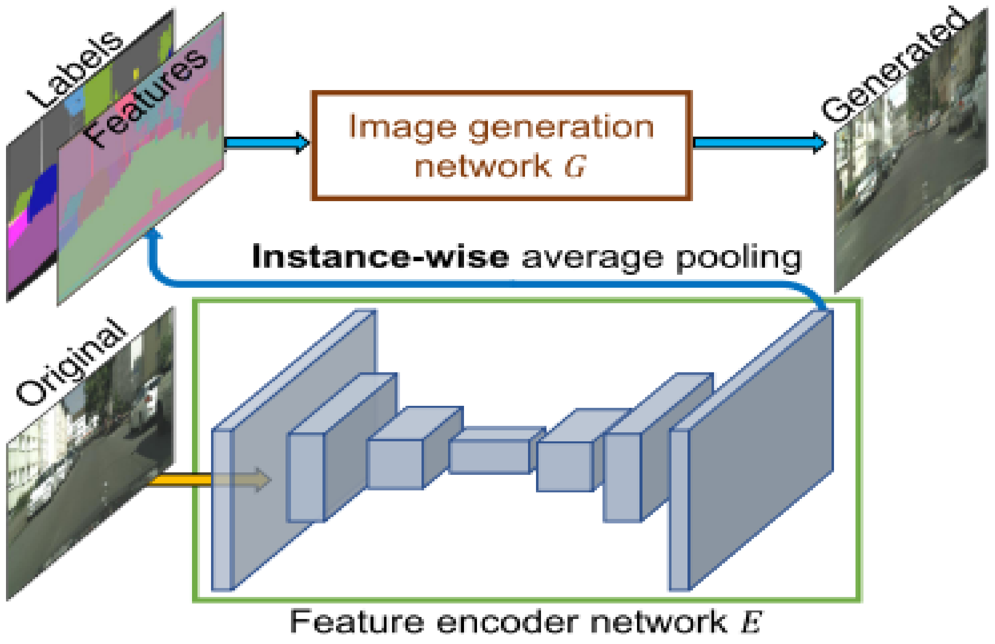

Labels
Features
Image generation network G
Generated
Instance-wise average pooling
Original
Feature encoder network E

Figure 2-8: Instance as well as label used for encoding. [12]

We will use a similar approach for our video synthesis approach, along with previous work on video-to-video synthesis.

Park et.al. [13] proposed a method also known as GauGAN, which shows how Spatially Adaptive Normalization (SPADE), with the help of GAN, can help us remove the ‘wash away’ problem in normalization layers and finally help us to reduce the problems in dealing with morph images and consistency with respect to images.

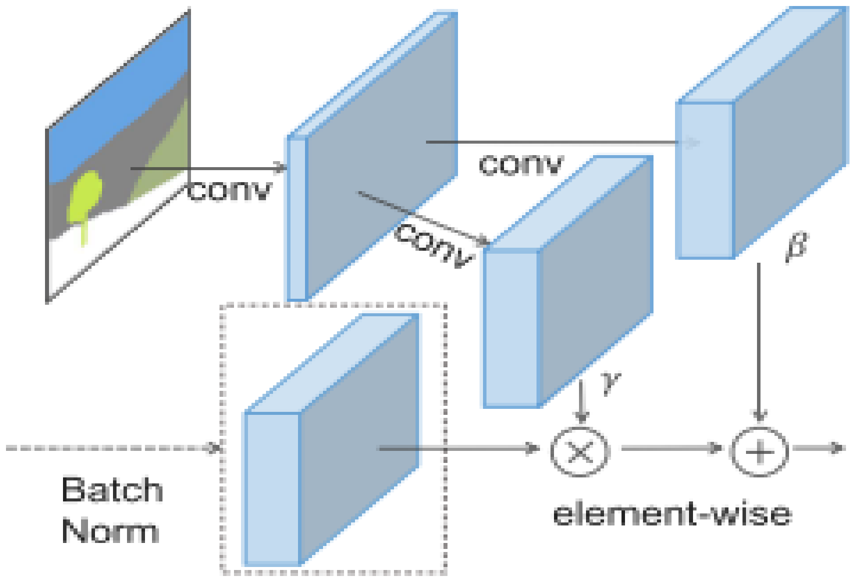


Figure 2-9: SPADE normalization. [13]

The 'wash away' problem is losing semantic information after the normalization layer. Figure 2-9 defines how segmented mask is not directly fed to batch normalization; instead, it is convolved and gives modulation parameters to save semantic information. The problem with this approach is that it lacks different styling information. Thus, we have used a modified version of SPADE in our proposed method.

Zhu et al. [3] describes different styling aspects and the shortcomings of GauGAN [13] which is a Generative Adversarial Networks having spatially adaptive normalization layer. It is mainly related to the quality of output image and styling corresponding to region of interest. SEAN helps us to improve the quality of synthesized images and region-based style encoding. We will use Semantic Region Adaptive Normalization (SEAN) based approach, as shown in Figure 2-10, to get region-based styling information in our proposed solution. SEAN seems to be a prime solution to use it on our proposed method. We are using its style feature to get diverse quality outputs.

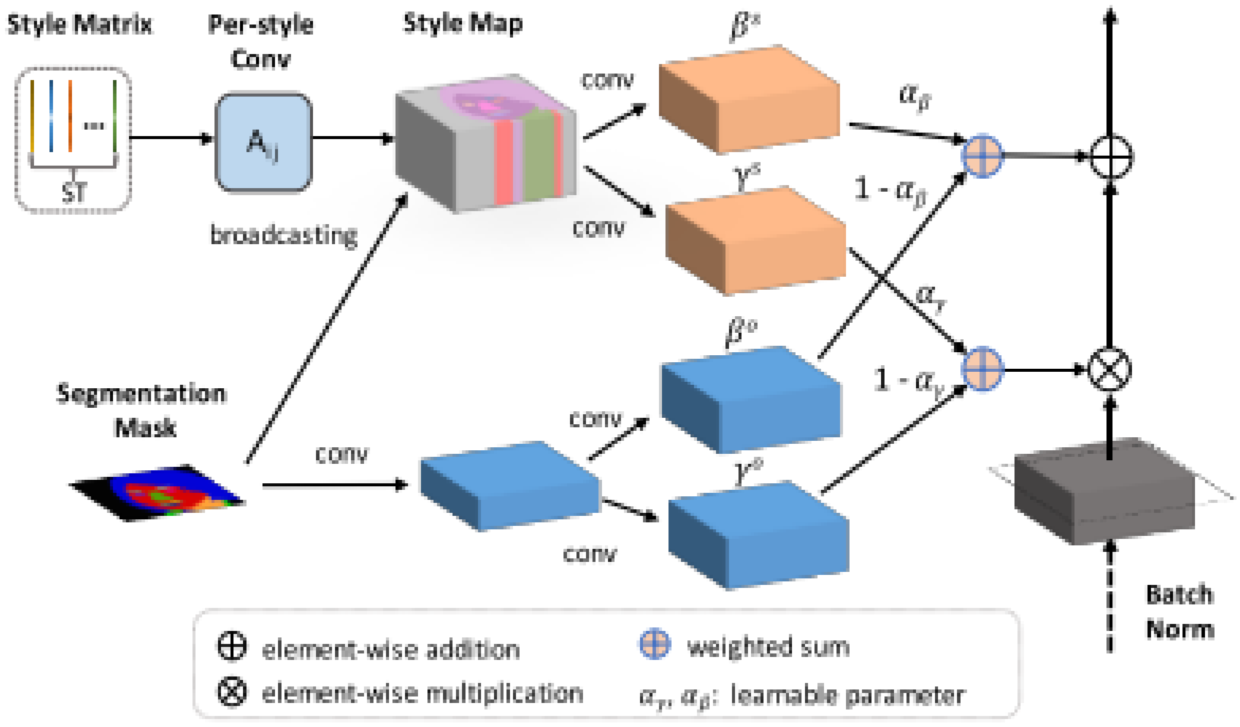


Figure 2-10: SEAN normalization. [3]

UnrealCV: Connecting computer vision to unreal engine proposed by Qiu et al. [9]. It is an open-source plugin that helps us deal with Game Engine outputs (Unreal Engine 4)[17] . We will see the different possibilities of generating inputs (Segmented, depth) for user perspective.

UnrealCV is one of the approaches along with CARLA. It is an open Source simulator powered by Unreal Engine. The generation of test inputs to our proposed system is done using various approaches which includes Unreal Engine integration with ROS Bridge, CarSim[18], and MATLAB [19].

Liao et al. [7] and Cheng et al. [8] discussed methods for generating inputs that will be useful to our proposed model. We will use these approaches to get the required data, which are advanced methods for generating corresponding outputs. Preprocessing while training any deep neural network is an important aspect.

In this study, we use similar configurations of inputs (image and semantic segmented) as proposed in video-to-video synthesis. It employs ideas from image-to-image translation methods for understanding different scenario generation and higher-quality image generation. SEAN seems to be a prime solution in our proposed system (to use its style feature in multiple ways) rather than existing SPADE with respect to quality, diversity, and styling aspects.

[17] *https://www.unrealengine.com/en-US*
[18] *https://www.carsim.com/products/carsim/index.php*
[19] *https://www.mathworks.com/help/matlab/release-notes-R2020a.html*

3 PROPOSALS AND CONTRIBUTIONS

The survey of the state-of-the-art indicates the necessity of the video-to-video synthesis solutions, which should consist of different key aspects as mentioned below in the realistic data generation process:

1. Scenario generation.
2. Label changing.
3. Adding different realistic objects.
4. Avoiding morphs.
5. Reduce flickering/distortion.

Our solution should focus on these key aspects and create an interactive environment to provide a one-stop solution for different AI rendering, scenario generation, and future video prediction problems. Our approach is to generate realistic data, which aims to provide a one-stop solution for all these aspects and explore how an end user can eventually use our system.

Having the right contextual information is vital to have meaningful datasets for autonomous driving. There may be requirements to convert the legacy data captured with smaller Field of View (FOV) cameras to larger FOV cameras, or there could be a need to generate larger datasets from limited data available from a specific geography.

On the other hand, in certain areas, there may also be limitations in sharing personal information such as human faces, vehicle number plates, etc. (for example, GDPR in the EU). Such personal information needs to be anonymized. This data must also be augmented with synthetic data for training or validation. These requirements can differ depending on the context, region, and scenarios. It would be challenging to have a simulation platform to generalize these different parameters.

The proposed model provides flexibility to add the limited unique real data in already available datasets with limited training. Existing Simulator based approaches are not so cost effective. 3D asset creation with these simulators is a complex process. Also generated output seems to be far from reality.

In this approach, the proposed methodology provides a fusion of generative AI, deep learning, and image processing techniques. It explores the possibilities of generating data from simulated, real, or fused environments. The proposed approach is a panoptic segmentation-based approach to create semantic labels. Introducing a flow map-based methodology taking cues from a sequence of frames helps address long-term temporal coherence, which is a key issue in generated data. This input to the Generative AI network is a novel methodology to handle dynamic objects, scene changes, and environment variations.

4 METHODOLOGY

Our methodology to generate realistic data is a GAN (Generative Adversarial Networks)-based approach in which, given an input image, the generator first synthesizes a fake image. Then, the fake image is combined with the input image and put into the discriminator, which should be able to determine that it's fake. On the other hand, when a real image is used, the discriminator should be able to give output as it is real. So, during generator training, the generator's goal is to fool the discriminator and make its output real.

4.1 GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) are the models that generate new data instances, which is like a GAN-based approach to generating synthetic outputs; that is why we call these models 'generative.' For example, GANs can create fake outputs that are like the real world, even when these outputs do not belong to any real-world scenario. GANs reduce the reality gap by combining a generator, that trains the model to produce an output, with a discriminator, that trains that model to differentiate real data from the generator's fake output. The generator trains to fool discriminators, and the discriminator tries to discriminate as much as possible to not be fooled.

- Generative models can generate new data instances by taking joint probability.
- Discriminative models discriminate distinct kinds of data instances by using conditional probability.

4.2 TRAINING OF GENERATIVE ADVERSARIAL NETWORKS

GANs training requires both generator and discriminator training simultaneously over every epoch. It follows a below pattern:

1. Training of discriminator for n epochs.
2. Training of generator for n epochs.
3. Repeat the training process as in 1 and 2.

We usually keep the generator constant during the training of the discriminator and vice versa.

4.3 OBJECTIVE FUNCTIONS IN COMMONLY USED GANS

The objective function in GANs consists of the commonly used Loss function known as Minimax Loss. The generator tries to minimize the objective function value and the discriminator tries to maximize it.

$$min\, max\ L(G,D) = E_X\left[\log\left(D(x)\right)\right] + E_Z\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

(1)

- Ex is the expected value of real input.
- Ez is the expected value of generated fake instance.
- D(x) is the expected estimate of the discriminator for the real instance to be fake.
- D(G(z)) is the expected estimate of the discriminator for the fake instance to be real.

4.4 GENERATOR ARCHITECTURE

The generator consists of a Multi SEAN-based image generator in which inputs are segmented as Image embedding, Flow and Level embedding as well previous frames outputs. Figure 4-1 and Figure 4-2 give an overview of our generator architecture. The generator comprises a series of Multi SEAN residual blocks and up-sampling layers. The structure of each Multi SEAN residual block is shown in Figure 4-2, which replaces the SEAN layers in the original SEAN residual blocks with Multi SEAN layers.

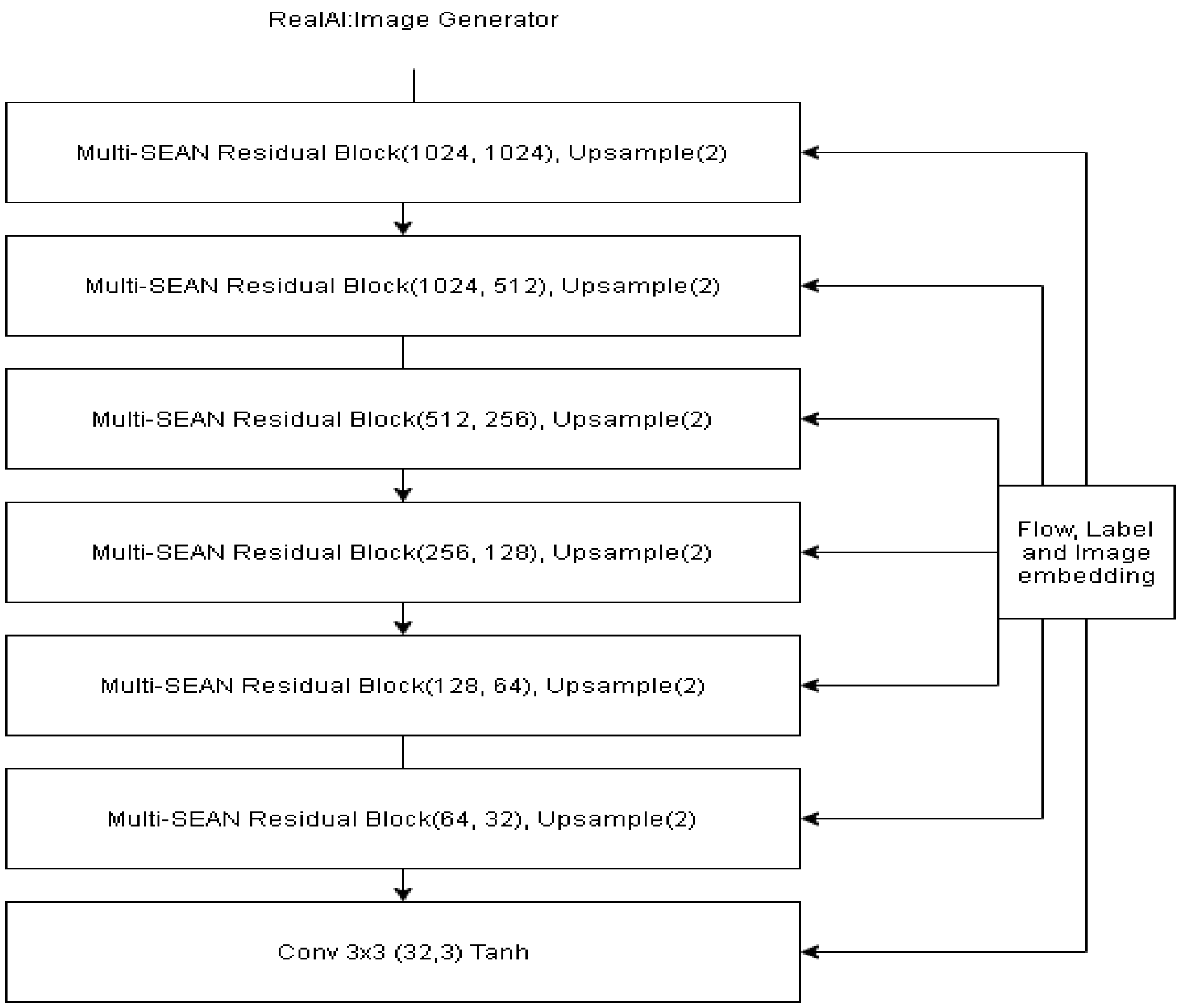

RealAI:Image Generator
Multi-SEAN Residual Block(1024, 1024), Upsample(2)
Multi-SEAN Residual Block(1024, 512), Upsample(2)
Multi-SEAN Residual Block(512, 256), Upsample(2)
Multi-SEAN Residual Block(256, 128), Upsample(2)
Multi-SEAN Residual Block(128, 64), Upsample(2)
Multi-SEAN Residual Block(64, 32), Upsample(2)
Conv 3x3 (32,3) Tanh
Flow, Label and Image embedding

Figure 4-1: Multi SEAN-based image generator.

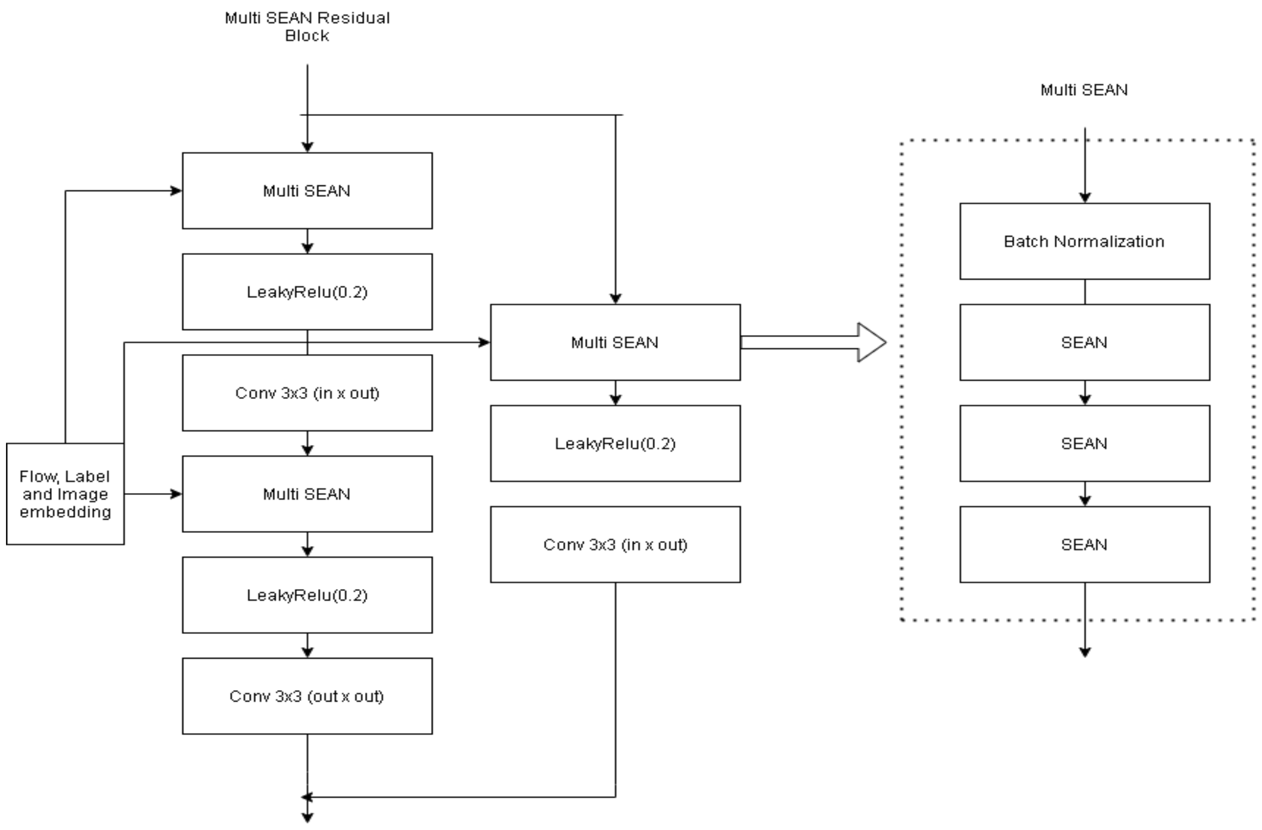


Figure 4-2: Multi SEAN residual block.

The inputs to these SEAN blocks are Flow, Image, and Label embeddings based on encoder-decoder architecture. Label encoding uses an encoder-decoder style[20] to embed input labels to distinctive features, which acts as one of the inputs to Multi SEAN in the Image generator. Flow embedding deals with optical flow outputs of the previous frame and is again fed through the Multi SEAN block of the image generator. Image and segmentation generator encodes image and segmented frames, respectively. Image encoder uses previously generated frames while segmentation encoder uses first frame semantics.

Figure 4-3 describes our system's Flow or Label embedding, Image, and Segmentation decoder. This helps to generate inputs for the image generator.

[20] *https://ieeexplore.ieee.org/document/9076374*

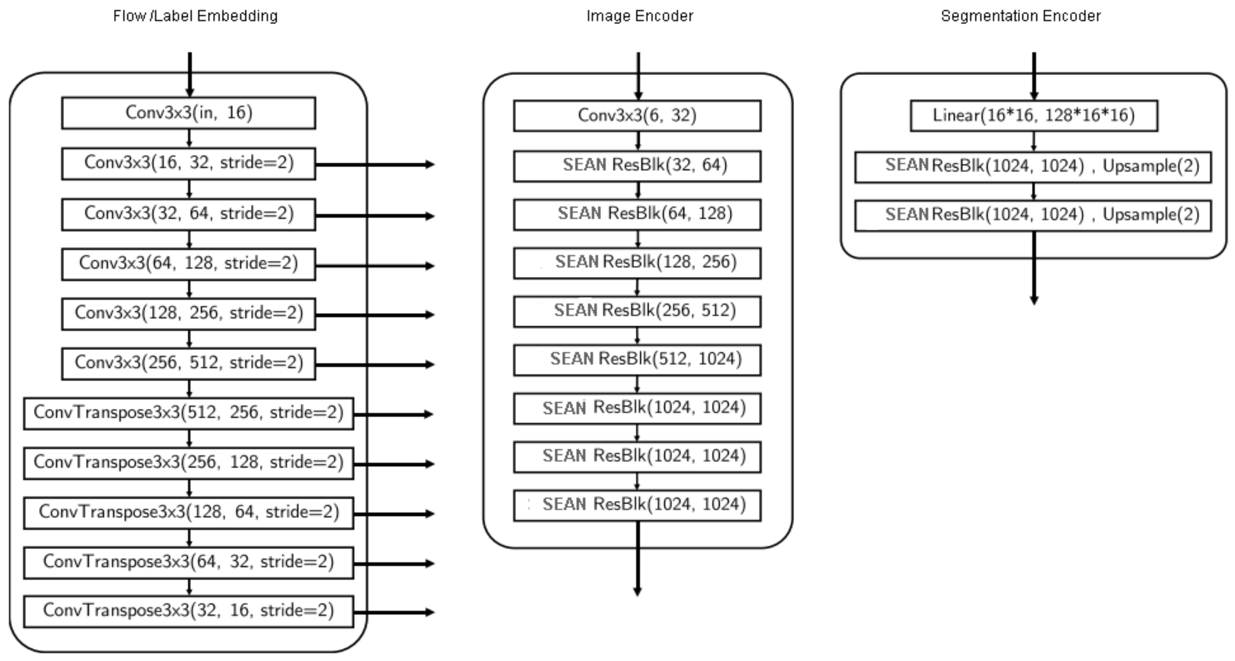

Flow /Label Embedding
Conv3x3(in, 16)
Conv3x3(16, 32, stride=2)
Conv3x3(32, 64, stride=2)
Conv3x3(64, 128, stride=2)
Conv3x3(128, 256, stride=2)
Conv3x3(256, 512, stride=2)
ConvTranspose3x3(512, 256, stride=2)
ConvTranspose3x3(256, 128, stride=2)
ConvTranspose3x3(128, 64, stride=2)
ConvTranspose3x3(64, 32, stride=2)
ConvTranspose3x3(32, 16, stride=2)
Image Encoder
Conv3x3(6, 32)
SEAN ResBlk(32, 64)
SEAN ResBlk(64, 128)
SEAN ResBlk(128, 256)
SEAN ResBlk(256, 512)
SEAN ResBlk(512, 1024)
SEAN ResBlk(1024, 1024)
SEAN ResBlk(1024, 1024)
SEAN ResBlk(1024, 1024)
Segmentation Encoder
Linear(16*16, 128*16*16)
SEAN ResBlk(1024, 1024) , Upsample(2)
SEAN ResBlk(1024, 1024) , Upsample(2)

Figure 4-3: Architecture of label/flow embedding, image and segmentation.

4.5 DISCRIMINATOR ARCHITECTURE

The Discriminator Architecture uses PatchGAN architecture [14], as shown in Figure 4-4 in which we will first take a different real and synthesized image of distinct resolution. Each one is fed to a different discriminator based on 70x70 PatchGAN to get a consistent output both locally and globally.

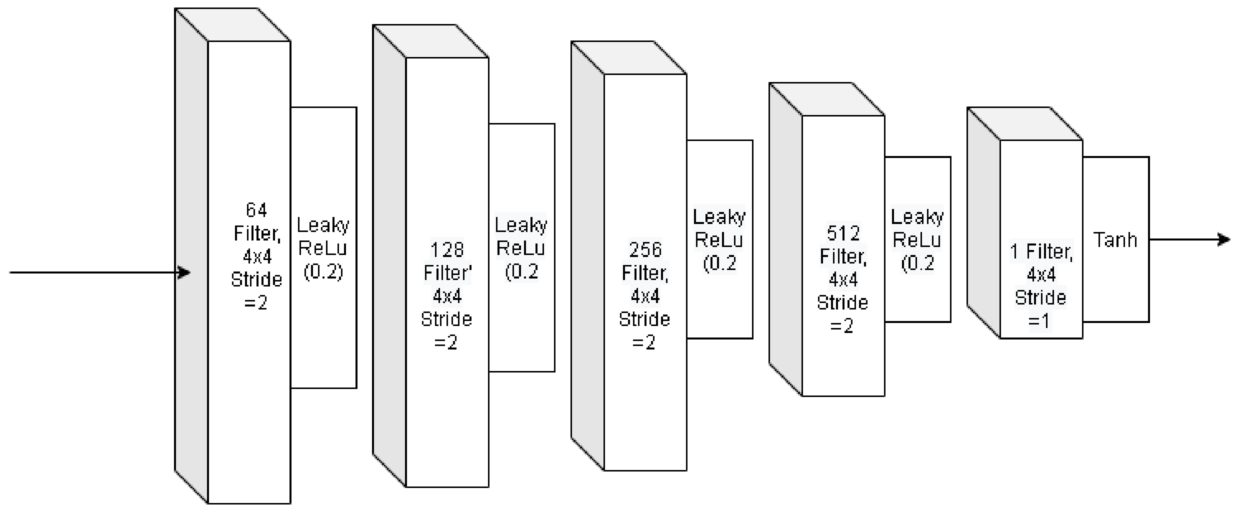


Figure 4-4: Architecture of PatchGAN.

4.6 Objective Functions Used in Proposed GAN

The objective functions in GANs consist of commonly used Loss functions known as Hinge Loss, feature extractor loss, and flow loss, like what was used in existing video-to-video synthesis methods [4]-[6]. Hinge loss is calculated as in (2). The generator tries to minimize, and the discriminator tries to maximize the objective function.

$$L_{Hinge}(G,D) = E_{X}\left[\max\left(0,(1-D(x)\right)\right] + E_z\left[\max\left(0,(1+D\left(G(z)\right)\right)\right]$$

(2)

Here Ex, Ez are expected value real inputs and random inputs, respectively, D(x) and D (G (z)) are the expected estimate of discriminator for real instance to be real and fake instance to be real, respectively.

Feature extractor loss, as in (3), is discriminator feature based on VGG19 Loss, which is calculated using the below formula. $\emptyset_i(G)$, $\emptyset_i(R)$ are features of VGG19 layers of generated and real images.

$$L_{vgg} = |\emptyset_i(G) - \emptyset_i(R)|$$

(3)

Flow loss is based on Flownet2.0[21] architecture which contains:

- Flow L1 loss: L1 loss with respect to ground truth flow.
- Flow warp loss: L1 loss between warped and target image.
- Flow mask loss: Loss of occlusion mask.

Calculation of each flow loss is done by as by changing respective input values x and y in (4).

$$L(x,y) = \{L_1, L_2, \ldots L_n\}^T, \;\; L_n = |x_n - \;\; y_n|$$

(4)

Combining all these objective functions, the generator aims to minimize, and the discriminator maximizes the function value to get a model with good generative capability.

[21] *https://arxiv.org/abs/1612.01925*

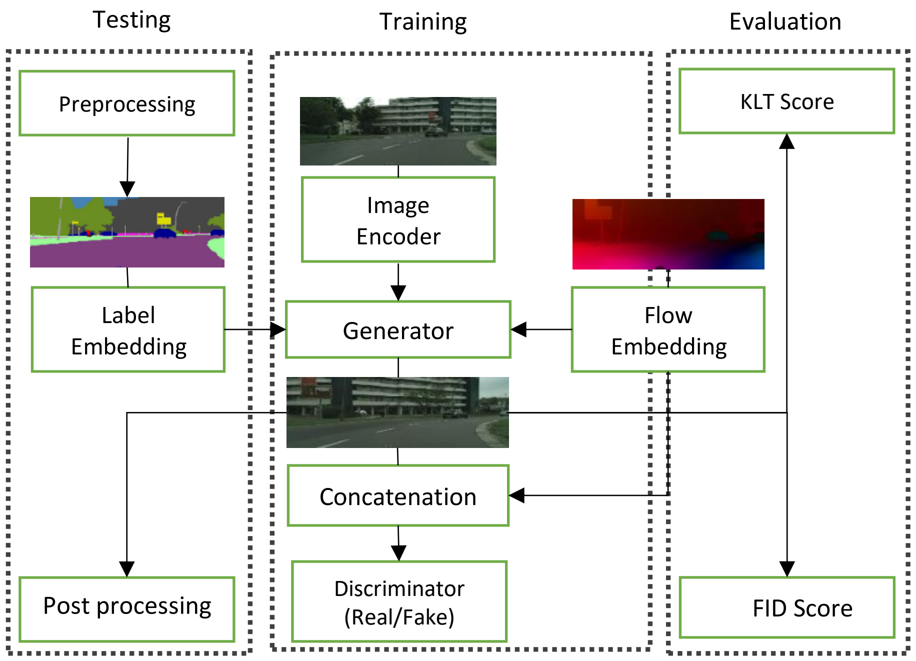


Figure 4-5: Overall system architecture.

Figure 4-5 describes the overall project architecture in which our input labels, optical flow outputs, and guidance images are fed to Multi SEAN-based architecture, for which we will take care of the styling part in our proposed model. We will use the cascading approach to improve the quality of produced output and start training on lower-resolution images first, then shift to higher-resolution images. The detailed description of generator and discriminator architecture and Generative Adversarial Networks (GANs) have already been described in the above sections.

4.7 Training and Validation

Our proposed model is trained and validated on Google Collaboratory[22] . ADAM[23] optimizer at a learning rate of 0.00008 is used for the encoder and generator networks, while a learning rate of 0.0005 is used for the discriminators described above. This study uses the Cityscapes[24] dataset for training. The dataset has been resized to 384x768 pixels and 512x1024 pixels and trained on a single GPU environment, which takes approximately 100 hours for training. This study adopted a cascading approach to train with 384x768 pixels first and then with 512x1024 pixels on the same network. Our model is tested on the CamVid[25] with the data generated from CARLA, Carla ROS Bridge, CarSim, Matlab, and Unreal CV (Computer Vision). For the segmentation data

[22] *https://colab.research.google.com/*
[23] *https://arxiv.org/abs/1412.6980*
[24] *https://arxiv.org/abs/1604.01685*
[25] *https://www.sciencedirect.com/science/article/abs/pii/S0167865508001220*

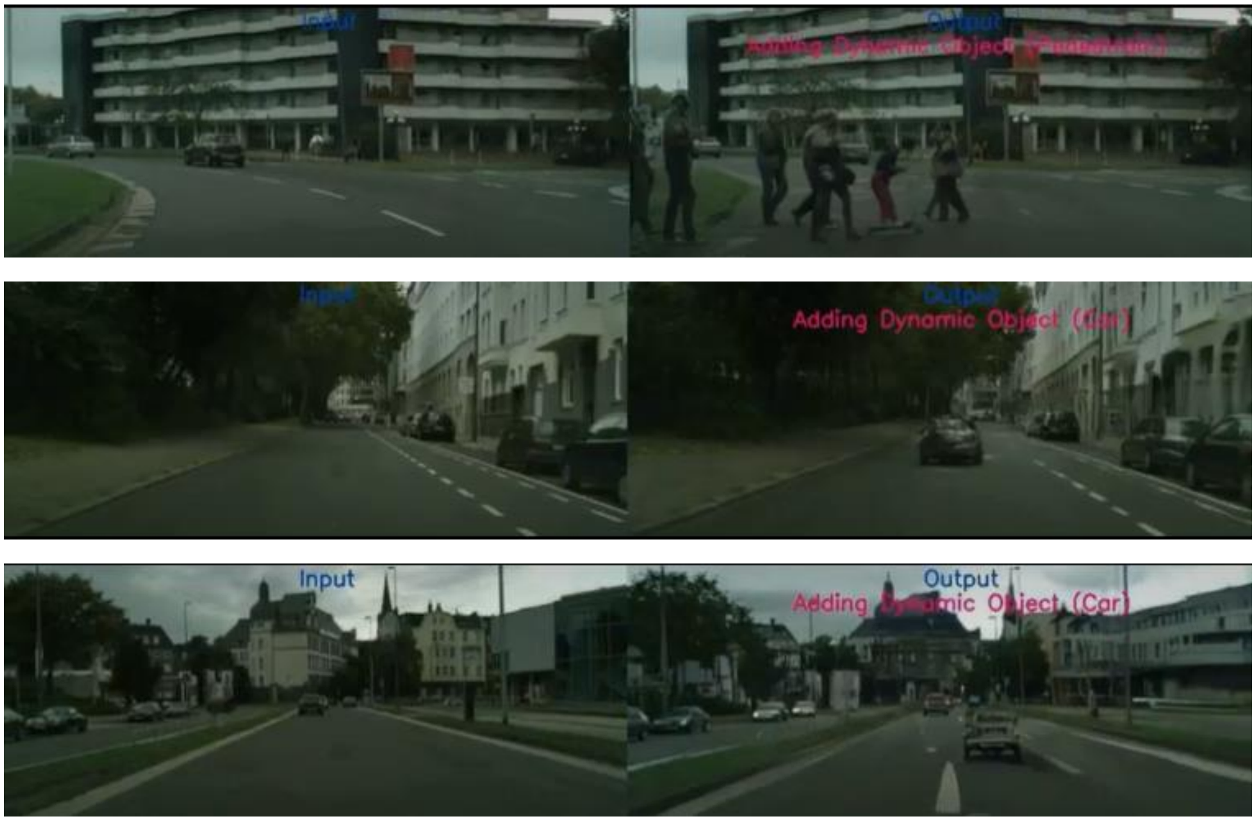
generation, we used pre-trained network DeepLabV3+[26]. We use FlowNet 2.0 [21] architecture to extract the optical flow as the ground truth flow. For measuring the closeness of output to the real world, we will be using the following:

- Fréchet Inception Distance (FID) -Inception-v3 network[27]
- Kanade-Lucas-Tomasi (KLT)-Based Score.

5 Results and Conclusions

The outputs of scenario generation which include changing environment, extended road and adding realistic objects (like pedestrians, car, bike etc.) are shown in Figure 5-1. This is achieved by converting color maps (semantic maps) as required. In extended road scenario generation, the proposed system will read color map for road and map the same into surroundings to generate corresponding semantic map and thus generate output. Similarly, for adding the objects as required, a masked area with the right flow and contextual information is added to the color map to get required scenarios.

5.1 Sample Frame Output



[26] *https://arxiv.org/abs/1706.05587*
[27] *https://arxiv.org/abs/1512.00567*

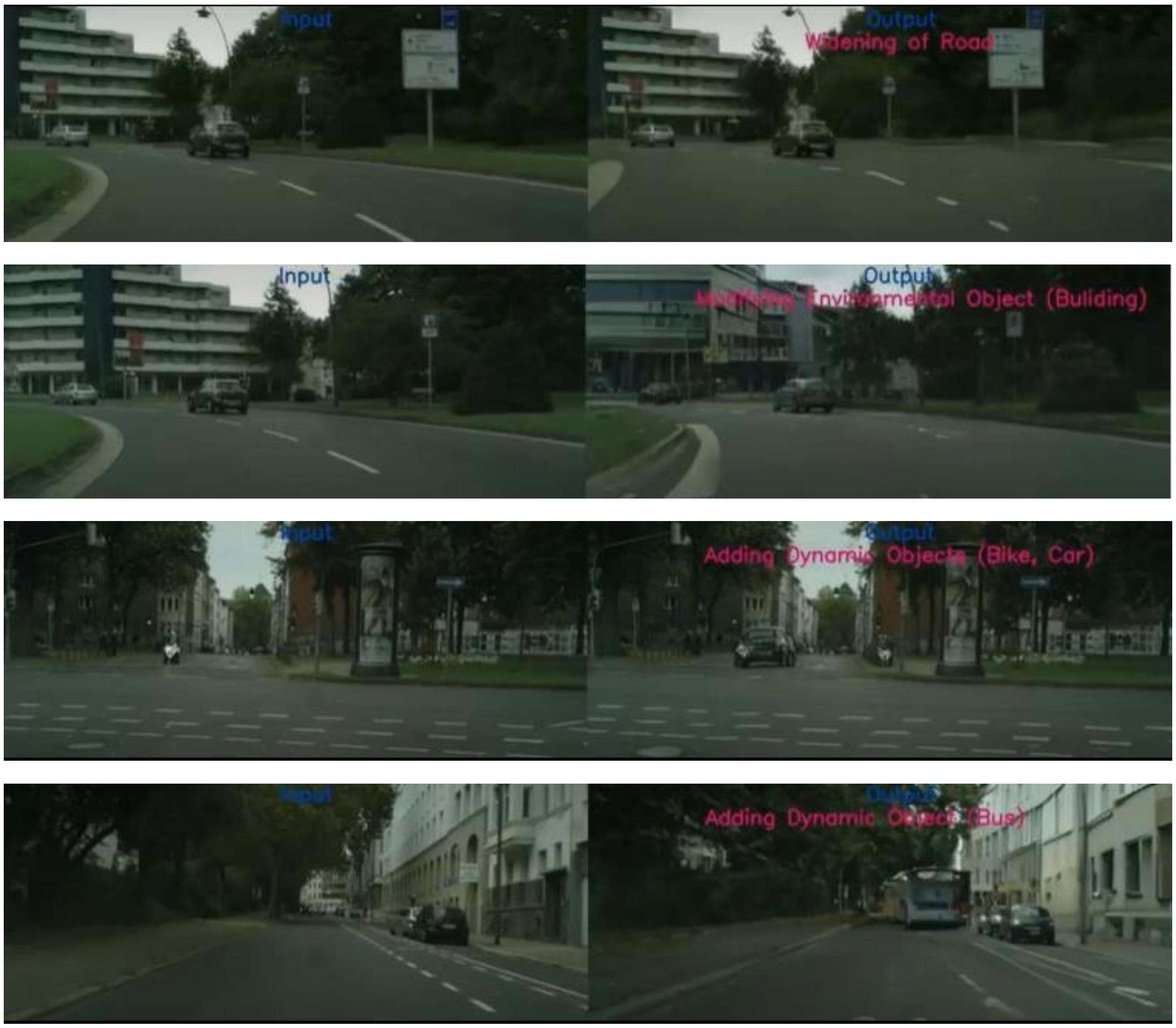

Input
Output
Widening of Road
Input
Output
Modifying Environmental Object (Buliding)
Input
Output
Adding Dynamic Objects (Bike, Car)
Input
Output
Adding Dynamic Object (Bus)

Figure 5-1: Scenario generation outputs.

5.2 KPI (Key Performance Indicators) Evaluation

KPI evaluation of our proposed system is done using two ways. In the KPI Evaluation A (Table 5-1), we have calculated the model performance over the same scenarios using two different training strategies. While in the KPI Evaluation B (Table 5-2), the synthetic data generation approach was leveraged to create training data for object detection tasks to improve the KPIs of automated labelling. Below is the improvement in KPI achieved by leveraging the synthetic data generated for training the automated algorithm (Refer to Table 5-2).

Parameters	Approach A	Approach B
FID (Fréchet inception distance) Score	180	115.4
KLT (Kanade Lucas Tomasi) Score	0.2154	0.0300

Table 5-1: KPI Evaluation A.

Approach A mentioned in Table 5.1 is trained with 4 input frames at start and doubling it after every 25th Epoch up to a maximum of 32 frames. That is training will start with batch of continuous 4 frames and subsequently after every 25th epoch the batch of input frame will change to 8, 16 and 32.

Similarly in Approach B training started with giving 2 frames and start and doubling after every 25th epoch up to a maximum of 32 frames.

Parameters	Approach 1 (No generated data used for training)	Approach 2 (With generated + original data)
% of accurate detection	75%	91%

Table 5-2: KPI Evaluation B.

FID is a commonly used evaluation score to measure the closeness of two images (here, in our case, the original image and the generated image). The lower the FID value, the better the output image/video quality. FID value 0 signifies there is no difference between input and generated image/video.

FID score is calculated by taking the mean (μ_1 , μ_2) and covariance (C1, C2) of feature vectors by considering the pre-trained Inception-v3 model.

$$\mathrm{FID} = ||\mu_1 - \mu_2||^{2} + \mathrm{Tr}\left(C_1 + C_2 - 2 * \sqrt{(C_1 * C_2)}\right)$$

Similarly, our proposed KLT score further reduces human intervention and gives a score to measure output quality based on temporal coherence between consecutive frames. The Lower KLT score signifies there is consistency between successive frames. Like FID score, “0” KLT score is the ideal case which means all frames are consistent.

KLT Score is calculated using the degree of similarity between consecutive frames of selected optical flow-based features.

The mean score to compute relative distance is calculated using the given formula where μ_r and μ_g are optical feature vector distances for real and generated frames respectively.

$$D^{2}_{KLT} = \left\|\mu_r^2 - \mu_g^2\right\|$$

5.3 Conclusion

- The proposed model could generate realistic scenarios if given different segmented inputs. It would perform better with increasing frames and cascading layers.
- FID and KLT score-based proposed metrics will help us evaluate any video synthesis models and eventually reduce human intervention (as shown in Table 5-1).
- The proposed model helps us generate adversarial data (as shown in Figure 5.1), which eventually helps improve object detection capabilities (as shown in Table 5-2).

References

[1] N. Kalra and S. M. Paddock, *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Santa Monica, CA: RAND Corporation, 2016.

[2] K. K. Patel, “A Simulation Environment with Reduced Reality Gap for Testing Autonomous Vehicles,” University of Windsor, 2020.

[3] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, “SEAN: Image synthesis with semantic region-adaptive normalization,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 5103–5112, 2020, doi: 10.1109/CVPR42600.2020.00515.

[4] A. Mallya, T.-C. Wang, K. Sapra, and M.-Y. Liu, “World-Consistent Video-to-Video Synthesis,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12353 LNCS, pp. 359–378, Jul. 2020, doi: 10.1007/978-3-030-58598-3_22.

[5] T. C. Wang *et al.*, “Video-to-video synthesis,” *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, pp. 1144–1156, 2018.

[6] T. C. Wang, M. Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro, “Few-shot video-to-video synthesis,” *arXiv*, 2019.

[7] M. Liao, F. Lu, D. Zhou, S. Zhang, W. Li, and R. Yang, “DVI: Depth Guided Video Inpainting for Autonomous Driving,” *arXiv*, pp. 1–16, 2020.

[8] B. Cheng *et al.*, “Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 12472–12482, 2020, doi: 10.1109/CVPR42600.2020.01249.

[9] W. Qiu and A. Yuille, “UnrealCV: Connecting computer vision to unreal engine,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9915 LNCS, pp. 909–916, 2016, doi: 10.1007/978-3-319-49409-8_75.

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” in *Advances in Neural Information Processing Systems*, 2017, vol. 2017-Decem, no. Nips, pp. 6627–6638.

[11] Jianbo Shi and Tomasi, “Good features to track,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994, vol. 178, pp. 593–600, doi: 10.1109/CVPR.1994.323794.

[12] T. C. Wang, M. Y. Liu, J. Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8798–8807, 2018, doi: 10.1109/CVPR.2018.00917.

[13] T. Park, M. Y. Liu, T. C. Wang, and J. Y. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 2332–2341, 2019, doi: 10.1109/CVPR.2019.00244.

[14] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 5967–5976, doi: 10.1109/CVPR.2017.632.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, “CARLA: An open urban driving simulator,” no. CoRL, pp. 1–16, 2017.

[16] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2014, doi: 10.1145/3422622.

[17] Epic Games, “Unreal Engine.” 2019, [Online]. Available: https://www.unrealengine.com.

[18] Mechanical Simulation, *Introduction to CarSim*, no. July. 2020.

[19] MATLAB, *9.8.0.1323502 (R2020a)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[20] S. O. Patil, V. . Sajith Variyar., and K. P. Soman, “Speed Bump Segmentation an Application of Conditional Generative Adversarial Network for Self-driving Vehicles,” in *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Mar. 2020, no. Iccmc, pp. 935–939, doi: 10.1109/ICCMC48092.2020.ICCMC-000173.

[21] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of optical flow estimation with deep networks,” in *Proceedings - 30th IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2017, 2017, vol. 2017-Janua, pp. 1647–1655, doi: 10.1109/CVPR.2017.179.

[22] E. Bisong, “Google Colaboratory,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Berkeley, CA: Apress, 2019, pp. 59–64.

[23] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980.

[24] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 3213–3223, 2016, doi: 10.1109/CVPR.2016.350.

[25] G. J. Brostow, J. Fauqueur, and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2009, doi: 10.1016/j.patrec.2008.04.005.

[26] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv*, 2017.

[27] C. Szegedy, V. Vanhoucke, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

ACKNOWLEDGEMENTS

The views expressed in the *OMG Journal of Innovation* are the author’s views and do not necessarily represent the views of their respective employers nor those of the Object Management Group® (OMG®).

➢ Return to *OMG Journal of Innovation landing page* for more articles and past editions.