2. Literature Review

This chapter discusses the various other related works. The first part explores some state of art autonomous vehicle simulators and identifies its key features, advantages and limitations. The subsequent part summarizes about the preceding approaches for defining a metric for the comparison of Autonomous vehicle simulators. The last part of this chapter discusses about the different types of Generative Adversarial networks (GANs), a Generative AI algorithm which is applied in this thesis and highlights the key advantages in those approaches.

2.1 State of art simulators

There are numerous simulators available in the market, however this section discusses about some of the relevant and popular simulators

2.1.1 CARLA

CARLA (Car learning to act) is an open-source software developed by the Computer Vision Centre (CVC) and the Barcelona Supercomputing Centre (BSC) in collaboration with the Toyota Research Institute for research and development of autonomous driving. It provides realistic and diverse environments with various urban scenarios, climates and sensors. It is developed using unreal engine 4 for rendering high quality visual effects and OpenDRIVE standard 1.4 to define roads and urban settings. The figure 1 depicts a scene from the simulators in various weather. The key feature of this simulator is that it is built as a server client architecture. The server handles the physics and computation of simulators whereas the user can control the simulator using C++ and python APIs making it scalable. Another upside of this simulator is that it facilitates the flawless process of developing, training and validating machine learning algorithms. Various algorithms like modular pipeline, Imitation learning, Reinforcement learning [2] can be trained and validated in this simulator making it one the best choices for researchers. Moreover, it provides variety of sensor data such as cameras, Lidars, various meta data and ground truth which makes it more beneficial and handy. This simulator also provides access to various digital assets (actors) in the environment carefully designed to maintain high level of realism. However, currently it supports only 2 pre-defined urban maps with 2.9 kms and 1.4 kms of driving failing to provide diversity and generalization.

2.1.2 LGVSL

LG Silicon Valley lab (LGSVL) is developed by LG electronics as an open-source simulation engine. It uses Unity gaming engine to render photo realistic environments and also takes advantage of technologies such as High-definition render pipeline (HDRP). The simulator is developed into two parts, Simulation engine and user autonomous driving (AD) stack. The simulation engine is open source and it receives input from the AD stack and simulates the environment, sensors and vehicle. The AD stack consist of three parts Perception, planning and control which can be configured by user. The AD stack and simulation engine are connected through communication bridge interface such as Cyber RT. The simulator comes with various default set of sensors such as camera Lidar, Radar… However, the one of the key feature of this simulator is that the user can built and configure their own sensor. In addition to default sensors, Model of real world sensors can be imported as a plug in. For Example, Velodyne VLP-16 LiDAR generate point clouds in the same format as real sensors. The sensor data and its position can be exported and defined using JSON formatted text which make it easy to use The unique feature of this simulator is it’s ability to use real world maps to build the environment. Maps in formats like Lanelet2, OpenDRIVE and Apollo 2 HD map can be and used as the virtual environment. This makes it more suitable for researchers in OEMs

2.1.3 SUMMIT

A Simulator for Urban Driving in Massive Mixed Traffic (SUMMIT) a open source simulator developed as an extension of CARLA simulator inheriting its physics and visual realism. It uses Python based APIs to communicate with CARLA Most of the other simulators simulates a rule-based traffic where all the actors acts according to a rule with minimal degree of randomness. However, the real world traffic is comparatively aggressive and chaotic. The distinguishable feature of SUMMIT is that is simulates the aggressive and chaotic behaviour of the traffic in real world. This attracts users who are interested in training and testing the algorithms which drives vehicles in un regulated traffic. In this simulator, a crowd behaviour algorithm “Context - GAMMA”, a velocity – space optimization algorithm is used to simulate the traffic behaviour geometrically and topologically. Moreover, it uses the real world map from OpenStreetMap to extract features such as roads, sidewalks, roundabouts, which can be further used in the simulator to replicate real world maps. The figure3 shows the Real map and its counterpart with un regulated traffic behaviour in SUMMIT of Magic- Roundabout, England. This feature of using real world map and taking advantage of Visual realism from CARLA and simulating chaotic traffic behaviour makes it even powerful.

2.1.4 Gazebo

Gazebo is an open source simulator used widely in robotics and autonomous systems. It is developed at Carnegie Mellon University in early 2000s. It is focussed mainly on the mobile robots. It stands out from other simulator in a way that it simulates the physics of non players within the environment accurately. In this simulator The Open Dynamics Engine [8], created by Russel Smith is a widely used physics engine in the open source community. It is designed to simulate the dynamics and kinematics associated with articulated rigid bodies. This engine includes many features such as numerous joints, collision detection, mass and rotational functions, and many geometries including arbitrary triangle meshes. This simulator Can accommodate several robots (in terms of ten) simultaneously which is beneficial when working in applications with multiple robots such as autonomous driving. Several robots, sensors, actuators collectively called as model can be added to simulator easily using Apis. Moreover, new 3D user defined objects can be created using OpenGL, a standard library for 3d object creation and GLUT (OpenGL utility toolkit) provides an interactive user interface. Though the simulator can accurately simulate the physical phenomenon, the larger environment required by autonomous driving in real world proves to be challenging to simulate. However, since it is an open source software several custom simulators are being developed by the users according to their requirements

2.1.5 TORCS

TORCS (The Open Racing Car Simulator) is an open source platform for racing car simulations developed in early 2000s. The notable feature of the torcs is its ability to configure the parameters of the car, which in crucial in a racing car simulations. The engine uses a discrete-time simulation, with simple Euler integration of differential equations. The discretisation is set to 0.002s of simulated time. In TORCS, the participating players are referred to as \robots". They are loaded as external modules in TORCS. This means that new articially intelligent agents can be developed independently and they only have to satisfy the basic API requirements for robot code. In addition, TORCS excels in emulating authentic racing scenarios, encompassing vehicle dynamics, track configurations, weather variations, and interactive environments. his flexibility extends to academic circles, where TORCS serves as a valuable testbed for evaluating algorithms related to artificial intelligence, machine learning, and autonomous driving. Challenges include scalability concerns in managing complex simulations and the aspiration for heightened realism in vehicle behavior and environmental interactions. The open-source nature of TORCS has facilitated a vibrant community of developers, researchers, and racing enthusiasts.

2.2 Comparative study on simulators

This section will analyse some other work involving in comparative study of Autonomous vehicle simulators and summarizes the analysis

In the work of Guan Yang, et.al., (2021) in the “Survey on autonomous vehicle simulation platform”, the team had conducted an extensive research on various autonomous vehicle simulation platform. They had broke down the objective of the simulator into 5 parts i.e., Static environment simulation, Dynamic environment and behaviour simulation, Traffic flow simulation, Sensor simulation and vehicle dynamics simulation. Moreover, for they have defined a taxonomy for existing simulator, They categorized the simulation platform into Point cloud based and 3D engine. Point based simulators are the one which simulates the sensory data and reconstructs the environment based on sensor data. Some example of this simulators are CarCraft from Waymo and Apollo from Baidu. Figure 4 shows the map from Apollo a point based simulation platform, whereas 3d engine based platform uses gaming 3 d engines such as Unity, Unreal to render a environments in accordance with laws of real physics. Figure 5 shows a map of PanoSim, a 3d engine based simulator. A Table with comparision simulators and its available feature is framed and the same is shown in figure 6. Thoug this table compares helps to compare the simulators the no of features is not enough for a concrete decision and there is no single metric which define the useability of that simulator to the user. It provides a categorization of simulators but a comparative method among simulator is not clearly defined

In the work of Md Salman Ahmed et.al (2016), an extensive research on connected vehicle simulator was discussed. The domain of connected vehicles includes Vehicle to vehicle communication, vehicle to server communication… and requires a simulator to train as it will be expensive to train in real world. In this paper several simulators which simulates the Vehicle communication system and compared based its memory consumption, computing environment (Sequential or Parallel) and no. of vehicles it can handle and the results are summarized. However these results corresponds to a specific domain of Connected and this method cannot hold for any other type of AV simulators.

2.3 Generative Advesarial Networks

Generative adversarial networks (GAN) is a Generative AI algorithm introduced in 2014 by Goodfellow et.al in the paper “Generative Adversarial Networks” [7]. From then it gained momentum in domain of Generative AI. In this Section, Some of the notable works in GANs will be discussed

2.3.1 VGAN

Video Genarative Adversarial Network (VGAN) was developed by Carl Vondrick [8] which generates videos with its scene dynamics. This model could generate video upto a second at full frame rate. The model is trained on over preprocessed 2 million videos available on the internet and categorized into 4 categories: Golf course, hospital room, beaches, train station. In terms of architecture the model uses a standard Genereator discriminator architecture. The generated video is segmented into 2 feautres, foreground and background and the authors assumes the camera to be static which results in static background. The generator is developed into two streams for foreground and background respectively. The foreground stream uses 5 layers of 3d spatio temporal convolution layer (time x width x height) with up samples f from a low dimensional latent code z sampled from standard normal distribution. A masking layer m is added before the last layer which describe the pixels of object in foreground. The second stream for background generate uses a 5 layer strided 2d convolution layers (width x height) and generates a background b. 2D convolution layer is used as the background is assumed to be static. The Foreground and background are synthesised using the as per the equation

The video of 32 frames with the dimension of 64x64 is generated from a 100 dimensional latent code sampled from a normal distribution. The Discriminator is designed to solve 2 problems, it should classify realistic scene and to recognize plausible motion between frames. In terms of architecture the same 5 layered spatio-temporal convolutional layers used in foreground generation in generator is used as discriminator with the replacement of downsampling instead of upsampling and the last layer outputs a binary classification (real or not). Batch normalization and ReLU activation function is used after every layer in generator and discriminator. The model was trained using Adam optimizar and with a batch size of 64. The results show that the model can able to generate videos with sharp background and a blurry foreground. The Figure 7 shows the results of various generated videos. Though the resolution of foreground is blurry, the dynamics of the generated foreground is plausible and convincing and the user doesn’t have control over the content of generation.

2.3.2 RV- GAN

RV – GAN is a recurrurnt gan for unconditional video generation proposed by Sonam gupta et. al., This gan utilizes a special lstm variant called Trans conv LSTM by modulating the conventional ConvLSTM to have a transpose convolutional structure in input-to-state transitions. This enables the network to model both spatial and temporal relationships across layers simultaneously inside the TC-LSTM unit. Figure 5 shows the diagrammatic representation of Trans convolution LSTM. The composite model consist of a generator constructed by blocks of TC- LSTM units and 2 discriminators like Mcocgan [6]. The input to the generator is a noise sampled from a random distribution, is fed into generator at each time step. Additionally, hidden state and cell state from the previous time step is passed as an input accounting for the temporal information. At each time a frame generated which has strong co relation with previous frames. The discriminator is constructed using 2D and 3D convolution layers for image and video discriminator respectively. The entire network is trained on the combined loss of the discriminators.

The model is trained on Weizmann, UCF101, MUG datasets and the results are evaluated using Frechet Inception distance. owing to the recurrent structure, our method is able to generate high-quality videos, up to 2 times longer (32 frames) than training videos at inference time. The author proves that the propose GAN is generic and can be adapted to class conditional video generation. RV-GAN outperforms almost all other methods except V3GAN on UCF101. This might be because V3GAN generates background of the frame separately which is same throughout the video, which helps in enhancement of the metric.

2.3.2 ImaGINator

ImaGINator is a conditional GAN developed to generate videos of human faces with different expressions by Yaohui Wang et.al. The video generated is conditioned on class label denoting expressions Unlike VGAN [8], the generated video is segmented into spatial and temporal information as a hypothesis. In terms of construction the ImaGINator has a generator and 2 discriminator. The Generator is designed to be encoder decoder architecture , with skip connections, The Generator accepts a static image with a face of a person and encodes onto a latent vector p. The one hot encoded class label is concatenated into the latent vector. In this way the spatial and temporal information are embedded into the laten vector. In addition a random noise sampled from a standard normal distribution is concatenated onto the latent vector. The decoder is designed to be (1+2) D convolution layer, explicitly splitting temporal and spatial information. Moreover, it mirrors the encoder architecture and skip connections for feature maps are designed from every layer in encoder to its corresponding decoder layer. In addition, the embedded class label vector is also passed to every layer in decoder, which makes sure that every layer in decoder has embedded spatial and temporal information. The no of frames in the generated video is fixed and outputs one step at time.

Two Discriminator are used with each has its unique purpose, The discriminator Di takes the frames of the generated video and classifies it real or fake base on its apperance, whereas the Discriminator Dv takes the all the frames in order and the class label as input and classifies the dynamics in the frame real or fake. The combined loss function from both the discriminator and the reconstruction loss for corresponding frames are used to optimize the generator and each discriminator loss is used to optimize the corresponding discriminators. ADAM optimizer with same learning rate is used for all generators and discriminator trainings. Various Evaluation metrics like Frechet Inception Distance (FID), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure are used to evaluate the performance of generated videos and to monitor the training. Figure 8 illustrates the frames of the generated video. The Model is trained on various popular datasets like MUG Facial expression dataset, NATOPS aircraft handling signal dataset, Weizmann Action dataset, UvA -NEMO smile dataset are used for training and the results are compared. In terms of Image quality this method had outperformed the VGAN , Moreover the content of the generated video can be controlled using class labels which finds various application.

2.3.3 MoCoGAN

MoCOGAN is Video Generation GAN proposed by Sergey Tulyakov et. al. in 2016. This GAN architecture has segmented the video into Content and Motion and sample input from the Content Subspace and Motion subspace respectively. This unique architecture equip the Model to generate videos of a same content different action and same action by different content. This GAN is designed to generate videos of shorter duration but the duration of the video is need not to be fixed length. Considering the fact that the duration of the video is shorter, the subject in the video is designed to be remain the same. In terms of architecture, the model has 4 networks, a Recurrent Neural Network Rm, Generator G1, Image Discriminator Di, Video Discriminator Dv. The Generator generates images sequentially frame by frame. It takes latent image Z as input which contains zc, zm. The Random vector zc corresponds to content in the video is sampled from the standard normal distribution and it is used for all the times, as the subject in the video is constant. The latent vector Zc is generated by Rm recursively by sampling from standard normal distribution at every timestep. The Network generates a trajectory in the motion subspace which results in the motion of the subject in the video. The parameters of the Rm is learned during the training as not all the motion trajectory is actually possible. The Zc and Zm is fed into the generator which generates video frames. The Image discriminator evaluates the quality of images in each frame, whereas the video discriminator evaluates the motion of the subject in the video. The RNN network is trained solely on Video discriminator loss and the entire Generator network is trained on the combined loss of the discriminators. The Model is trained on Weizmann Database [15] and Tai-Chi database and the performance evaluation was summarized. The Average Content distance (ACD) is used for evaluation. Figure 9 shows the frames of video clips generated from then MoCoGAN model. It had outperformed the VGAN and TGAN in terms of quality of images in the video. Moreover it has the advantage of generating videos of variable length. However, the control over the content creation is missing as it is sampled randomly from the motion subspace.

2.3.4 pix2Pix GAN

Pix2Pix is an image translation techinique introduced in 2016 by Philip Isola et al [11] which uses Conditional GAN. Image translation is the method of transferring image in one domain to other. For example, translating grayscale image to colour images. In the paper [12], the author had used this method to translate satellite terrain images to google map style images. The conditional GAN has similar structure to Normal GAN, with a Generator and a discriminator. In terms of architecture, the generator is built using a encoder decoder architecture. The encoder and decoder are built using blocks, each block contain a convolution layer, Batch norm and ReLU activation function. The decoder part is mirrored version of encoder, where every layer in the decoder is connected with encoder through the skip connection referred to a U-Nets. This architecture prevents the loss of information from input to output layer when flowing only through a bottle neck [11]. The Generator is designed to take image from input domain and outputs image in target domain. The discriminator takes paired images from both the domain as input. It uses a standard convolutional layer with a final layer predicting weather the given image is real/fake. The discriminator segments the image in 70x70 patches and the outputs binary classification for each patch, this type of Gan is called PatchGAN. Reconstruction loss between the generated image and the target image is computed and the gradients of the generators are optimized to reduce this reconstruction loss. The composed model is trained on weighted sum of Discriminator loss and reconstruction loss. The model is trained on the satellite image to google map image dataset with 1097 images. Figure 10 illustrates the satellite images and its corresponding google map images and generated image after 10 epochs. This problem of image translation finds application in various fields. However, this method demands paired dataset which is difficult to obtain in various scenarios.

**CLEAN TEXT**

2. Literature Review

This chapter explores different works related to the topic. The initial section examines the existing state-of-the-art autonomous vehicle simulators. The second part provides a summary of previous methodologies used to define a comparative metric for evaluating autonomous vehicle simulators. The last part of this chapter discusses about the different types of Generative Adversarial networks (GANs), a type of Generative AI algorithm utilized in this thesis.

2.1 State-of-the-Art Simulators

Numerous simulators are available in the market, but this section analyse some relevant and popular simulators and highlighting their advantages, limitations and applications.

2.1.1 CARLA

CARLA, (Car Learning to Act) is an open-source software developed collaboratively by the Computer Vision Centre (CVC) and the Barcelona Supercomputing Centre (BSC) in partnership with the Toyota Research Institute. It is primarily designed for autonomous driving research and development, which features diverse and realistic environments, various climates and wide range of sensors. CARLA operates on a server-client architecture, built on Unreal Engine 4 and utilizing the OpenDRIVE standard 1.4 to define roads and urban settings. This unique structure allows the server to manage physics of simulator and computation while enabling user to communicate the server through C++ and Python APIs, providing scalability.

A notable feature of CARLA is its seamless support for developing, training, and validating machine learning algorithms. Researchers had experimented various algorithms like modular pipelines, imitation learning, and reinforcement learning within this simulator [2]. Leveraging Unreal Engine 4, CARLA offers high-quality, realistic rendering of environments. Figure 1 showcases scenes from the simulator in different weather conditions. Additionally, it supports an array of built in sensors such as camera, LiDAR and provides the user with various metadata, and ground truth. Moreover, CARLA offers access to diverse digital assets (actors, buildings) within the environment, meticulously designed to maintain a high level of realism [2]. However, it currently offers support for only two pre-defined urban maps covering 2.9 km and 1.4 km, which limits its diversity and generalization capabilities.

Figure1: Scenes from the CARLA simulator in different weather conditions.

2.1.2 LGVSL

LG Silicon Valley Lab (LGSVL) is an open-source simulation engine developed by LG Electronics. It utilizes the Unity gaming engine to render photorealistic environments and taking advantage of technologies like the High-Definition Render Pipeline (HDRP) from Unity.

This simulator is developed in two parts: the Simulation Engine and the User Autonomous Driving (AD) Stack [3]. The Simulation Engine, a customizable open-source platform, receiving its inputs from AD stack and simulate the environment, sensors, and vehicle dynamics. The AD Stack comprises three major layers: Perception, Planning, and Control, offering various user-configurable functionalities. The AD Stack and the Simulation Engine is connected through communication bridge interface, such as Cyber RT, ensuring seamless integration [3]. While the simulator comes with various default sensors including cameras, LiDAR, and Radar, its unique feature lies in its adaptability. Users can build and configure their own sensors and importing models of real-world sensors as plugins. For instance, the plugin for Velodyne VLP-16 LiDAR replicates point cloud generation similar to its actual counterpart [3]. These sensors' data and its mounting positions are defined through JSON-formatted text, simplifying their utilization. Figure 2 showcases the array of default sensors available within this simulator.

Figure 2: Different types of sensors. Left (top to bottom): Fish-eye camera,LiDAR, Radar; Right (top to bottom): Segmentation, Depth, 3D Bounding Box.

A distinguishing aspect of this simulator is its capability to incorporate real-world maps to construct virtual environments within the simulator. Map formats like Lanelet2, OpenDRIVE, and Apollo 2 HD map can be imported and used as the virtual environment. This features of LGSVL appeals to engineers of automakers and making it a highly suitable tool for their research.

2.1.3 SUMMIT

SUMMIT (The Simulator for Urban Driving in Massive Mixed Traffic) is an open-source simulator developed as an extension of the CARLA simulator [2], inheriting its physics and visual realism. Unlike many other simulators that predominantly which simulate rule-based traffic with minimal randomness, SUMMIT stands out for its ability to replicate the aggressive and chaotic nature of real-world traffic. This distinctive feature attracts users interested in training and testing algorithms for vehicles navigating unregulated traffic scenarios. SUMMIT employs the 'Context-GAMMA', a velocity-space optimization crowd behaviour algorithm [4] to simulate traffic behaviour geometrically and topologically. Additionally, it utilizes real-world maps from OpenStreetMap, extracting features such as roads, sidewalks, and roundabouts which are incorporated into the simulator, enabling the replication of real-world maps. An illustrative example can be seen in Figure 3, which showcases the comparison between a real map and its counterpart with unregulated traffic behaviour in SUMMIT at the Magic Roundabout in England.

Figure3: Scenes in the real world and corresponding scenes in SUMMIT

SUMMIT's utilization of real-world maps, combined with CARLA's visual realism and the simulation of chaotic traffic behaviour, significantly enhances its capabilities and making is a best option for inroad commercial autonomous vehicles research.

2.1.4 Gazebo

Gazebo stands as a prominent open-source simulator extensively utilized in the field of robotics and autonomous systems. Originating from Carnegie Mellon University in the early 2000s, its primary focus remains on mobile robots. What distinguishes Gazebo from other simulators is its ability to accurately emulate the physics of non-player elements within the environment.

This simulator integrates the Open Dynamics Engine [8], crafted by Russel Smith, a widely adopted physics engine within the open-source community. Tailored to replicate the dynamics and kinematics associated with articulated rigid bodies, this engine boasts a comprehensive range of features. These includes various joints, collision detection, mass and rotational functions, as well as a multitude of geometries, including arbitrary triangle meshes.

Gazebo's adaptability is another notable aspect, enabling the concurrent simulation of up to ten robots. This capability proves advantageous particularly in applications involving multiple robots, such as within the domain of autonomous driving. The inclusion of multiple robots, sensors, and actuators, collectively referred to as a model, is seamlessly facilitated through APIs, allowing easy integration into the simulator. Furthermore, Gazebo extends its functionality by facilitating the effortless addition of new 3D user-defined objects using OpenGL, a standard library for 3D object creation. Additionally, GLUT (OpenGL Utility Toolkit) offers an interactive user interface, enhancing its accessibility and usability.

While Gazebo excels in accurately simulating various physical phenomena, the challenge arises when attempting to replicate the extensive environments requisite for real-world autonomous driving scenarios. Despite this limitation, owing to its open-source nature, users actively engage in developing custom simulators tailored to their specific requirements such as Gazebo golfcart [9] which is developed for autonomous golf cart mobility within the golf field.

2.1.5 TORCS

TORCS, or The Open Racing Car Simulator, was developed in the early 2000s as an open-source platform for simulating car races. Its standout feature is its ability to adjust car settings, which is crucial in racing simulations. The engine uses a method called discrete-time simulation, where equations are processed in small steps of 0.002 seconds. In TORCS, participants are called "robots" and are loaded as separate parts of the system. This setup allows developers to create new smart agents that follow basic rules for robot code.

TORCS is good at mimicking real racing situations, including how cars move, track designs, weather changes, and interactive settings. This flexibility also makes it useful for academic purposes, letting researchers test artificial intelligence, machine learning, and self-driving algorithms.

However, there are challenges, like handling complex simulations and making the interactions between cars and the environment more realistic. TORCS benefits from a lively community of developers, researchers, and racing fans because it's open-source, letting people collaborate and improve the platform together.

2.2 Comparative Study on Simulators

This section examines notable studies that compare existing autonomous vehicle simulators and summarizes their approach and results.

In Guan Yang et al. work (2021), "Survey on Autonomous Vehicle Simulation Platforms," [5] the team extensively researched different autonomous vehicle simulation platforms. The objective of a simulator is broken down into five parts: Static environment simulation, Dynamic environment and behaviour simulation, Traffic flow simulation, Sensor simulation, and Vehicle dynamics simulation. They also established a taxonomy for existing simulators, categorizing them into Point Cloud-based and 3D Engine-based platforms. Point-based simulators, such as CarCraft from Waymo and Apollo from Baidu, reconstruct the environment based on simulated sensor data. Figure 4 displays the map from Apollo, a point-based simulation platform. On other hand, 3D engine-based platforms, like PanoSim, utilize gaming engines like Unity and Unreal to render environments following laws of physics (Figure 5).

Figure 4: Scene from Apollo simulator

Figure 5: Scene from PanoSim simulator

They further created a table comparing some of popular and relevant simulation platforms and their features [5] (Figure 6).

Figure 6: A Comparison table of various simulator

Although this table aids in comparing simulators roughly, it doesn’t compare sufficient features for making a concrete decision, and lacks a single metric defining the level of usability of the simulator for a user. While the categorization of simulators is provided, a clear comparative method among simulators is not clearly defined.

In Md Salman Ahmed et al.'s work (2016), an extensive study on connected vehicle simulators was presented [6]. The focus was on the domain of connected vehicles, including vehicle-to-vehicle and vehicle-to-server communication. The paper assessed several simulators based on their memory consumption, computing environment (Sequential or Parallel), and the number of vehicles they could handle. However, these results are specific to the connected vehicle domain and may not be applicable to other types of autonomous vehicle simulators effectively.

2.3 Generative Adversarial Networks

Generative adversarial networks (GAN) mark a significant advancement in the domain of Generative AI, first introduced in 2014 by Goodfellow et al. in the paper "Generative Adversarial Networks" [7]. Since then, GANs have gained substantial momentum in the field of Generative AI especially in image generation. This section discusses some noteworthy works within the domain of GANs.

2.3.1 VGAN (Video Generative Adversarial Network)

VGAN, developed by Carl Vondrick et al. [8], specializes in generating videos with its scene dynamics. The model is capable of generating videos up to a second at full frame rate. Its training involves over 2 million pre-processed videos sourced from the internet, categorized into four distinct groups: golf courses, hospital rooms, beaches, and train stations.

The architecture of VGAN employs a standard Generator-Discriminator structure. The generated video is segmented into two features: foreground and background, assuming a fixed camera resulting in a static background. The generator comprises two network streams dedicated to foreground and background, respectively. The foreground stream consists five layers of 3D spatiotemporal convolution layers (time x width x height), which upsamples the information from a low-dimensional latent code z, sampled from a standard normal distribution. A masking layer 'm' is introduced before the final layer, outlining the pixels of objects in the foreground. Meanwhile, the background stream utilizes five layers of 2D convolution layers (width x height), responsible for generating a background 'b'. The background stream uses 2D convolution layers as the background is assumed to be same for all the generated frames. The synthesis of foreground and background follows the equation

The resultant video, comprising 32 frames with dimensions of 64x64, is generated from a 100-dimensional latent code sampled from a normal distribution. The Discriminator is designed for two primary objectives: classifying realistic scenes and recognizing plausible and smooth motion between frames. It mirrors the architecture of the foreground stream of generator with a five-layered spatiotemporal convolutional setup, employing downsampling instead of upsampling. The final layer outputs a binary classification (real or not). Batch normalization and ReLU activation functions are used after every layer in both the generator and discriminator.

The VGAN is trained using Adam optimizer with a batch size of 64. Results demonstrate the model's ability to generate videos with a sharp background and a slightly blurry foreground. While the resolution of the foreground might be blurred, the motion of the generated foreground are convincing. However, the user had no control over the content of generation. Figure 7 illustrated the frames of various generated videos.

Figure 7: Videos generated using VGAN

2.3.2 RV-GAN

The RV-GAN, introduced by Sonam Gupta et al., [] stands for Recurrent Variational Generative Adversarial Network and serves the purpose of unconditional video generation. This particular GAN employs a specialized variant of LSTM known as TransConv LSTM. The innovation lies in modifying the conventional ConvLSTM by integrating a transpose convolutional structure into its input-to-state transitions. This unique design allows the network to simultaneously capture spatial and temporal relationships across layers within the TC-LSTM unit. (See Figure 5 for a diagrammatic representation of the TransConv LSTM [4].)

Figure: Diagrammatic representation of the TransConv LSTM. Ht−1,Ct−1,Ht,Ct are the hidden states and cell states at previous and current timestamps. Xt is input to the TC-LSTM unit.

The composite model comprises a generator built with blocks of TC-LSTM units and incorporates 2 discriminators similar to Mcocgan [6]. In the generator, noise sampled from a random distribution is fed at each time step, along with the inputs of the hidden state and cell state from the preceding time step to account for temporal information. This process generates frames that exhibit strong correlation with previous ones. On the other hand, the discriminator is structured using 2D and 3D convolution layers, specifically for image and video discrimination, respectively. The entire network is trained on the combined loss function derived from both discriminators.

Training of the model occurs on datasets such as Weizmann, UCF101, and MUG, with evaluation based on the Frechet Inception Distance metric (FID) [15]. Due to its recurrent structure, the proposed method demonstrates the ability to generate high-quality videos, extending up to 2 times the length (32 frames) of the training videos during inference. The authors substantiate the versatility of this GAN by showcasing its adaptability to class conditional video generation. RV-GAN exhibits superior performance compared to most other methods, falling short only when compared to V3GAN[] on UCF101.

2.3.2 ImaGINator

ImaGINator is a conditional Generative Adversarial Network (GAN) developed by Yaohui Wang et al [9]. Its primary aim is to produce human videos depicting various expressions. Unlike VGAN [8], this model generates videos conditioned on specific class labels for expressions for example smiling, jogging... The model is designed to segment the generated video into spatial and temporal parts.

ImaGINator comprises a generator and two discriminators. The generator adopts an encoder-decoder architecture with skip connections. It takes a static image featuring a person's face as input and encodes it into a latent vector 'p'. In addition, this vector is concatenated with one-hot encoded class label c, representing the expression, and random noise sampled from a standard normal distribution. This fusion embeds spatial (p) and temporal information (c) into the latent vector. The decoder, structured as a (1+2)D convolution layer, explicitly separates temporal and spatial information. It mirrors the dimensions of the encoder's architecture and had skip connections from encoder, ensuring that each decoder layer retains embedded spatial details from the corresponding encoder layer. Moreover, the embedded class label vector is integrated into every decoder layer, ensuring the preservation of temporal information throughout the model. The generated video consists of a fixed number of frames and the last layer of the generator outputs an image with all the frames.

Two discriminators serve distinct purposes: 'Di' evaluates individual frames of the generated video to classify real from fake based on appearance, while 'Dv' examines the sequence of frames alongside the class label to classify the dynamics within the frames as real or fake. The Generator is optimized on the combined loss function from both discriminators and a reconstruction loss for corresponding frames. Each discriminator's loss function optimizes the weights of corresponding discriminator. The ADAM optimizer with the same learning rate is used across all generator and discriminator components.

Evaluation metrics such as Frechet Inception Distance (FID), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure are used to evaluate the performance of the generated videos and to monitor the training progress. Figure 8 provides a visual representation of the frames within the generated video.

Figure 8: Frames of the video generated using ImaGINator

The model was trained on various widely used datasets like MUG Facial Expression Dataset [13], NATOPS Aircraft Handling Signal Dataset [14], Weizmann Action Dataset [15], and UvA-NEMO Smile Dataset [16]. Comparative analysis shows the superior image quality compared to VGAN. Furthermore, the model's ability to control the content of generated videos using class labels holds significant potential across various applications.

2.3.3 MoCoGAN

Motion and content decomposed GAN (MoCOGAN), a Video Generation GAN introduced by Sergey Tulyakov et al. in 2016 [10], operates on a unique architecture that segments videos into Content and Motion. This segmentation allows the model to sample inputs separately from the Content and Motion subspaces (unsupervised). Notably, this architecture enables the model to generate videos depicting the same content with different actions or the same action with different content. While designed for shorter video durations, MoCOGAN doesn't require a fixed length for the generated videos. Given the shorter duration, the video's subject is assumed to remain constant.

The model comprises four networks: a Recurrent Neural Network (Rm), Generator (G1), Image Discriminator (Di), and Video Discriminator (Dv). The Generator sequentially produces frames by taking a latent image Z as input, containing zc and zm. The random vector zc represents the video's content and is sampled from a content subspace, remaining constant throughout the video as the subject remains unchanged. On the other hand, the latent vector Zm, which determines the motion trajectory of the subject, and it is recursively outputted by Rm by sampling from a motion subspace at each timestep.

The Rm's parameters are learned during training, as not all motion trajectories are physically possible. The generator G takes Z (combination of Zc and Zm) and generates video frames sequentially. The Image Discriminator assesses frame quality, while the Video Discriminator evaluates subject motion in the video. The RNN is trained exclusively on the Video Discriminator loss, while the entire Generator network is trained on the combined loss of the discriminators.

The model's training utilizes the Weizmann Dataset [15] and Tai-Chi Dataset [17], and its performance was assessed using the Average Content Distance (ACD). Visual representations in Figure 9 depict video clip frames generated by the MoCoGAN model.

Figure 9: Frames of video generated by MoCoGAN

Notably, MoCoGAN had outperformed VGAN [8] and TGAN in image quality within the videos and offers the flexibility of generating videos of varying lengths. However, it lacks direct control over content creation like ImaGINator as content is randomly sampled from the content subspace

2.3.4 Pix2pix GAN

Pix2Pix, introduced in 2016 by Philip Isola et al [11], is an image translation technique using Conditional GANs. Image translation involves transforming images from one domain to another, such as converting grayscale images to colour images. In [12], the method was applied to translate satellite terrain images into Google Maps style images.

The structure of the conditional GAN resembles that of a Normal GAN, comprising a Generator and a Discriminator. In [12], the generator adopts an encoder-decoder architecture, where the encoder consisting of blocks that include a convolution layer, Batch Normalization, and ReLU activation function. The decoder mirrors the encoder, employing skip connections (known as U-Nets) between corresponding layers to retain information and prevent loss during transmission through bottle necks [11]. The Generator takes an image from the input domain and produces an image in the target domain. On the other hand, the Discriminator intakes the paired images from both domains, uses standard convolutional layers, converging in a final layer that determines the given image (real or fake). Using PatchGAN, the Discriminator segments images into 70x70 patches and performs binary classification for each patch, this makes the discriminator suitable for any given image size.

Training involves computing the reconstruction loss between the generated image and the target image. The generator's gradients are optimized to minimize this loss. The composed model training combines the Discriminator loss and the reconstruction loss using a weighted sum for gradient update. The dataset used for training, consisting of 1097 paired images of satellite and Google Maps images. Each image is pre-processed and rescaled to 256X256 pixels before training. Figure 10 illustrates the satellite images and its corresponding google map image and generated image.

Figure 10: The satellite images and its corresponding google map image and generated image after 10 epochs

While image translation using this method has diverse applications, this method often requires paired datasets, which is difficult to obtain in most of the cases.

List of figures

Figure 1: Scenes from the CARLA simulator in different weather conditions.

Figure 2: Different types of sensors. Left (top to bottom): Fish-eye camera,LiDAR, Radar; Right (top to bottom): Segmentation, Depth, 3D Bounding Box

Figure3: Scenes in the real world and corresponding scenes in SUMMIT

Figure 4: Scene from Apollo Simulator

Figure 5: Scene from PanoSIM simulator

Figure 6: A Comparison table of various simulator

Figure 7: Frames of the video generated using VGAN

Figure 8: Frames of the video generated using ImaGINator

Figure 9: Frames of the video generated using MoCoGAN

Figure 10: The satellite images and its corresponding google map image and generated image after 10 epochs

List of tables

References

[2]: Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. &amp; Koltun, V.. (2017). CARLA: An Open Urban Driving Simulator. <i>Proceedings of the 1st Annual Conference on Robot Learning</i>, in <i>Proceedings of Machine Learning Research</i> 78:1-16 Available from <https://proceedings.mlr.press/v78/dosovitskiy17a.html>.

[3]: G. Rong et al., "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 2020, pp. 1-6, doi: 10.1109/ITSC45102.2020.9294422.

[4]: Cai, Panpan, et al. "Summit: A simulator for urban driving in massive mixed traffic." 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020.

[5]: G. Yang et al., "Survey on Autonomous Vehicle Simulation Platforms," 2021 8th International Conference on Dependable Systems and Their Applications (DSA), Yinchuan, China, 2021, pp. 692-699, doi: 10.1109/DSA52907.2021.00100.

[6]: M. S. Ahmed, M. A. Hoque and P. Pfeiffer, "Comparative study of connected vehicle simulators," SoutheastCon 2016, Norfolk, VA, USA, 2016, pp. 1-7, doi: 10.1109/SECON.2016.7506701.

[7]: Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

[8]: Vondrick, Carl, Hamed Pirsiavash, and Antonio Torralba. "Generating videos with scene dynamics." Advances in neural information processing systems 29 (2016).

[9]: Wang, Yaohui, et al. "Imaginator: Conditional spatio-temporal gan for video generation." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2020.

[10]: Tulyakov, Sergey, et al. "Mocogan: Decomposing motion and content for video generation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[11]: Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[12]: Henry, Joyce, Terry Natalie, and Den Madsen. "Pix2Pix GAN for Image-to-Image Translation." Research Gate Publication (2021): 1-5.

[13]: N. Aifanti, C. Papachristou and A. Delopoulos, "The MUG facial expression database," 11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10, Desenzano del Garda, Italy, 2010, pp. 1-4.

[14]: Y. Song, D. Demirdjian and R. Davis, "Tracking body and hands for gesture recognition: NATOPS aircraft handling signals database," 2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG), Santa Barbara, CA, USA, 2011, pp. 500-506, doi: 10.1109/FG.2011.5771448.

[15]: Mauthner, Thomas, Peter M. Roth, and Horst Bischof. Action recognition from a small number of frames. na, 2009.

[16]: M. Taskirar, M. Killioglu, N. Kahraman and C. E. Erdem, "Face Recognition Using Dynamic Features Extracted from Smile Videos," 2019 IEEE International Symposium on Innovations in Intelligent SysTems and Applications (INISTA), Sofia, Bulgaria, 2019, pp. 1-6, doi: 10.1109/INISTA.2019.8778400.

[17]: Sun, Shan, et al. "Taichi: A fine-grained action recognition dataset." Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval. 2017.