Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Применение генеративно-состязательной нейросети для моделирования дорожной среды / Application of Generative Adversarial Networks to road environment simulation

Обучающийся / Student Хассам Мухаммад
Факультет/институт/кластер/ Faculty/Institute/Cluster центр химической инженерии
Группа/Group B42621c
Направление подготовки/ Subject area 12.04.04 Биотехнические системы и технологии
Образовательная программа / Educational program Биоинженерия и биотехнические системы 2020
Язык реализации ОП / Language of the educational program Русский, Английский
Статус ОП / Status of educational program СОП, МОП
Квалификация/ Degree level Магистр
Руководитель ВКР/ Thesis supervisor Успенская Майя Валерьевна, профессор, доктор технических наук, Университет ИТМО, центр химической инженерии, профессор (квалификационная категория "ведущий профессор")
Консультант не из ИТМО / Third-party consultant Nikekhin Aleksei, Continental AG, Engineer, PhD, PhD

Обучающийся/Student	Документ подписан Хассам Мухаммад 06.06.2022		Хассам Мухаммад
	(эл. подпись/ signature)		(Фамилия И.О./ name and surname)
Руководитель ВКР/ Thesis supervisor	Документ подписан Успенская Майя Валерьевна 06.06.2022		Успенская Майя Валерьевна
	(эл. подпись/ signature)		(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

АННОТАЦИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ SUMMARY OF A GRADUATION THESIS

Обучающийся / Student Хассам Мухаммад
Факультет/институт/кластер/ Faculty/Institute/Cluster центр химической инженерии
Группа/Group B42621c
Направление подготовки/ Subject area 12.04.04 Биотехнические системы и технологии
Образовательная программа / Educational program Биоинженерия и биотехнические системы 2020
Язык реализации ОП / Language of the educational program Русский, Английский
Статус ОП / Status of educational program СОП, МОП
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Применение генеративно-состязательной нейросети для моделирования дорожной среды / Application of Generative Adversarial Networks to road environment simulation
Руководитель ВКР/ Thesis supervisor Успенская Майя Валерьевна, профессор, доктор технических наук, Университет ИТМО, центр химической инженерии, профессор (квалификационная категория "ведущий профессор")
Консультант не из ИТМО / Third-party consultant Nikekhin Aleksei, Continental AG, Engineer, PhD, PhD

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ DESCRIPTION OF THE GRADUATION THESIS

Цель исследования / Research goal
Training and testing the object detection algorithms using real data can be very time consuming and expensive. Our goal is to solve this problem by using GAN models to generate road environment simulations, and then use the simulations to test object detection algorithms.
Задачи, решаемые в ВКР / Research tasks
Provide a comprehensive review of deep learning and theoretical background of Generative Adversarial Networks. Generation of road environment simulations using several GAN models and analyzing the results. Perform object detection on the generated simulations. Evaluation of the results.
Краткая характеристика полученных результатов / Short summary of results/findings
We successfully generate road environment simulations using Pix2Pix and Vid2Vid GANs. The generated simulations are tested on object detection algorithms such as Yolov3. We observe that object detection algorithms successfully detect the objects in the GAN generated simulations.

Обучающийся/Student	Документ подписан	

Хассам Мухаммад
06.06.2022

(эл. подпись/ signature)

Хассам Мухаммад

(Фамилия И.О./ name and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан
Успенская Майя Валерьевна
06.06.2022

(эл. подпись/ signature)

Успенская Майя Валерьевна

(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ / OBJECTIVES FOR A GRADUATION THESIS

Обучающийся / Student Хассам Мухаммад
Факультет/институт/кластер/ Faculty/Institute/Cluster центр химической инженерии
Группа/Group B42621c
Направление подготовки/ Subject area 12.04.04 Биотехнические системы и технологии
Образовательная программа / Educational program Биоинженерия и биотехнические системы 2020
Язык реализации ОП / Language of the educational program Русский, Английский
Статус ОП / Status of educational program СОП, МОП
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Применение генеративно-состязательной нейросети для моделирования дорожной среды / Application of Generative Adversarial Networks to road environment simulation
Руководитель ВКР/ Thesis supervisor Успенская Майя Валерьевна, профессор, доктор технических наук, Университет ИТМО, центр химической инженерии, профессор (квалификационная категория "ведущий профессор")
Консультант не из ИТМО / Third-party consultant Nikekhin Aleksei, Continental AG, Engineer, PhD, PhD

Основные вопросы, подлежащие разработке / Key issues to be analyzed
Terms of reference:
Research on generative adversarial networks(GAN), Translation of semantic map to RGB image using GAN, use of GANs for road environment simulation, selection of appropriate model for road environment simulation.
Initial Data:
Cityscape images and their semantic segmentation.
Content of work:
Developing an understanding of deep learning, analysis of the GANs, comparison of different GAN architectures. Understanding the suitability of GANs for video translation and road environment simulation.
Key issues to be Analyzed:
Analyze GANs and provide justification for their use in simulation of road environment. Selection and justification of appropriate GAN architecture for road environment simulation.

Дата выдачи задания / Assignment issued on: 10.11.2021

Срок представления готовой ВКР / Deadline for final edition of the thesis 01.06.2022

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not
Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/ Thesis supervisor

Документ подписан
Успенская Майя Валерьевна
16.05.2022

(эл. подпись)

Успенская Майя Валерьевна

Задание принял к исполнению/ Objectives assumed BY

Документ подписан
Хассам Мухаммад
30.05.2022

(эл. подпись)

Хассам Мухаммад

Руководитель ОП/ Head of educational program

Документ подписан
Успенская Майя Валерьевна
30.05.2022

(эл. подпись)

Успенская Майя Валерьевна

Table of Contents

1 Introduction

1.1 Motivation

Object detection is crucial for self-driving technology. Until a vehicle can accurately detect the objects on the road, it cannot be expected to drive safely. A lot of sensors have been developed that scan the environment and detect the objects on the road. One of these sensors are cameras. Video signals from these cameras are used by the object detection algorithms to detect the objects on the road. These algorithms are machine learning models that are pre trained on a large amount of visual data.

Acquiring large amounts of camera videos to train the object detection algorithms can be time consuming and expensive. The data can also be biased if it is gathered from a few locations or from similar environments. This is why a lot of effort is being spent to generate high quality synthetic data that can be used to train the object detection algorithms. One way to produce synthetic visual data is by using GANs (Generative Adversarial networks)[14].

The aim of this thesis is to produce synthetic videos using various GAN models and to test their efficacy in training and testing of object detection algorithms.

1.2 Purpose and Research Question

In this thesis, different types of GAN models are used to generate videos. The videos are generated using semantic label maps as input. We test the performance of object detection algorithms on these synthetic videos. We test the quality of these models by analyzing their output. We also try to analyze the advantages and disadvantages of different models.

1.3 Approach and Methodology

This thesis focuses on producing synthetic videos for training and testing the object detection algorithms. There are three main problems that we face in this task. First is that there are different kinds of models that can be used to produce synthetic videos, we have to analyze and select the appropriate model. Second is the selection of training data for these models. The models have to produce synthetic road environment data and that is why appropriate road environment training data has to be selected to train these models. Third problem that we face is the selection of appropriate object detection algorithms that can be trained and/or tested using the synthetic data that our model generates.

2 Fundamentals of Machine Learning

2.1 Machine Learning and AI

Artificial intelligence can be defined as the automation of human performed intellectual tasks. Artificial intelligence does not necessarily have to involve machine learning, as some tasks can be automated by coding human defined rules in if-else form.

Machine learning systems are subsets of AI systems in which the system learns to perform the task using training examples. The system is not explicitly programmed. Deep learning is one of the many ways to train the machine learning systems.

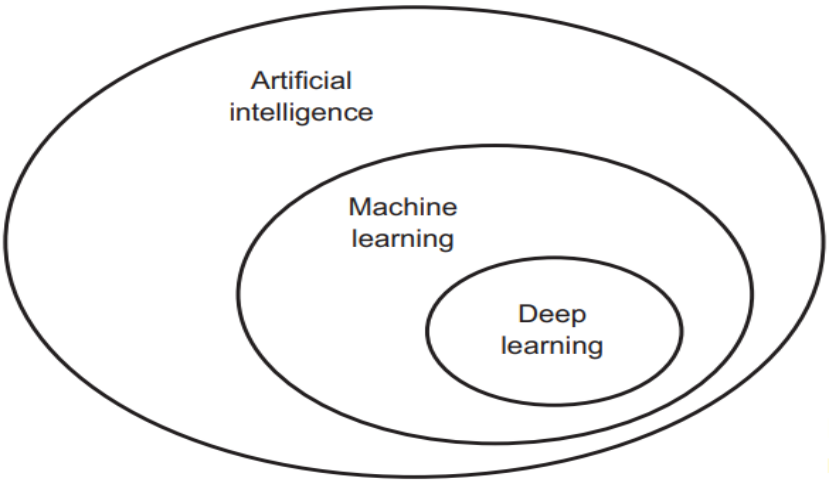

Figure 1. Artificial intelligence subsets

A Machine learning model uses a lot of data about the task it has to perform, this data is called training data. The model is then trained to perform the task using the training data. In this manner, the rules of the task are learned by the model itself.

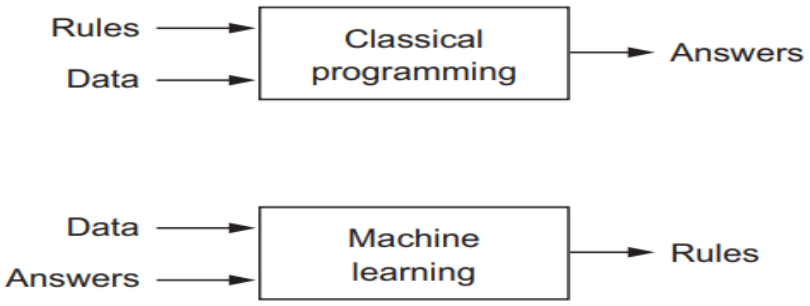

Figure 2. Classical programming vs Machine learning

2.2 Branches of machine learning

Machine learning models can be separated into four different categories.

2.2.1 Supervised Learning

Supervised learning deals with mapping input data to an output. Output can be discrete or continuous. Mapping is learned through labeled input-output pairs known as training data. Most common supervised learning problems are classification and regression. These problems can be solved by various algorithms such as Linear regression, Logistic regression and Decision trees.

2.2.2 Unsupervised Learning

Unsupervised learning is concerned with finding interesting and useful patterns in the input data. Most common unsupervised learning algorithms are Principal Component Analysis and K-means clustering.

2.2.3 Self Supervised Learning

Self supervised learning learns from unlabeled training data. It is in the middle of unsupervised and supervised learning. Most common instances of self supervised learning are autoencoders and generative adversarial networks.

2.2.4 Reinforcement learning

In reinforcement learning, an agent takes actions in the environment and gets some reward in return. In this manner, the agent gets information about the environment and learns to choose actions that will maximize the reward. Reinforcement learning is becoming more popular now.

2.3 Deep Learning

Deep learning is the machine learning method that has become very famous and has seen a wide array of applications in numerous fields. Deep learning methods can be supervised, unsupervised or self supervised.

Deep learning models have more than one processing layer that can learn to represent the data with multiple abstraction levels. The layers learn to represent the data by changing their parameters according to the observed data. The parameters are changed through the technique of backpropagation. Deep learning has been very successful in solving many artificial intelligence problems. It has provided breakthroughs in computer vision problems such as object detection, image generation, image enhancement. It has also been successfully applied in signal processing problems such as audio detection and audio generation. In addition to that, it has been applied in biology for prediction of drug molecule activity, in physics for analysis of particle accelerator data[24].

Traditional machine learning algorithms had a limited ability to process raw data. They required a lot of preprocessing and very complex feature engineering to make data usable by machine learning algorithms. A lot of times, the feature engineering is very unintuitive and time consuming. Deep learning requires less feature engineering. They solve the problem of feature engineering by representing the data at a higher level of abstraction. The higher level of abstraction is achieved by having more than one computational layer[24].

Deep learning has been very successful in learning structures in high dimensional data, Due to this reason it is being successfully applied in the domains of physical science, social science and economics and finance. This thesis will provide the theoretical and mathematical description of deep learning

models. It will also provide detail on various deep learning models used for image generation.

3 Neural Networks

Neural networks are an application of stochastic gradient for classification and regression with potentially very rich hypothesis classes. There are many variations of neural networks, the most basic are feed forward neural networks with back propagation. The working of these networks will be discussed here.

3.1 Basic element

The basic element of a neural network is a neuron. It is a nonlinear function of an input vector $x \in R^m$ to a scalar output $a \in R$. It is parametrized by a vector of weights $w \in R^m$ and an offset $w_0 \in R$. For the function to be nonlinear we also specify an activation function $f: R \rightarrow R$. The function should be differentiable.

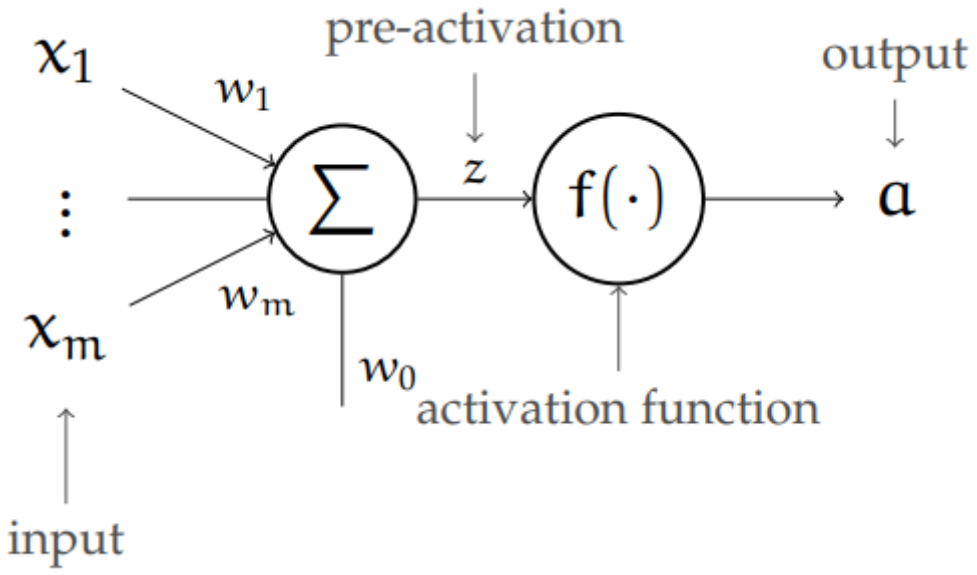

Figure 3. Neuron

The function computed by the neuron can be written as:

$$a = f(z) = f\left(\sum_{j=1}^{m} x_j w_j + w_0\right) = f(w^T x + w_0) \qquad (1)$$

To train a neuron, we need a loss function $L(guess, actual)$ and a dataset $\{(x^1, y^1), ..., ... (x^i, y^i)\}$. In training, we do stochastic gradient descent on the weights to minimize

$$J(w, w_0) = \sum L(NN(x^i; w, w_0), y^i) \qquad (2)$$

Where NN is the output of the neural network.

3.2 Networks

Multiple neurons are put together into a network. A neural network takes input $x \in R^m$ and generates an output $a \in R^n$. ; the inputs of each neuron might be elements of x and/or outputs of other neurons. The outputs are generated by n output units.

There are many types of neural networks. Some of the most commonly used are:

1. Feed-Forward Neural Networks
2. Convolutional Neural Networks
3. Recurrent Neural Networks

We will consider the first two as they are used in GANS

4 Types of neural networks

4.1 Feed-Forward Neural Networks

In a feed-forward network, the network can be thought of as defining a function-call graph that is acyclic: that is, the input to a neuron can never depend on that neuron's output. Data flows, one way, from the inputs to the outputs, and the function computed by the network is just a composition of the functions computed by the individual neurons.

In software packages, the graph is organized in layers. A layer is a group of neurons that are in parallel. They take the output of the previous layer as input and their outputs are inputs for the next layer.

4.1.1 Single Layer

A layer is a group of neurons that are in parallel and are not connected to each other. In a fully connected layer, each neuron in the layer takes the same input. Input to the layer is $x \in R^m$ and output is $a \in R^n$.

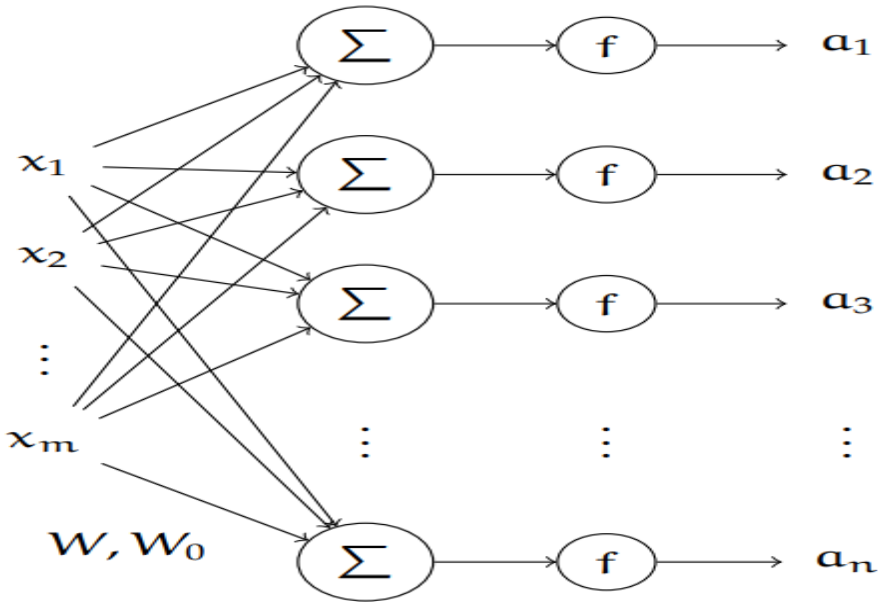

Figure 4. Layer of neurons

Each unit in the layer has a vector of weights and an offset. Since the layer consists of many units, its weights can be considered as a matrix W. All the offset can be considered as a vector W_0. If we have m inputs, n units, n outputs, then

- W is a $m \times n$ matrix.
- W_0 is a $n \times 1$ vector.
- The input X is a $m \times 1$ vector.
- The preactivation $Z = W^T X + W_0$ is a $n \times 1$ vector.
- A the activation is a $n \times 1$ vector.

Mathematically, the layer operation can be written as:

$$A = f(Z) = f\left(W^T X + W_0\right) \qquad (3)$$

4.1.2 Multiple Layers

A neural network usually consists of more than one layer, where output of the previous layer is fed to the next layer.

Let l be the name of the layer, m^l be the input and n^l be the output of the layer. $W^l$and W_0^l are the weights and biases of the layer. Then the shape of the weight matrix and the bias vector is $m^l \times n^l$ and $n^l \times 1$ respectively. The preactivation output Z^l is a $n^l \times 1$ vector and can mathematically written as

$$Z^l = W^{l^T} A^{l-1} + W_0^l \qquad (4)$$

The activations A^l is a $n^l \times 1$ vector.

$$A^{l} = f^{L}(Z^{l}) \qquad (5)$$

The schematic for the network is shown below.

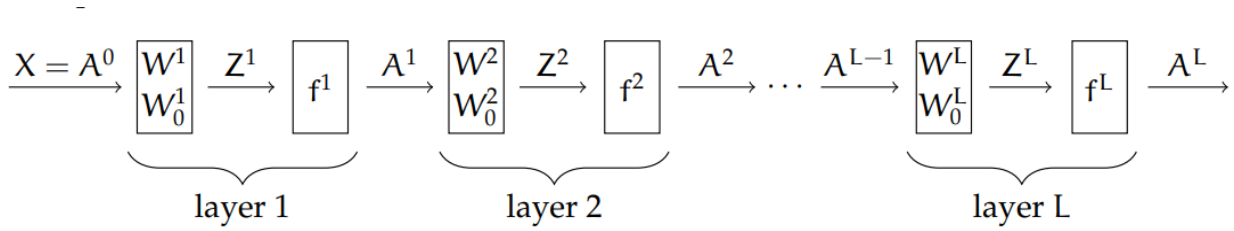

Figure 5. Neural network

4.1.3 Activation Function

Activation functions introduce nonlinearity to our network. It allows generalization and better adaptation to data. A lot of activation functions have been developed along the time. They are chosen based on the problem we want to solve [20].

Some of the activation functions are:

Step function

$$step(z) = \{0\ if\ z < 0,\ 1\ if\ z \geq 0$$

Rectified Linear unit

$$ReLU = \{0\ if\ z < 0\ ,\ z\ if\ z \geq 0$$

SoftMax function

Takes the output $Z \in R^{n}$ from the whole layer as input and produces output as a vector $A \in [0, 1]^{n}$. The output has the property that $\sum_{i=1}^{n} A_i = 1$. It can be interpreted as a probability distribution over n items.

$$softmax(z) = exp\left(z_1\right) / \sum_{i=1}^{n} exp(z_i), .. \ .. \ , exp\left(z_n\right) / \sum_{i=1}^{n} exp(z_i)$$

4.1.4 Loss function

To train a neural network, we need a loss function $L(guess, actual)$ and a dataset $\left\{\left(x^1, y^1\right) ..... \left(x^i, y^i\right)\right\}$. The output of the last layer is the guess. It is fed into the loss function together with the real output and loss is calculated. Loss is a measure of distance between actual output and guessed output. This loss is used to train the network through gradient descent.

Some of the loss functions are:

- Squared loss
- Hinge loss
- Negative Log Likelihood

4.1.5 Gradient Descent

Gradient descent is a mathematical optimization technique that is used to calculate the point at which the output of the function is the lowest. In neural networks, it is used to minimize the loss with respect to the weights.

Negative of the gradient calculated at any specific point in the function domain represents the direction of steepest descent . This is the idea behind gradient descent, where a small step is taken in the direction of the negative of the gradient to reduce the loss. The process is repeated until the algorithm finds a local minima[27].

Stochastic gradient descent is the main form of gradient descent used in optimization of neural networks. In this technique, few data points are used to compute the gradient of the loss with respect to the weights.

Let θ be a vector of weights and then the weights can be updated as:

$$\theta_{t+1} = \theta_t - \lambda \cdot \nabla_{\theta_t} L\left(\left(f_{\theta_t}(x_i), y_i\right)\right) \qquad (6)$$

Convergence is not guaranteed but a local minima can be reached with stochastic gradient descent[27].

4.1.6 Error back-propagation

Neural networks are trained through gradient descent. Gradient of the loss function with respect to the weights is computed and then the weights are updated through gradient descent. It is also possible to perform a gradient descent over the complete training dataset in a batch gradient descent. In stochastic gradient descent (SGD), gradient is taken over a single data point at a time.

Only the contribution of one data point to the gradient of the loss with respect to the weights will be discussed here. To do SGD for a training example (x, y), we need to compute $\nabla WLoss(NN(x;\ W),\ y)$, where W represents all weights W^l and bias W_0^l in all the layers $l\ =\ (1, \ldots,\ L)$. Chain rule is used to calculate the gradient for all the layers.

Let us see how the loss depends on the weights of the final layer W^L. Let $loss\ =\ Loss(NN(x;\ W),\ y)\ =\ Loss(A^L, y)$ $and\ A^L = f^L(z^L)$ and $Z^L = W^{L^T} A^{L-1}$.

By chain rule:

$$\frac{\partial loss}{\partial W^L} = \frac{\partial loss}{\partial A^L} \cdot \frac{\partial A^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial W^L}$$

For dimensions to match in matrix multiplication, the above equation can be written as

$$\frac{\partial loss}{\partial W^L} = A^{L-1} \cdot \frac{\partial loss}{\partial Z^L}$$

where

$$A^{L-1} = \frac{\partial Z^L}{\partial W^L}$$

By applying the chain rule repeatedly, we can get the equation for the gradient of the loss with respect to pre-activation of all the layers upto first layer.

The gradient of the loss with respect to pre-activation of the first layer can be written as

$$\frac{\partial loss}{\partial A^L} \cdot \frac{\partial A^L}{\partial Z^L} \cdot \frac{\partial Z^L}{\partial A^{L-1}} \cdot \frac{\partial A^{L-1}}{\partial Z^{L-1}} \cdots \frac{\partial A^2}{\partial Z^2} \cdot \frac{\partial Z^2}{\partial A^1} \cdot \frac{\partial A^1}{\partial Z^1} \qquad (7)$$

Schematically it can be shown as:

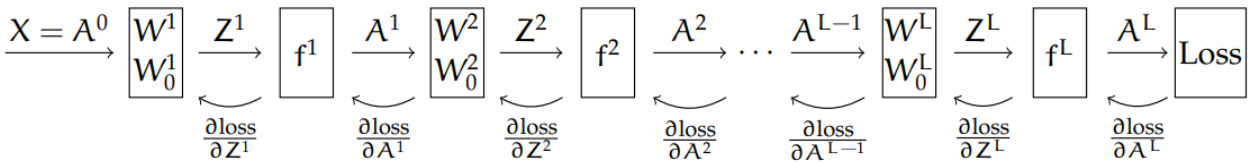

Figure 6. Error back propagation

This process of updating the weights through gradient descent and chain rule is called Error Back-propagation.

4.1.7 Training

During training, a random data point is selected from the data set. The data is propagated through the network and output is calculated. The generated output

is compared to the actual output and loss is calculated. This loss is used to update the weights through gradient descent and Error-back propagation.

4.2 Convolutional neural networks

Convolutional neural networks are used if there are patterns in the data. The most important application domain of convolutional neural networks is signal processing. Signals might be spatial (two-dimensional camera image) or temporal (speech or music). If we are dealing with a signal processing problem, we can take advantage of stationary properties of data.

4.2.1 Example

Imagine that we have a problem designing and training a neural network that can take an image as input and output a classification which is positive if the image contains a cat and negative otherwise.

There are two important properties of the problem that we can take into consideration

1. Spatial locality: The set of pixels we will need to take into consideration to find a cat will be near one another in the image.
2. Translational invariance: The pattern of pixels that characterizes a cat is the same no matter where in the image the cat occurs.

Convolutional networks take advantage of these properties of the problem.

4.2.2 Filters

An image filter is a function that takes in a local spatial neighborhood of pixel values and detects the presence of some pattern in that data [21].

Let us consider a simple scenario in which we have a 1-dimensional binary image and a filter F of size two. The filter is a vector of two numbers which will

be moved along the image, taking a dot product between the pixel value and filter value at each step, and combining the outputs to produce a new image.

Let X be the original image, of size d; then pixel i of the output image is specified by:

$$Y_i = F.(X_{i-1}, X_i) \qquad (8)$$

4.2.3 Tensors

More than one filters can be applied to the data, if there are k such filters, then the result is k new images, which are called channels. The new images can be stacked to form a cube of data, indexed by the original row and column indices of the image, as well as by the channel. The set of filters that can be applied to this cube of data are three dimensional. They will be applied to a sub-range of the row and column indices of the image and to all of the channels. These blocks of data are called tensors. Tensors are useful for extracting features from a signal [21].

4.3 Recurrent Neural Network

Recurrent neural networks(RNNs) are used for sequential inputs such as sentences. They successively process input from the input sequence, taking single input at a time. In their hidden units they contain a state vector. This vector remembers all the information from previous inputs to the network[24].

Following is the basic RNN element.

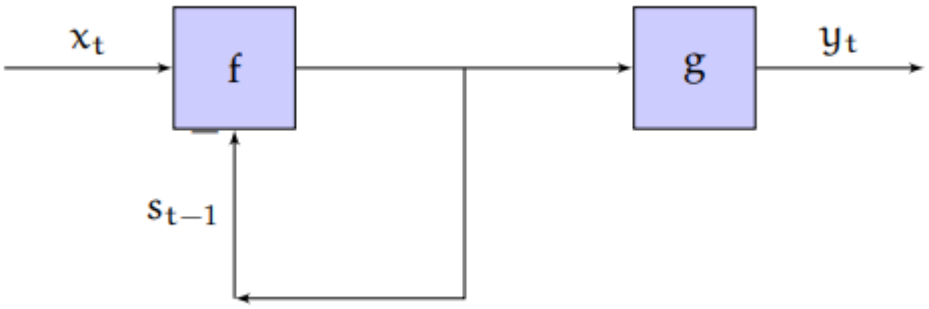

Figure 7. Recurrent neural network

where

x_t is an input at time t.

$s_t = f(s_{t-1}, x_t)$ is a state vector at time t-1.

$y_t = g(s_t)$ is an output at time t.

5 Latent Spaces

Latent space is representation of compressed data. In this space, images that are similar will be closer together and images that are different will be farther apart. Latent means “hidden”. It is hidden because it is hard to visualize any space beyond 3-dimensions[7].

5.1 Data compression

Encoding information using fewer bits is called data compression. In deep learning, data is compressed to learn the important features of data.

5.2 Feature learning

Suppose we have a deep learning classification model that is using convolutional neural networks. Learning for this model means to learn the features of images at each layer and assigning a combination of features to a specific output.

In this learning process, the dimensions of the input image are first decreased and then increased. The dimensionality reduction is called encoding and expansion is called decoding. Because the model has to reconstruct the compressed data, it learns to encode only the important features. This compressed data is the latent space representation[7].

5.3 Latent spaces in GANs

In Generative adversarial networks, Latent space contains compressed representation of data in such a way that the generator can convert a point from latent space into an image. New images can be generated by interpolating on latent space.

6 Deep Generative models

6.1 Introduction

Generative models generate new data by observing the instances in training data. The generated data can be in the form of images, videos,text and speech. These models are able to generate the data by determining the probability distribution over the observed data[29][30]. The new data can be generated by sampling from this probability distribution. Learning for generative models means adjusting the parameters of the models so that the distance between the model distribution and real data distribution is minimized[29].

Some of the commonly used generative models are:

- Variational Autoencoders
- Normalizing flow models
- Generative adversarial networks

6.2 Normalizing flow models

Normalizing flow model is a generative model that consists of a sequence of invertible functions[31]. This model directly approximates the probability distribution of the data. The model achieves this by minimizing the negative log likelihood of real data distribution from model distribution.

Normalizing flow has following advantages

- Training is more stable compared to generative adversarial networks.
- Convergence of the distributions is easier.

It has following disadvantages

- Low quality of the generated samples
- Latent space has high dimensions. This makes interpretation difficult.

6.3 Variational Autoencoders

An autoencoder is an unsupervised machine learning algorithm that consists of an encoder and a decoder. The encoder learns to compress the real data to latent space, and the decoder takes the point in the latent space as input and learns to recreate the original data. After training, the decoder is usually discarded and the encoder is used for data compression.

Variational autoencoder is a type of autoencoder used for data generation[29]. In this model, the encoder learns to encode the input to a normal distribution in latent space. The decoder takes a point from this distribution and generates the data. After training, the decoder can be used to generate data by sampling over the latent space.

6.4 Generative Adversarial Networks

Generative Adversarial networks is a type of generative models introduced by Ian Goodfellow in 2014[22]. Due to the very high quality of the generated data, GANs have become very popular. Our research focuses on GANs and more detail on them will be given in next sections.

7 Image to image translation

A lot of problems in computer vision can be thought of as translating an input image into an output image. The concept is similar to language translation. Just as a sentence can be translated from English to Russian, an image can be translated from RGB image to semantic label map. Image to image translation can be defined as the task of translating one plausible representation of a scene into another, provided enough data.

Traditionally, each of the image-to-image translation tasks has been solved by a special-purpose code, despite the fact that the problem is always the same, predicting pixels from pixels. Convolutional neural networks (CNNs) are being used to provide a common framework for image-to-image translation problems. CNNs learn to minimize a loss function, a function that outputs the quality of the result. Although the learning is based on gradient descent and is automatic, a lot of human effort goes into designing an effective loss function.

An obvious loss function will be calculating Euclidean distance between the predicted and ground truth pixel and then minimizing that loss function through gradient descent. This approach produces blurry results, this is because Euclidean distance is minimized by averaging over all possible outputs. Creating loss functions that can force CNNs to output sharp, realistic images is an unsolved problem.

Generative adversarial networks can be used to automatically learn a loss function appropriate for image-to-image translation. GAN learns a loss that tries to classify if the output image is real or fake, while simultaneously training a generative model to minimize this loss. Because GANs learn a loss that adapts to the data, they can be applied to a multitude of tasks that traditionally would require very different kinds of loss functions.

7.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are deep learning methods in which two neural networks contest with each other. The two networks are: generative which produces data, and discriminative which classifies the result.

The generative model tries to learn latent space of the data in such a way that it can be used to generate data that can pass for real data. Discriminative models classify produced data as real or fake for a specific domain. GANs have been applied to various domains such as time series synthesis, natural language processing and computer vision. In these domains, GANs can learn to mimic the distribution of data[25].

The first GAN model was published in 2014 by Ian Goodfellow[22]. It was an unsupervised learning algorithm in which the generator tried to learn the distribution of the data. It learns this distribution by mapping a noise vector z to data space. The discriminator D learns by assigning the correct label to real instance and generated instance.

After the seminal work of Ian Good fellow, various different GAN models have been developed. Some of them improve the quality of generated data and some of them adapt and design the GANs for specific problems. Today GANs can provide solutions for problems such as image editing, image generation, transferring style and enhancing the resolution[26].

Most significant impact of GANs has been in the area of computer vision, where they have been used for image synthesis, image to image translation, image super resolution and image completion. In this report, we will discuss the theory of Neural networks, Convolutional neural networks, how they can be used to build GANs and application of GANs in computer vision with focus on Image-to-Image translation.

7.2 Conditional Generative Adversarial networks

GANs are generative models that takes input in form of a noise vector z and maps it to an output image y, $G: z \rightarrow y$. The conditional GANs learn a mapping from observed image x and noise vector z, to output y, $G: \{x, z\} \rightarrow y$. This means that output of the generator is conditional on input image.

7.3 Translation with Pix2Pix GANs

In image-to-image translation with GANs, the generator is trained to produce images that cannot be distinguished from real images by the discriminator. The discriminator is trained at distinguishing generated images from real images.

Conditional GANs are suitable for this task because the output of the generator is conditional on the input image. This was first introduced with a neural network known as Pix2Pix GAN which operates in a conditional setting. In these settings, the generator takes the input image and generates a translated image. The discriminator takes generated translation and real translation of images as inputs and tries to distinguish fake from the real. It does so by returning probability, a number between 0 and 1, where 1 means that image is certainly authentic and 0 means that image is certainly generated.

As the discriminator gets better at distinguishing, it forces the generator to generate images that are more similar to real images. And as the generator gets better at generating images, it forces the discriminator to become better at distinguishing. Both networks are trained concurrently, in this adversarial manner, they are trying to optimize different and opposite loss functions.

7.4 Loss Function

The loss function of the conditional generative adversarial network can be defined as

$$L_{cGAN}(G, D) = E_{x,y}[log\ D(x, y)\ ] + E_{x,z}[log(1\ -\ D(x, G(x, z))]$$

The generator tries to minimize this function and the discriminator tries to maximize it. Mixing the GAN loss with Euclidean distance can improve the results. This way, the generator is tasked to produce a result that can fool the discriminator and is also near the ground truth output is Euclidean distance. This loss is added in the form of L1 distance as L1 distance produces less blurring.

$$L_{L1}(G) = E_{x,y,z}[||y\ -\ G(x, z)||]$$

The final objective is

$$G^{*} = argL_{cGAN}(G, D) + \dot{\lambda}L_{L1}(G) \qquad (9)$$

The details of loss functions are given below.

7.5 Generator Loss

Generator is tasked with not only fooling the discriminator but also generating an image that is near the ground truth output. This is why the generator has two loss functions, generator loss and L1 loss.

- The generator loss is a sigmoid cross-entropy loss of the generated images and an array of ones. This loss function trains the generator to produce images that can fool the discriminator.
- L1 loss is the absolute distance error between the original image and generated image. This function trains the generator to produce images that are near the ground truth.
- Both of these loss functions allow the generated image to be similar to the target image.
- Total loss is $gan_{loss} + lambda\ *\ L1_{loss}$.

The generator is trained with the following procedure.

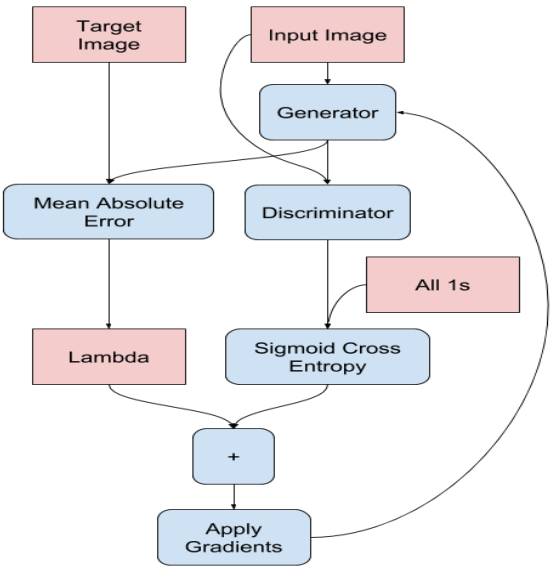

Figure 8. Generator schematics

7.6 Discriminator loss

The loss function for discriminator takes two inputs: real images and generated images and calculates two losses, real loss and generated losses.

- The discriminator should discriminate real images by assigning them probability 1. That is why real loss is cross entropy loss of real images and an array of 1s.
- The discriminator should discriminate generated images by assigning them probability 0. That is why generated loss is cross entropy loss of generated images and an array of 0s.
- Total loss is the sum of real loss and generated loss.

The procedure for training the discriminator is shown below.

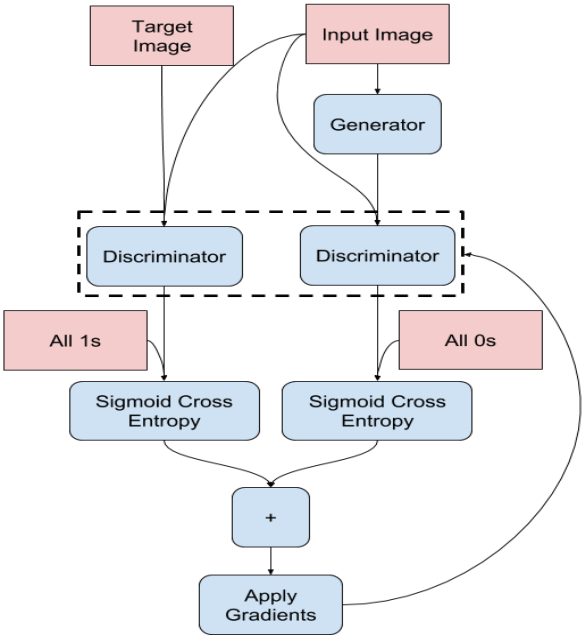

Figure 9. Discriminator schematic

7.7 Pix2Pix Conditional GAN Architecture

The generator consists of an encoder and a decoder, and the Discriminator consists of an encoder.

Each module in the encoder is

$$convolution \rightarrow Batch\ normalization \rightarrow Leaky\ RELU$$

Each module in the decoder is

$$transposed\ convolution \rightarrow Batch\ normalization \rightarrow Dropout \rightarrow RELU$$

Each image is first fed into the generator where the encoder compresses the image and creates a latent space representation of the image. The decoder takes the compressed data as input and tries to generate a representation in the target domain. The output of the decoder is fed to the Discriminator. It is also used for calculating the mean absolute error between the generated image and the target image.

The Discriminator produces two outputs. First with the generated image and the input image as input, and second with the target image and input image as input. The discriminator consists of an encoder. The two images in each input are concatenated and compressed by the encoder. The outputs of this encoder are used to calculate the Discriminator loss and the Generator loss as defined above.

The architecture of conditional GAN has the following general form

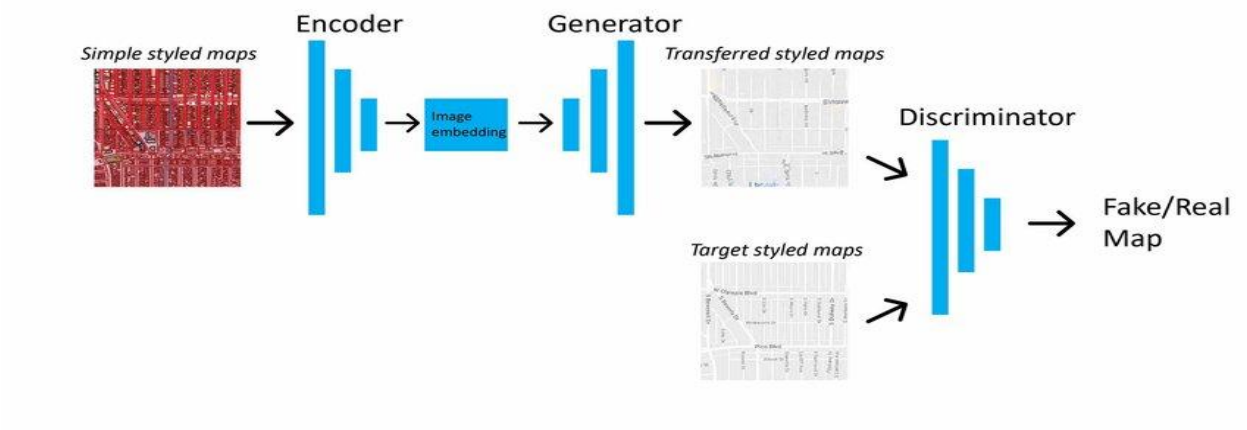

Figure 10. C-GAN architecture

7.8 CycleGAN

For image translation with Pix2Pix GANs, it is necessary to have paired input and output images. This is because the loss is calculated between the generated output image and the real output image of the same input image. However, for a lot of tasks we do not have a paired training set of input and output images. Cycle GAN learns the mapping between the source domain X and the target domain Y in the absence of the paired training set[4].

The goal of the CycleGAN is to learn the mapping $G: X \rightarrow Y$ such that distribution of the images generated by the generator G is same as the distribution Y. However in CycleGAN, the training data is not paired, this means that the generated image will not be an accurate translation of the input image. To produce an accurate translation, we have to introduce a new loss function. This is achieved by introducing cycle consistency loss in addition to the losses already defined in the Pix2Pix GAN section[4].

7.8.1 Cycle consistency loss

Cycle consistency means that the resulting image is close to the input image. For example, If a sentence is translated from russian to spanish and then translated back to russian, the final translation should be the same as input. In this way, the cycle consistency loss ensures that the translation is accurate.

In Cycle GAN we have:

- Generator $G: X \rightarrow Y$ produces image $\widehat{Y}$ by taking input image X
- Generator $F: Y \rightarrow X$ produces image $\widehat{X}$ by input input image $\widehat{Y}$.
- L1 loss is calculated between the images X and $\widehat{X}$.

Mathematically the loss function can be written as:

$$L_{cyc(G,F)} = E_{x \sim p_{data}(x)}[|F(G(x)) - x|] + E_{y \sim p_{data}(y)}[|G(F(y)) - y|] \quad (10)$$

Schematically it can be shown as:

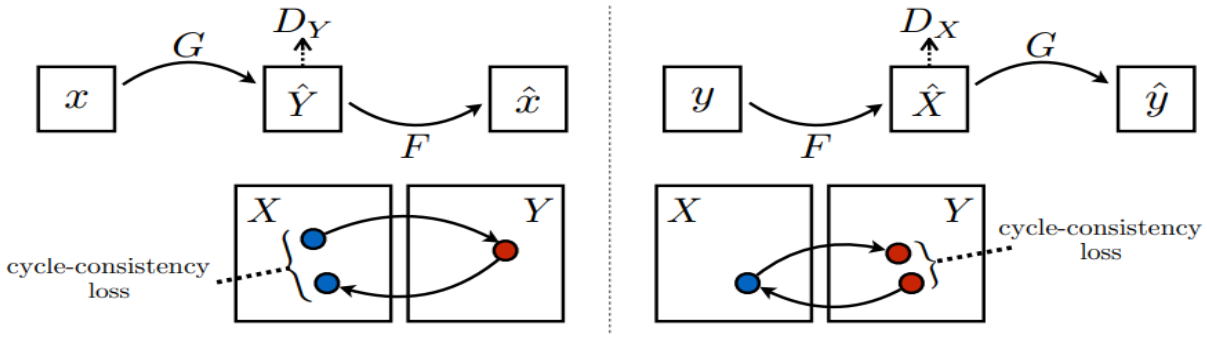

Figure 11. Cycle GAN loss

8 Video translation

Like image translation, we can also translate videos from one domain to another using GANs. Synthesizing artificial visual experience is an important topic in artificial intelligence. It has application in many domains such as computer vision, robotics and computer graphics. An input video can be mapped into an output video in a similar manner to image mapping. A video can be decomposed into frames and these frames can then be used to produce to output images. The output images are then combined to produce a video.

The main problem in video to video translation is temporal consistency. A usual image to image translation GAN does not take temporal dynamics into account. Each generated image is independent of the previously generated image. Because of this, a sequence of generated frames of a video is temporally incoherent. Any Conditional GAN model that translates videos should take time dependency of consecutive frames into account and produce temporally coherent videos.

8.1 Introduction

The problem of video translation can be thought of as a distribution matching problem. The goal is to train the model such that the conditional distribution of the generated videos given the input videos matches the conditional distribution of real videos. This problem is solved in vid2vid GAN. It is a Conditional GAN model that learns from paired input and output videos. This model takes time dependency of consecutive frames into account and produces a temporally coherent video.

8.2 Video to Video translation

Let $s_1^T \equiv \{s_1, s_2, \ldots, s_T\}$ be a sequence of source video frames. For example, it can be a sequence of semantic segmentation masks or edge maps. Let

$x_1^T \equiv \{x_1, x_2, ..., x_T\}$ be the sequence of corresponding real video frames. The goal of video-to-video synthesis is to learn a mapping function that can convert s_1^T to a sequence of output video frames, $\widehat{x}_1^T \equiv \{\widehat{x_1}, \widehat{x_2}, ..., \widehat{x_T}\}$, so that the conditional distribution of $\widehat{x_1^T}$ given s_1^T is identical to the conditional distribution of x_1^T given s_1^T.

$$p(\widehat{x_1^T} | s_1^T) = p(x_1^T | s_1^T) \qquad (11)$$

By decreasing the distance between these distributions the model learns to generate photorealistic translation of input video.

The model used for the conditional distribution matching task is a Conditional Generative adversarial network. Let G be a generator that takes a source image sequence as input and generates an output frame sequence $x_1^T = G(s_1^T)$. Generator is trained by solving an optimization function:

$$\max_D \min_G E_{(x_1^T, s_1^T)}[\log D(x_1^T, s_1^T)] + E_{s_1^T}[\log(1 - D(G(s_1^T), s_1^T))] \qquad (12)$$

Solving this equation minimizes the Jensen-Shannon divergence between $p(\widehat{x_1^T} | s_1^T)$ and $p(x_1^T | s_1^T)$. This optimization function is similar to the optimization function for image translation GAN. The difference is that it takes a sequence of frames as input, rather than a single frame. Following sections will provide the details of the network and the optimization function.

8.3 Sequential Generator

Conditional distribution of generated sequence $p(\widehat{x_1^T}|s_1^T)$ can be given in the form of product

$$p(\widehat{x_1^T}|s_1^T) = \prod_{t=1}^{T} p(\widehat{x_t}|\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t}). \qquad (13)$$

These means that frames can be generated sequentially and the generation of the t-th frame depends on three factors:

1. current source frame s_t.
2. Past L source frames , s_{t-L}^{t-1}.
3. Past L generated frames $\widehat{x_{t-L}^{t-1}}$.

The feed forward neural network F models the conditional distribution of $\widehat{x_t}$ using $\widehat{x_t} = F(\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t})$. The sequence of frames is generated by applying F recursively.

Video signals contain a large amount of redundant information in consecutive frames. If the optical flow between consecutive frames is known, we can estimate the next frame by warping the current frame. This estimation would be largely correct except for the occluded areas. Based on this observation F is modeled as:

$$F(\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t}) = (1 - m_t) \odot \widehat{w}_{t-1}(\widehat{x_{t-1}}) + m_t \odot h_t$$

- $\widehat{w}_{t-1} = W(\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t})$ is the estimated optical flow from $\widehat{x}_{t-1}$ to $\widehat{x}_t$.

- $\widehat{h_t} = H(\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t})$ is the hallucinated image generated by the generator H
- $m_t = M(\widehat{x_{t-L}^{t-1}}, s_{t-L}^{t})$ is the occlusion mask with continuous values between 0 and 1.

W, H and M are residual networks.

8.4 Conditional Image Discriminator

The purpose of D_I is to ensure that each output frame resembles a real image given the same source image. This conditional discriminator should output 1 for a true pair (x_t, s_t) and 0 for a fake one $(\widehat{x_t}, s_t)$.

8.5 Conditional Video Discriminator

The purpose of D_V is to ensure that consecutive output frames resemble the temporal dynamics of a real video given the same optical flow. While D_I conditions on the source image, D_V conditions on the flow. Let w_{t-k}^{t-2} be K − 1 optical flow for the K consecutive real images x_{t-K}^{t-1}. This conditional discriminator D_V should output 1 for a true pair $(x_{t-1}^{t-K}, w_{t-2}^{t-K})$ and 0 for a fake one $(\widehat{x}_{t-1}^{t-K}, w_{t-2}^{t-K})$. For the sake of this discussion, two sampling operators are defined. Let Φ_I be a random image sampling operator such that $\Phi_I(x_1^T, s_1^T) = (x_i, s_i)$ where i is an integer uniformly sampled from 1 to T. Let Φ_V be a sampling operator that randomly selects K consecutive frames. It is

defined as $\Phi_V(w_1^{T-1}, x_1^T, s_1^T) = (w_{i-K}^{i-2}, x_{i-K}^{i-1}, s_{i-K}^{i-1})$ where i is an integer uniformly sampled from K + 1 to T + 1.

8.6 Learning Objective Function

The video generating function F is trained by solving

$$\min_F \left(\max_{D_I} L_I(F, D_I) + \max_{D_V} L_V(F, D_V) + \lambda_W L_W(F) \right) \quad (14)$$

L_I is the GAN loss on images defined by conditional image discriminator D_I. L_V is the GAN loss on K consecutive frames defined by D_V. L_W is the flow estimation loss.

Image conditional GAN loss L_I is defined by sampling operator Φ_I:

$$E_{\Phi_I(x_1^T, s_1^T)}[\log D_I(x_i, s_i)] + E_{\Phi_I(\widehat{x}_1^T, s_1^T)}[\log(1 - D_I(\widehat{x}_i, s_i))] \quad (15)$$

The video GAN loss L_V is defined by:

$$E_{\Phi_V(w_1^{T-1}, x_1^T, s_1^T)}[\log D_V(w_{i-K}^{i-2}, x_{i-K}^{i-1})] + E_{\Phi_V(w_1^{T-1}, \widehat{x}_1^T, s_1^T)}[\log (1 - D_V(\widehat{x}_{i-K}^{i-1}, w_{i-K}^{i-2}))] \quad (16)$$

The flow loss L_W includes two terms. The first is the endpoint error between the ground truth and the estimated flow, and the second is the warping loss when the flow warps the previous frame to the next frame. Let w_t be the ground truth flow from x_t to x_{t+1}. The flow loss L_W is given by:

$$L_W = \frac{1}{T-1} \sum_{t=1}^{T-1} \left(\left| \widehat{w}_t - w_t \right| - \left| \widehat{w}_t(x_t) - x_{t+1} \right| \right) \quad (17)$$

9 Object Detection Algorithms

Object detection means detecting samples of visual objects of specified classes. As the purpose of the research is to test object detection algorithms using synthetic data, it is pertinent to mention it and some of the object detection algorithms.

Object detection is a very important task in computer vision. It has a lot of applications, such as detecting faces in pictures, surveillance, autonomous driving and classification. Object detection algorithms can be divided into two categories[15].

- Traditional machine learning with feature engineering
- Deep Learning

Before the advent of Deep learning, traditional machine learning algorithms were used for object detection algorithms. These algorithms depended on a lot of feature engineering and customization. With the introduction of R-CNN in 2014, deep learning algorithms have become the main tools for object detection. Deep learning algorithms are not as much dependent on feature engineering as traditional machine learning algorithms. They also show better performance on large datasets. Today, deep learning is the most used algorithm for object detection[16]. Deep learning based object detection algorithms can be divided into two categories

- Two stage object detection algorithms. This category includes R-CNN, Fast R-CNN, Faster R-CNN, G-RCNN
- One stage object detection algorithms. This category includes Yolo, SSD, RetinaNet.

These two categories will be discussed here briefly

9.1 Two stage object detection algorithms

As the name suggests, two stage object detectors consist of two stages. First stage is the generation and extraction of proposed regions. The proposed regions can be thought of as bounding boxes. Second stage is the extraction of the features and classification of each proposed region.

Some of the most used two stage object detection algorithms are:

- R-CNN
- Fast R-CNN
- Faster R-CNN

9.1.1 R-CNN

The R-CNN model was proposed in 2014. It consists of three modules.

- Region proposal module
- Feature extractor module
- Classification module

Region proposal module generates the regions to be classified. These regions are then passed through the convolutional neural network based feature extractor. The features are then classified by a linear classifier such as Support Vector Machines[17].

The problem with this approach is that 2000 regions are proposed per image at test time, this makes the object detection task very slow. This problem is solved in Faster R-CNN by using a trainable region proposal network.

9.1.2 Faster R-CNN

Faster R-CNN was proposed in 2016. Its architecture differed from that of R-CNN, in that it had a trainable region proposal network. By making the

feature extractor a part of the training process, it makes the object detection task more efficient.

9.2 One stage detectors

Detectors that can generate bounding boxes over the images without a region proposal step are called one stage detectors. These detectors are faster and for that reason, more suitable for real time detection[16]. Here we discuss following one stage detectors:

- YOLO
- YOLO v2

9.2.1 YOLO

The YOLO(You Only Look Once) model was introduced in 2015. This model contains a single neural network that takes an image as input and predicts bounding regions and labels every region. This model has a lower accuracy but it is considerably faster than two stage detectors. It can operate at 45 frames per second to 155 frames per second.

9.2.2 YOLO v2

This model is an improvement over YOLO. Some of the changes that were made are the use of batch normalization and high resolution pictures.

10 Object Detection with synthetic data

10.1 Benefits of using Synthetic data

Artificially generated videos and images can be used to train the object detection algorithms. The synthetic data can come from many sources. CAD software can be used to generate data. Gaming engines are also being used to simulate the environment for creating synthetic data. While the synthetic data may not be as detailed or real as the original data, it holds several advantages over real data which can have these problems [12][13].

- Collecting enough samples of real data to train the object detection algorithms can be time consuming. The data has to be collected from different sources and obtaining permissions for using the data can be difficult. The real data also has to be preproprocessed for training. The preprocessing is done manually and can be very time consuming.
- Obtaining quality real data for training is very expensive.
- The data can be biased. If the data is not obtained from all kinds of scenarios or all the scenarios are not covered then there can be a bias in the data. Sometimes, extreme examples are not present in data, which can introduce a bias in the data. In training object detection algorithms for self-driving cars, one challenge is to train the algorithms on situations that occur very rarely, the data on these situations is usually not available.

The synthetic data solves the above mentioned problems due to its following properies.

- Generating synthetic data is not as time consuming. No preprocessing is needed, which saves a lot of time.
- Obtaining synthetic data is very cheap. If we have a generating model, we can generate as many examples as we want.

- Model can be directed to generate extreme examples, this can eliminate the bias in the training set.

10.2 GANs for generating synthetic data

One way to generate synthetic data is by using Generative adversarial networks. Using GANs to generate synthetic images has following advantages[14]:

- Generating data with GANs is very easy. Infinite amount of data can be generated by sampling from the latent space.
- The data generated by GANs is very realistic and sometimes it is indistinguishable from real data. The current GAN models such as Pix2Pixhd and vid2vid produce very realistic data. This makes it a very good model for training object classification algorithms.
- Data for extreme examples, rare examples that are not present in real dataset, can be generated easily. This property can make GANs a good choice to augment real data for training object detection algorithms for autonomous driving.

11 Road Environment simulation using GANS

This chapter will discuss how we generated synthetic data using GANs for testing object detection algorithms. Our task was accomplished in the following three steps.

1. Analyzing different GAN algorithms for simulating road environment
2. Gathering video data and simulating road environment
3. Analyzing the simulation

As has been discussed in the previous section, we have analyzed Pix2Pix and Vid2Vid for simulating road environments. Now we will discuss how we simulated the road environment using these models and what results we obtained.

11.1 Simulating road environment using Pix2Pix

First model that we used to simulate a road environment was Pix2Pix. This is the earliest GAN image translation model and has a simpler architecture. We use the cityscape dataset to train the Pix2pix GAN. The cityscapes dataset is obtained from the site http://efrosgans.eecs.berkeley.edu/pix2pix/datasets/cityscapes.tar.gz. The model learns to translate the semantic label map to a real representation.

After training, the result looks like this:

Figure 12. Training result

We also trained and tested the model using grayscale images. The following result was obtained:

Figure 13. Grayscale training result

For the purpose of simulating the road environment, we use consecutive frames from two semantic segmented videos. The first video was obtained from https://youtu.be/S2VIP0qumas. The second video was obtained from https://www.cityscapes-dataset.com/. The first video originally had instance level segmentation. We did some image processing to get semantic segmentation. The output frames were then combined together to form a video.

An image from first simulation

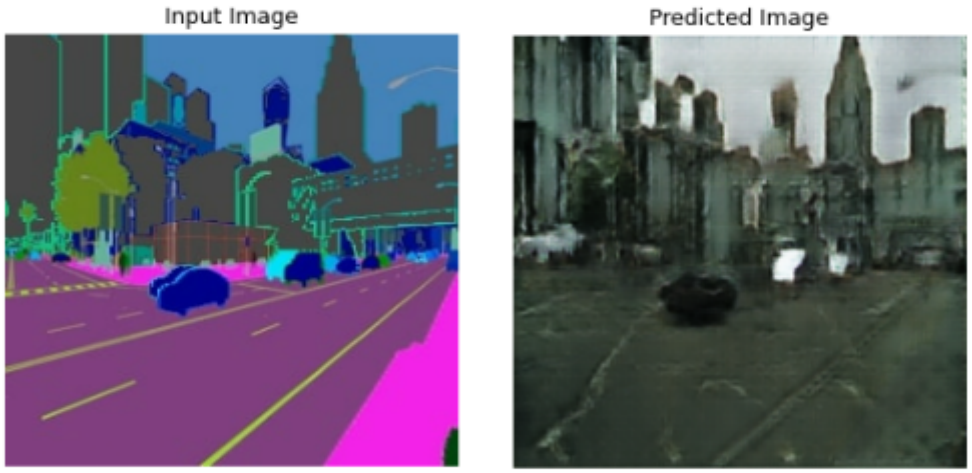

Figure 15. Output frame for first video

Complete output simulation for the first video can be viewed here.

An image from the second video

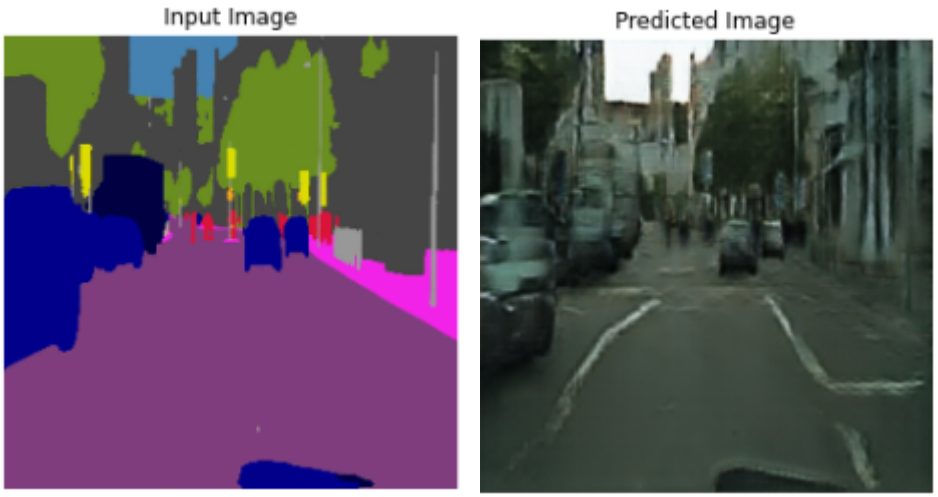

Figure 17. Output frame for second video

Complete output simulation for the second video can be viewed here.

11.1.1 Issues with Pix2Pix

There are two issues with Pix2Pix are

- Pix2Pix is a relatively old model and has low resolution of 256*256. This resolution is not enough to create high quality road environment simulation.
- Second issue that is specific to video simulation is that frames are generated independantly. In video generation, this can cause problems as the transition from one frame to the next is not smooth.

Both of the above mentioned issues can be solved by Vid2Vid.

11.2 Simulating road environment using vid2vid

We also simulated the road environment using vid2vid. Semantic segmented frames were obtained from https://www.cityscapes-dataset.com/. Obtained real output was then converted to a video.

Example frame from output video

Figure 18. Vid2Vid output frame

Complete output simulation for the vid2vid video can be viewed here.

Benefits of using vid2vid model

- The vid2vid model is a newer model and it generates output with higher resolution.
- The vid2vid is made specifically to generate videos. It takes information from the previous frames to generate the next frame. In this manner, images transition smoothly and a better and smoother video is produced.

12 Evaluation techniques for GAN output

In GANs, the generator model through the discriminator and both models are together. Both models learn to establish an equilibrium, that is why convergence is not guaranteed and there is no objective loss function. Because of that, there is no way to objectively analyze the performance during training .

Different techniques have been developed that can assess the performance qualitatively and quantitatively[18]. Some of the ways to assess the GAN performance are:

- Manual Generator output evaluation
- Qualitative generator output evaluation
- Quantitative generator output evaluation

12.1 Manual Evaluation

A lot of machine learning developers do the manual inspection of the generated images to assess the quality of the generator. This process involves creation of a batch of synthetic images and then judging the quality and variety of the generated images with respect to the target space.

Manual inspection is the simplest method to evaluate the model but it has some limitations

- It is subjective and therefore can be biased
- It requires the evaluator to know the difference between what is real and what is not.
- Time constraint is high as only few images can be evaluated manually at a time

12.2 Qualitative Evaluation

In this type of evaluation, human evaluators are asked to judge images as real or fake. A lot of human judges are used for this purpose. They are shown different pairs of fake and real images from the target domain, and are asked to select the preferred image. A rating is calculated based on how many times the generated images are preferred.

Downsides of this approach are:

- This approach is labor extensive
- Performance of human beings at detecting fake images can improve over time.

12.3 Quantitative Evaluation

Different techniques have been developed to give quantitative evaluation of the generated images. Some of these techniques are

- Inception Score
- Modified Inception Score
- Frechet Inception score

These methods usually include using a pretrained neural network to extract the features from the images and use those features to distinguish real data and fake data.

12.3.1 Frechet inception distance

Frechet inception distance estimates the distribution by measuring quantities like mean, variance and covariance of various features in the both image sets. The features are extracted through a pre-trained Inception v3 model[28]. After that it calculates the distance between distributions in both sets.

13 Result

As has been discussed above, the objective evaluation of the GAN output is very difficult. For our purposes, we use the object detector Yolov3 to detect the objects in the generated images. The model is available here [19]. Results are shown below.

13.1 Pix2Pix Result

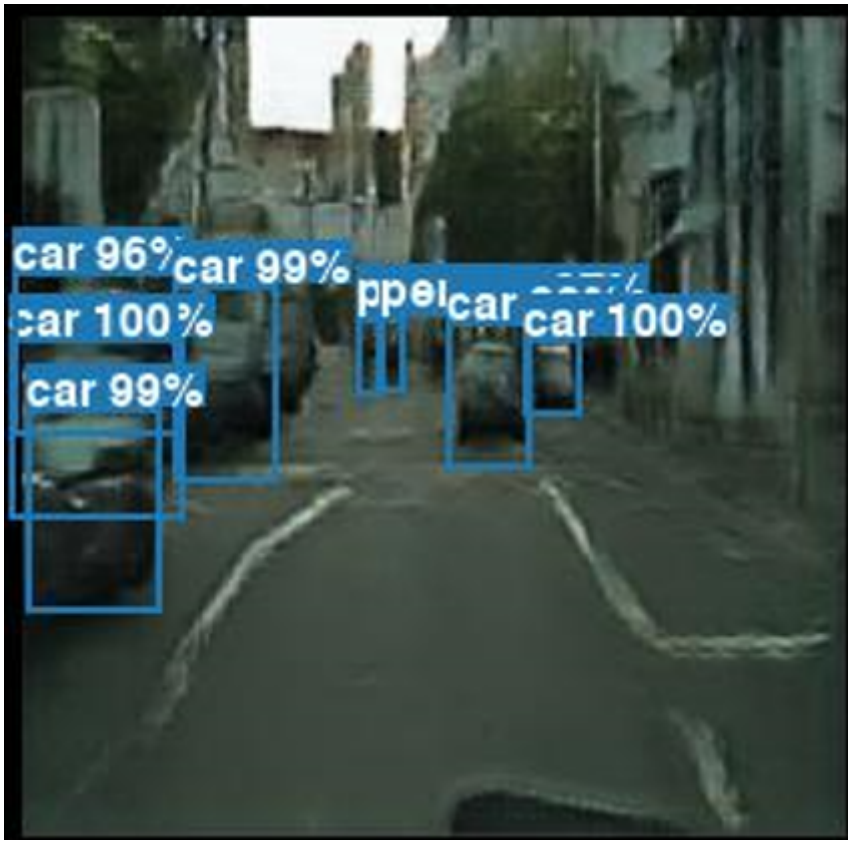

Figure 19. Object detection for Pix2Pix output

Complete video output can be seen here.

The above result shows that even though the quality of the pix2pix output is not very good, the object detection algorithm is still able to find objects in the output.

13.2 Vid2Vid result

We perform object detection on Vid2Vid output frames and the real frames. The object detection is performed by the YOLOv3 model. We also analyze and compare the results.

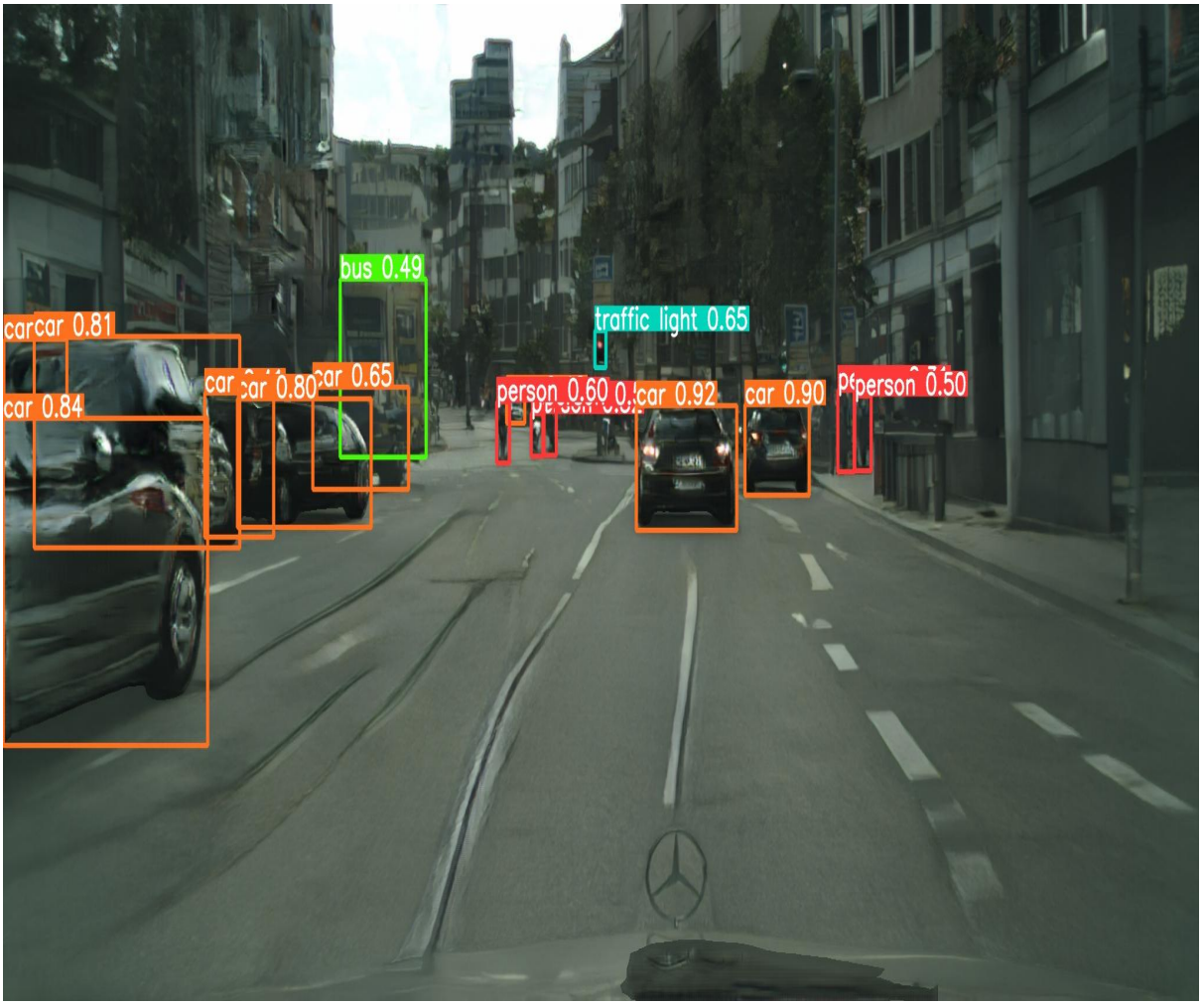

Figure 20. Object detection for Vid2Vid output

Complete video output can be seen here.

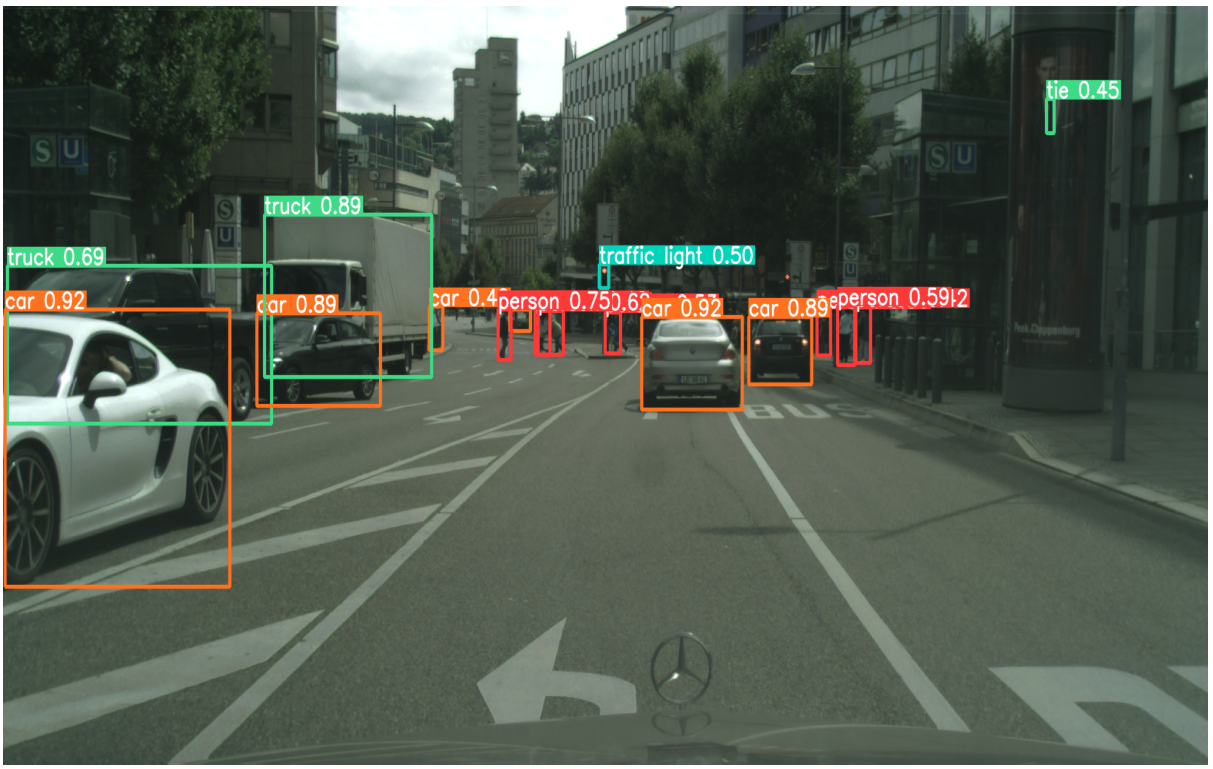

Figure 21. Object detection for real output

Complete output video can be seen here.

The comparison for the above two frames is illustrated in the following table.

Object	Number of detections for real frame	Detections for vid2vid generated frames
cars	6	7
trucks	2	0
people	7	5
traffic light	1	1
tie	1	0
Bus	0	1

Table 1. YOLO detection comparison

From the above results, it can be seen that the YOLO object detector is able to detect the objects such as people, cars and street lights in the Vid2Vid output. Even though the result is good, it is not perfect. A single truck in the semantic map was generated as two separate cars, and was classified as such.

14 Conclusion

Assisted and autonomous driving functions in modern personal and cargo vehicles are becoming a deciding feature for marketing success. Competition for available solutions is extremely high, driving functionality of products nearly to the same level.

Deep learning-based solutions need an extremely big amount of data for learning and testing. In addition to that, data needs to be collected from different environments to decrease the bias in the model. Such data is traditionally collected during test drives i.e. drives are taken in different countries with different drive sides, road markup, traffic signs, prevalent road vehicles, temporary construction markup e.t.c.. This technique makes data gathering very expensive.

To decrease the bias in the model, variety in the variables such as day time, weather condition, road lighting technology is required. Additionally important are specific rare scenarios to be tested (very particular examples: traffic signs or pedestrians drawn on moving vehicles, accidental pedestrians appearing in front of vehicles). Due to these limitations of real data, use of synthetic data is unavoidable.

This work offers an approach to create such synthetic data using cGAN and provides proof of concept. The results show the possibility to generate video based on semantically mapped templates which are output of available scenario generation tools already available in industry. Training data used is day-light, dry cloudy weather, American city, right-hand traffic (consequently). Generated video was tested on an open object detection algorithm to ensure that it is suitable for testing. Results show that objects are correctly detected at the

correct position. On the limited amount of test data few missing or false detection were noticed.

14.1 Further research

The positive results allows further development in following main directions:

- Improve visual quality of video and implement quantitative metric to measure the quality (FID).
- Test on different types of segmentation/detection algorithms providing metrics to ensure comparable quality with real video.
- Apply generated video for segmentation/detection algorithm learning.
- Develop longer road environment simulations using GANs
- Achieved results shows that GAN approach in synthesizing video for learning/testing of assisted and autonomous driving functions is feasible and useful. Generative models used in current work are thoroughly learned and can be used in other areas out of scope of this paper.

References

1. Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125-1134. 2017.
2. (pix2pix: Image-to-image translation with a conditional GAN, 2022) https://www.tensorflow.org/tutorials/generative/pix2pix
3. (GANs: Explanation on GANs, 2022) https://wiki.pathmind.com/generative-adversarial-network-gan
4. Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In *Proceedings of the IEEE international conference on computer vision*, pp. 2223-2232. 2017.
5. Wang, Ting-Chun, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798-8807. 2018.
6. (GANs: Introduction to conditional GANs, 2019) https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
7. (Latent spaces, Explanation on latent spaces, 2020) https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d
8. Chollet, Francois. "Deep learning with python, vol. 1." *Greenwich, CT: Manning Publications CO* (2017).

9. Wang, Ting-Chun, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "Video-to-video synthesis." *arXiv preprint arXiv:1808.06601* (2018).

10. (GANs,GANs algorithms, 2019) https://developers.google.com/machine-learning/gan

11. (Cycle GAN, Cycle GANs tutorial, 2019) https://www.tensorflow.org/tutorials/generative/cyclegan

12. Tanaka, Fabio Henrique Kiyoiti dos Santos, and Claus Aranha. "Data augmentation using GANs." *arXiv preprint arXiv:1904.09135* (2019).

13. (Synthetic data: Benefits and characteristics of synthetic data,2021) https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/

14. Xu, Weihuang, Nasim Souly, and Pratik Prabhanjan Brahma. "Reliability of gan generated data to train and validate perception systems for autonomous vehicles." In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 171-180. 2021.

15. Zou, Xinrui. "A Review of object detection techniques." In *2019 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, pp. 251-254. IEEE, 2019.

16. (Object detection: Deep learning based object detection, 2022) https://viso.ai/deep-learning/object-detection/

17. (Object detection: Deep learning based object detection, 2021) https://machinelearningmastery.com/object-recognition-with-deep-learning/

18. (GANs, Evaluation of GANs, 2019) https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/

19. (Object detector, Model for Yolov3, 2020) https://colab.research.google.com/github/ultralytics/yolov3/blob/master/tutorial.ipynb#scrollTo=zR9ZbuQCH7FX

20. (Activation Functions, Activation functions for deep learning, 2021) https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=An%20activation%20function%20in%20a,a%20layer%20of%20the%20network.

21. O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." *arXiv preprint arXiv:1511.08458* (2015).

22. Chang, Trenton, Akash Modi, and Roshan Toopal. "Beyond Vid2Vid: Few-Shot Video Synthesis for Road Scene Generation."

23. Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object detection in 20 years: A survey." *arXiv preprint arXiv:1905.05055* (2019).

24. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521, no. 7553 (2015): 436-444.

25. Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014).

26. Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. "Generative adversarial networks: An overview." *IEEE Signal Processing Magazine* 35, no. 1 (2018): 53-65.

27. Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747* (2016).

28. Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." *Advances in neural information processing systems* 30 (2017).

29. Harshvardhan, G. M., Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. "A comprehensive survey and analysis of generative models in machine learning." *Computer Science Review* 38 (2020): 100285.

30. Ruthotto, Lars, and Eldad Haber. "An introduction to deep generative modeling." *GAMM‐Mitteilungen* 44, no. 2 (2021): e202100008.

31. (Normalizing flow, Introduction, 2021) https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b#:~:text=In%20simple%20words%2C%20normalizing%20flows,is%20not%20a%20reversible%20function.