Chapter 1

Introduction

In the quest to advance autonomous driving technologies, the creation of sophisticated machine learning models is imperative. These models are the linchpins of an autonomous vehicle's perception system, enabling them to interpret and navigate through the multifaceted environments they encounters. An essential aspect of training these models is the inclusion of diverse scenarios that vehicles may face on the road, with a particular emphasis on critical and dynamic situations involving emergency vehicles. The presence of emergency vehicles is a common yet complex event that requires an immediate and appropriate response from autonomous systems, posing a significant challenge in the realm of autonomous vehicle training. The traffic scene analyzed by a YOLO-NAS (You Only

Figure 1.1: YOLO-NAS Object Detection Model

Look Once) object detection model reveals a common problem: the misclassification of emergency vehicles, in this case, a fire truck mistakenly labeled as a bus. YOLO is

known for its speed and accuracy in real-time object detection. It divides an image into a grid and makes predictions on multiple objects within the grid, enabling simultaneous detection of different objects. YOLO-NAS is a more recent advancement that incorporates Neural Architecture Search (NAS), which automates the design of neural network architectures to optimize various parameters.

This research project aims to improve the accuracy of models in recognizing emergency vehicles, addressing a challenge that's often caused by inadequate training data. By enriching datasets with a broader range of emergency vehicle images, the project seeks to enhance the precision of object detection systems for critical real-world applications.

Recognizing that publicly available datasets lack crucial instances of emergency scenarios, this thesis proposes a two-pronged strategy to improve the data landscape for autonomous driving systems. The research is based on real-world data sourced from the extensive repository of 911 emergency call reports from Orange County. For context, in the United States, "911" is the emergency telephone number that people call for immediate assistance in situations like accidents, fires, or medical emergencies. These reports offer a rich array of emergency situations and contain a range of descriptive details that highlight the urgency and complexity of real-life events.

The first phase of this study focuses on text scene generation, where the textual content of 911 calls is converted into structured scene descriptions. This task involves extracting narrative elements and reshaping them into a format that generative models can understand. By capturing the subtle nuances from these reports, the research generates a set of textual scenes representing various emergency situations with differing levels of complexity and detail.

Building on this text-based foundation, the research moves to the second phase: image scene generation. Here, the descriptive text scenes are used to create synthetic images that depict the emergency scenarios. This step is crucial because it translates the unpredictable and varied nature of real-world emergencies into visual data that can be used to train and evaluate autonomous driving systems. The synthetic images aim to capture a range of lighting, weather, and environmental conditions, as well as the positioning and

signaling of emergency vehicles as described in the 911 call narratives.

The generated images are not just visual representations; they are infused with contextual richness from the original 911 call reports, designed to challenge and improve the perception algorithms of autonomous vehicles. The fidelity of these images to real-world scenarios is critical, as is the variety of scenes they represent. To achieve this, the research employs advanced techniques in deep learning, computer vision, and computational graphics, ensuring that the synthesized data is realistic and varied enough to significantly boost the performance of object detection and classification algorithms.

Moreover, the thesis addresses the complexities of data synthesis, examining the ethical implications and methodological challenges associated with creating and using synthetic data. It maintains a high standard of quality to prevent the introduction of unintended biases or artifacts that could affect the performance of autonomous systems.

In summary, this thesis contributes to the field of autonomous driving by introducing a unique data augmentation approach that leverages real-world emergency scenarios. Through meticulous synthesis of text and image data, it enriches existing datasets with a range of challenging emergency vehicle situations. This ultimately equips autonomous driving models with the diverse experiences needed to navigate safely and effectively in an unpredictable world.

1.1 Contributions

The following items summarize the research's main contribution. Each one of these contributions will be elaborated on in depth in subsequent chapters.

- Real-World Data Utilization: Our research begins with the innovative approach of harnessing real-world data derived from 911 emergency call reports in Orange County. This foundational dataset provides a rich tapestry of emergency scenarios, each brimming with the urgency and complexity characteristic of real-life situations. By integrating these detailed narratives into our dataset, we ensure that the training material for autonomous driving systems reflects the true diversity and dynamism

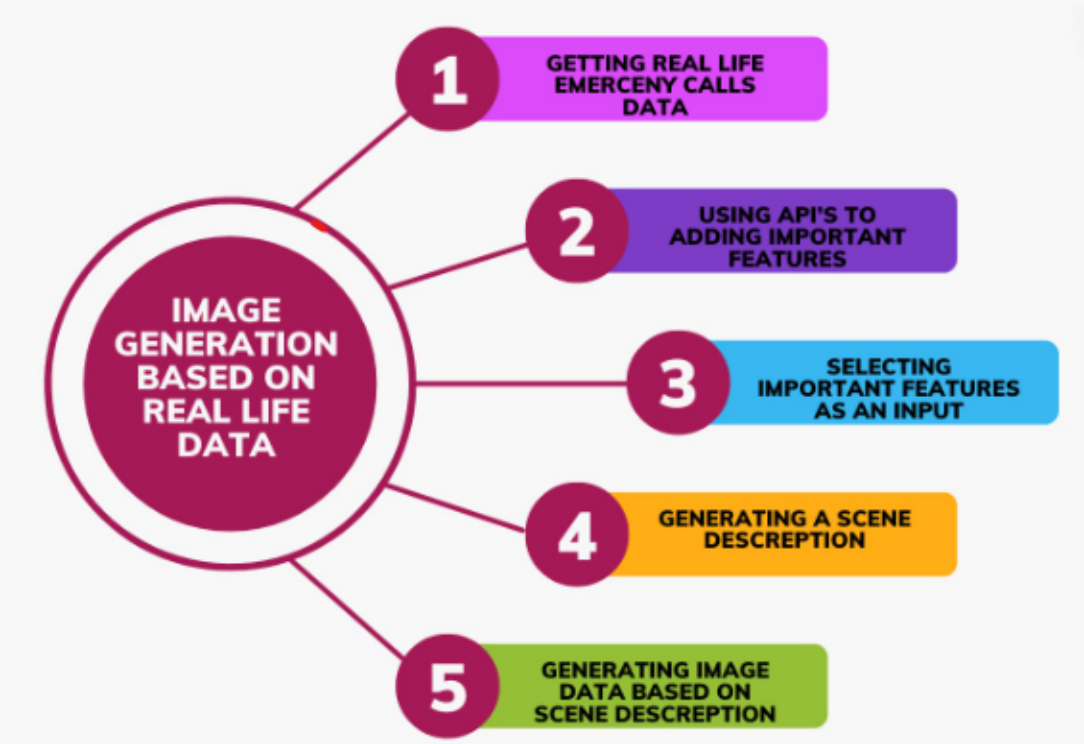

Figure 1.2: General Structure of Thesis

of on-road emergency encounters.

- Textual Scene Generation: We introduce a methodical process for transforming the narrative content of 911 emergency calls into structured, detailed scene descriptions. This crucial step involves meticulously extracting narrative elements and translating them into a format that is both comprehensible and actionable for diffusion models. The resulting textual scenes represent a broad spectrum of emergency situations, providing a diverse base for further synthetic data generation.

- Synthetic Image Generation: Building on the textual descriptions, we advance to generating synthetic images that visually depict the emergency scenarios outlined in the text. This stage is pivotal in translating the descriptive narratives into visual data, enriching the training and evaluation pools for autonomous vehicle perception systems with images that reflect the unpredictability of real-world emergencies under varying conditions.

  These contributions make AD training more realistic and effective, helping systems handle real-world emergencies better. Furthermore, by using real-world data

and innovative techniques, this research directly improves how accurately AV can understand and respond to emergencies, making transportation safer and smoother

1.2 Structure of the Thesis

Chapter 1: Introduction - This chapter sets the stage for the research, establishing the thesis's central question, objectives, and the significance of generating synthetic image data for emergency cases.

Chapter 2: Literature Review - A thorough analysis of existing research is presented, focusing on synthetic image generation and machine learning as they relate to virtual scenes, highlighting the current gaps the thesis aims to address.

Chapter 3: Pipeline - Details the methodology and workflow used for data collection, processing, and analysis of real-world data.

Chapter 4: Reverse Engineering - Discusses the process of deconstructing existing models to understand their functioning and identify opportunities for refinement and innovation within the context of the research.

Chapter 5: Language Models - Explores the use of large language models for generating textual descriptions, which are essential for creating accurate and contextually relevant synthetic images.

Chapter 6: Image Generation and Results - Describes the image generation process, specifically using diffusion models, and presents the results, along with a discussion on the effectiveness and potential uses of the synthetic images generated.

Chapter 7: Conclusion - Summarizes the findings of the research, reflects on the implications and contributions of the study, and proposes directions for future research.

Chapter 2

Literature Review

The domain of transportation is undergoing a transformative change, with Autonomous Driving (AD) at the vanguard of this revolution. As technology continues its relentless march forward, AD ushers in an era where the twin ideals of transportation safety and efficiency are closer to realization than ever before (Feng et al., 2023). However, every revolution has its challenges, and in the realm of AD, ensuring the consistent safety and reliability of algorithm-driven vehicles across varied traffic scenarios stands out (Feng et al., 2023).

With the burgeoning role of AD in modern transport, the value of simulation environments has skyrocketed. These aren't just digital recreations but pivotal testbeds where AD systems are rigorously evaluated, mimicking real-world demands (Manivasagam et al., 2020).

Yet, a deep dive into the simulation realm reveals some constraints: established simulators, including CARLA (A. Dosovitskiy & Koltun, 2017) and SMARTS (Ming & Jun, 2020), primarily rely on predefined, hand-crafted rules for their traffic scenarios (Feng et al., 2023). This approach, while valuable, might not capture the intricate complexities of real-world traffic dynamics fully. This could include unpredictable human behaviors, varying road conditions, and a wide range of environmental factors.

This discrepancy emphasizes the need for more evolved simulations, ones that can authentically mirror real-world traffic nuances. Enhancing simulation fidelity is essen-

tial to thoroughly evaluate the Artificial Intelligence-driven (AI-driven) decision-making prowess of AD systems in diverse scenarios (Feng et al., 2023).

Pushing the envelope further, crafting high-fidelity traffic simulations becomes a linchpin in the journey towards perfecting self-driving capabilities (Tan et al., 2021). These simulations act as incubators for self-driving vehicles (SDVs), refining their functions and assuring their top-tier performance (Tan et al., 2021). However, achieving true-to-life virtual replications is not without hurdles. Contemporary AD projects often draw from real-world recorded scenarios (Manivasagam et al., 2020) or depend on engineering expertise to manually craft fresh scenarios (A. Dosovitskiy & Koltun, 2017), (Lopez et al., 2018). While these approaches offer a semblance of realism, they inherently raise scalability issues (Tan et al., 2021). Such challenges have fueled the pursuit of automation in traffic scenario generation, emphasizing granular details from actor dynamics to their in-scene interactions (Tan et al., 2021).

Zooming out, simulations play an undeniable role in shaping the future of autonomous vehicles (E. Pronovost & Roy, 2023). Cutting-edge advancements in graphics and realistic physics within simulators have enriched the development and validation processes of AD models (E. Pronovost & Roy, 2023). Yet, a formidable challenge looms: aligning simulated realities seamlessly with real-world intricacies. Whether it's ensuring accurate vehicle orientations or emulating nuanced human driving behaviors, the quest for unmatched authenticity is complex (E. Pronovost & Roy, 2023). Ground-breaking solutions in this arena are gravitating towards leveraging deep learning paradigms, showcasing the potential to craft intricate and expansive simulated terrains (E. Pronovost & Roy, 2023).

As we stand at the forefront of these technological advancements, our paper makes a crucial contribution to the field of autonomous driving. We bridge the gap between theoretical models and practical applications, providing a nuanced understanding of how deep learning and virtual scene generation coalesce to enhance AD systems' efficacy. This paper not only dissects the underlying mechanics of these systems but also paves the way for future innovations in the domain. The following section delves into the methodology

of our comprehensive survey, detailing how we dissect current state-of-the-art techniques in virtual scene generation and evaluate their impact on the evolution of autonomous driving technologies.

2.1 Deep Learning Methods

In the world of autonomous driving, the significance of deep learning cannot be understated. These methods, rooted in neural network architectures, have revolutionized the way vehicles perceive, understand, and navigate their environments. This section delves into the most prevalent deep learning techniques that have been instrumental in the advancement of autonomous driving and scene generation.

2.1.1 Convolutional LSTM (ConvLSTM) Network

The ConvLSTM network represents a sophisticated development beyond the classic LSTM (Long Short-Term Memory) network, tailored specifically for handling data that embodies both spatial and temporal dynamics. Unlike the traditional LSTM, which is built on fully connected architectures, the ConvLSTM integrates convolutional components into the mechanisms that govern both the flow of information from input to the network's internal state and the transitions between states over time. This architectural enhancement renders the ConvLSTM adept at managing tasks where understanding the spatial relationships within the data is key.

In particular, the ConvLSTM has been deployed to tackle the challenge of precipitation nowcasting, a task focused on forecasting the near-term intensity of rainfall within a specific locale. Studies comparing the ConvLSTM's performance against that of its fully connected LSTM counterpart have demonstrated its superior ability to grasp the complex spatiotemporal patterns essential for accurate precipitation prediction. Experiments exploring various ConvLSTM configurations—altering layer counts and kernel dimensions—consistently show that the ConvLSTM models excel, offering marked improvements over the fully connected LSTM(FC-LSTM) approach in this domain.(Graves,

2013).

As described in Figure 2.1, the observation of a dynamical system is conducted over an $M \times N$ grid, with each cell containing P time-varying measurements. This system can be represented at any time as a tensor $X_t \in \mathbb{R}^{P \times M \times N}$. The key objective is to predict future sequences $\tilde{X}_{t+1}, \ldots, \tilde{X}_{t+K}$ from past J observations, including the current state X_t. This methodology is crucial for modeling the spatial-temporal dynamics of the system, incorporating the interactions among variables M, N, P, and the tensors X, $\hat{X}$, $\tilde{X}$. the ConvLSTM updates are followed formulation of FC-LSTMas in Figure 2.1:

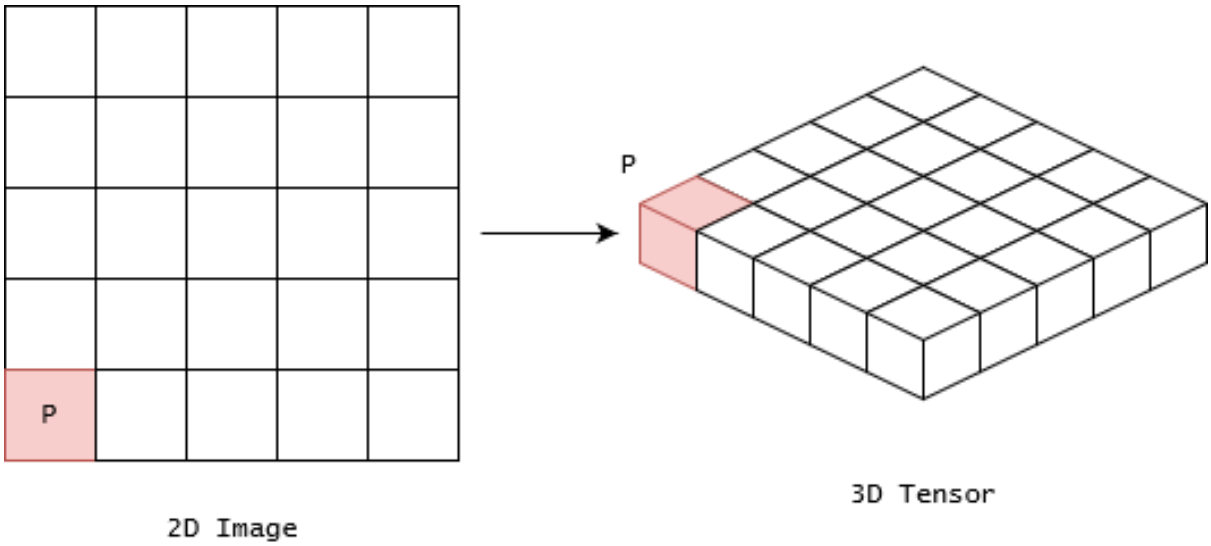

Figure 2.1: Transforming 2D image to 3D

1. **Input Gate**:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i)$$

2. **Forget Gate**:

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f)$$

3. **Cell State**:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c)$$

4. **Output Gate**:

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_t + b_o)$$

5. **Hidden State**:

$$H_t = o_t \odot \tanh(C_t)$$

Where:

- $*$ denotes the convolutional operation.
- $\odot$ denotes the Hadamard product (element-wise multiplication).
- σ is the sigmoid activation function.
- W and b are the weights and biases for the respective gates.

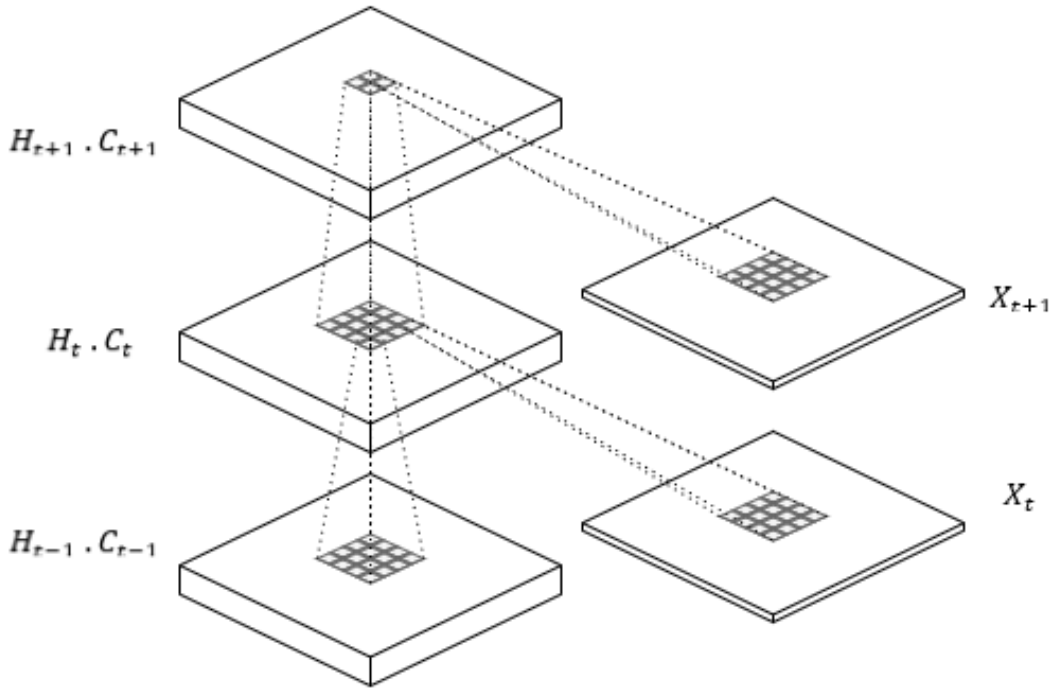

Figure 2.2: Structure (inner) of ConvLSTM

2.1.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) were introduced by Ian J. Goodfellow and his colleagues. The core idea behind GANs is to have two networks, a generative model (G) and a discriminative model (D), trained simultaneously through adversarial training.

- The **Generative Model (G)** tries to capture the data distribution. It takes random noise as input and generates samples as output.
- The **Discriminative Model (D)** estimates the probability that a given sample is from the real training data rather than generated by G.

The training methodology for the generator (G) involves optimizing it to increase the likelihood of deceiving the discriminator (D) into errors of judgment. This dynamic is akin to a scenario where G aims to produce indistinguishable counterfeit money, while D strives to differentiate between authentic and counterfeit bills. As the process evolves, G becomes adept at creating currency so convincing that D is unable to distinguish the genuine from the fraudulent.(Goodfellow et al., 2014).

This adversarial interaction leads to G producing data of exceptional quality. The structure of this relationship is analogous to a minimax game involving two players. Within the realm of potential functions for G and D, there exists a unique equilibrium where G perfectly mimics the distribution of the training data, and D, unable to make a distinction, assigns a probability of 0.5 across the board.(Goodfellow et al., 2014).

Given:

- $G(z;\theta_g)$: A function that maps from noise z to data space using parameters θ_g.
- $D(x;\theta_d)$: A function that outputs the probability that x came from real data.

The training process involves the following minimax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

Where:

- The first term $\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)]$ represents the log probability that the discriminator correctly classifies real data as real.
- The second term $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$ represents the log probability that the discriminator correctly classifies fake data (generated by G) as fake.

2.1.3 Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) (Doersch, 2016; Kingma & Welling, 2014; Salimans et al., 2015; Rezende et al., 2014; Kulkarni et al., 2015) have become a pivotal approach in the unsupervised learning domain for modeling complex distributions. Built upon conventional function approximators like neural networks, VAEs are amenable to training through stochastic gradient descent. Their prowess in generating a multitude of complex data types, ranging from handwritten digits (Kingma & Welling, 2014; Salimans et al., 2015) and facial images (Kingma & Welling, 2014; Rezende et al., 2014; Kulkarni et al., 2015) to house numbers (Kingma et al., 2014; Gregor et al., 2015), CIFAR images (Gregor et al., 2015), models of physical scenes (Kulkarni et al., 2015), image segmentation tasks (Sohn et al., 2015), and the forecasting of future scenarios from still images (Walker et al., 2016), underscores their broad applicability and efficiency in generating varied complex datasets.

The core principle of Variational Autoencoders (VAEs) is their proficiency in capturing the distributions $P(X)$ across data points X that may exist in spaces of high dimensionality. Consider the example of images, where each image (or data point) is composed of thousands to millions of dimensions, represented by pixels. The fundamental goal of the VAE's generative model is to model the interdependencies among these vast dimensions(Doersch, 2016).

Variational Autoencoders (VAEs) adopt a novel strategy for addressing the difficulty associated with specifying latent variables z. Rather than assigning specific meanings to each dimension of z through manual selection, VAEs propose that the dimensions of z

Figure 2.3: Example of CIFAR Image (Gregor et al., 2015)

do not lend themselves to direct interpretation (Doersch, 2016). This perspective leads to the practice of generating samples of z from a straightforward distribution, such as a standard normal distribution $N(0, I)$, where I represents the identity matrix (Kingma et al., 2014).

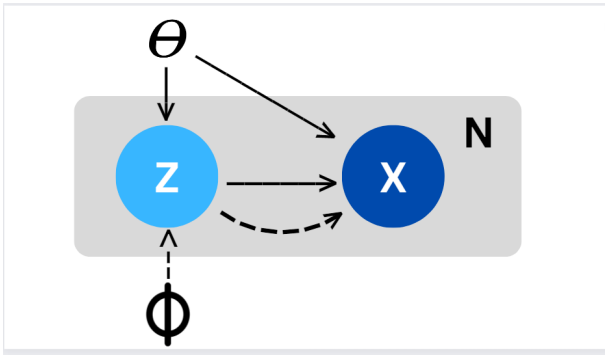

Figure 2.4: Example graphical model for VAEs.

1. The generative model's job is to capture the distribution $P(X)$.

2. VAEs model the distribution as:

$$P(Y|X) = N(f(z, X), \sigma^2 \times I)$$

where f is a deterministic function learned from data.

3. The objective function for VAEs can be represented as:

$$\begin{aligned}&\mathbb{E}_{X\sim D}\left[\log P(X) - D[Q(z|X)\|P(z|X)]\right]\\&= \mathbb{E}_{X\sim D}\left[\mathbb{E}_{z\sim Q}[\log P(X|z)] - D[Q(z|X)\|P(z)]\right]\end{aligned}$$

4. Another representation of the objective function is:

$$\begin{aligned}&\log P(Y|X) - D[Q(z|Y,X)\|P(z|Y,X)]\\&= \mathbb{E}_{z\sim Q(\cdot|Y,X)}[\log P(Y|z,X)] - D[Q(z|Y,X)\|P(z|X)]\end{aligned}$$

2.1.4 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

Introduced in 1997, the Long Short-Term Memory (LSTM) network, an advanced variant of the Recurrent Neural Network (RNN), has since played a pivotal role across various fields including language modeling, speech recognition, and machine translation (Hochreiter & Schmidhuber, 1997; Lin & Tegmark, 2017). The LSTM was primarily developed to address the vanishing gradient problem—a significant issue in standard RNNs. Its unique architecture features nonlinear, data-driven gates within the RNN cell, which help preserve the gradient of the loss function over time (Sherstinsky, 2020; Hochreiter & Schmidhuber, 1997). Traditional RNNs often struggle with training due to vanishing or exploding gradients. The LSTM cell architecture, with its ability to regulate the flow of information, offers a robust solution to sustain the objective function's gradient with respect to the state signal, making it a critical advancement in neural network design (Hochreiter & Schmidhuber, 1997; Graves, 2008; Pascanu et al., 2013; Pascanu, 2014).

At the heart of RNNs lies the concept of a hidden state h_t, serving as a memory of the information from past sequences. This state is dynamically updated at every timestep t, influenced by the current input x_t and the preceding hidden state h_{t-1} (Sherstinsky, 2020):

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2.2}$$

Where:

- W_{hh} and W_{xh} are weight matrices.
- b_h is the bias.
- σ is an activation function, often the hyperbolic tangent.

The LSTM cell can be described by the following set of equations(Sherstinsky, 2020):

$$\text{Forget Gate:} \quad f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

$$\text{Input Gate:} \quad i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\text{Cell Update:} \quad \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.5}$$

$$\text{New Cell State:} \quad C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.6}$$

$$\text{Output Gate:} \quad o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$\text{New Hidden State:} \quad h_t = o_t \times \tanh(C_t) \tag{2.8}$$

Where:

- W terms denote weight matrices (e.g., W_f is the weight matrix for the forget gate).
- b terms denote bias vectors.
- σ is the sigmoid activation function.
- tanh is the hyperbolic tangent activation function.
- $[h_{t-1}, x_t]$ denotes the concatenation of h_{t-1} and x_t.

2.1.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized kind of neural network designed primarily for processing grid-like data structures, such as images. They are analogous

to traditional Artificial Neural Networks (ANNs) but are optimized for image processing tasks(O'Shea & Nash, 2015; Ciresan et al., 2012, 2011, 2013).

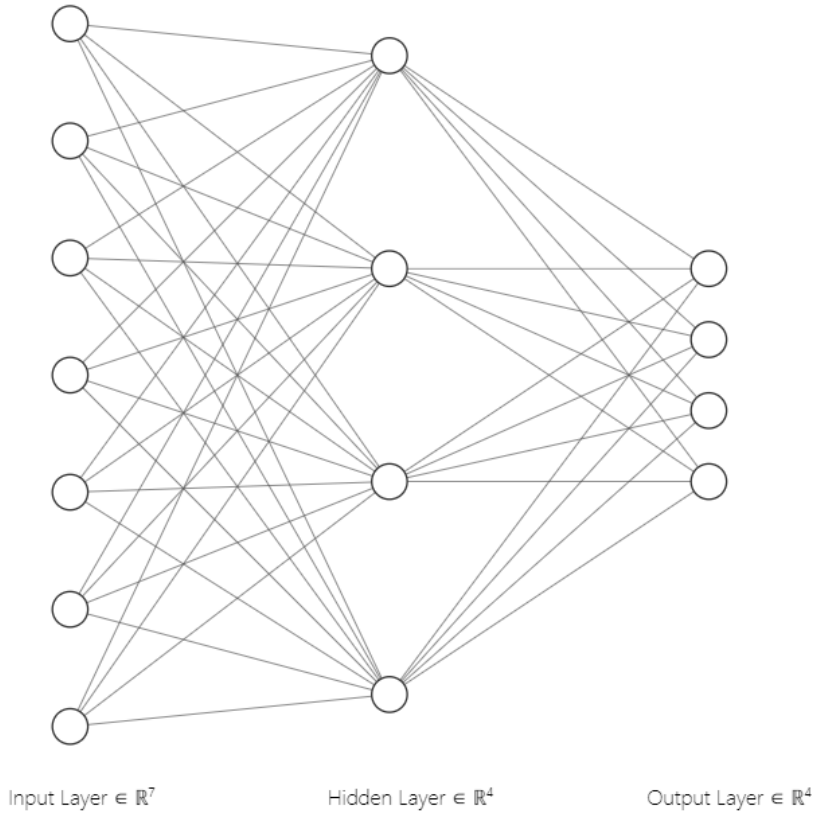

Figure 2.5: Example CNN Architecture.

1. **Input Layer**: The input layer of a CNN typically holds the pixel values of an image. For instance, for a grayscale image, the input layer would contain the intensity values of each pixel, while for a colored image, it would contain RGB values.(O'Shea & Nash, 2015)

2. **Convolutional Layer**: The convolutional layer is the core building block of a CNN. It focuses on local regions of the input and computes the dot product between the weights of the network and the region of the input image. This layer helps in detecting local features like edges, textures, and patterns.

3. **Pooling Layer**: Pooling layers are used to reduce the spatial dimensions of the data, thereby reducing the computational complexity. Max-pooling is a common

technique where the maximum value from a region of the image is taken.

4. **Fully-Connected Layer**: This layer is similar to the traditional ANN, where neurons are fully connected to the adjacent layers. It helps in classifying the features extracted by the convolutional layers into various classes.

5. **Output Layer**: The final layer provides the classification result, assigning the input image to one of the classes based on the features recognized by the network.

The foundational building blocks of a Convolutional Neural Network (CNN) include several key operations that allow these networks to efficiently process and learn from image data. Each of these operations plays a distinct role in the network's ability to identify patterns and features in input data, ultimately contributing to the network's predictive capabilities. Below is an overview of these essential operations:

1. **Convolutional Operation**: Given an input matrix I and a filter (or kernel) matrix F, the convolution operation is defined as:

$$(I * F)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) \times F(x - i, y - j) \tag{2.9}$$

   This operation slides the filter over the input matrix and computes the dot product at each position.

2. **Pooling Operation**: In max-pooling, for a given region R in the matrix, the output is:

$$\text{max-pool}(R) = \max(R) \tag{2.10}$$

   Where $\max(R)$ represents the maximum value in the region R.

3. **Activation Function**: After the convolution operation, an activation function, such as the Rectified Linear Unit (ReLU), is applied element-wise to introduce non-linearity:

$$\text{ReLU}(x) = \max(0, x) \tag{2.11}$$

4. **Fully-Connected Operation**: In this layer, given an input vector x and weight matrix W, the output is:

$$y = Wx + b \tag{2.12}$$

Where b is the bias vector.

2.1.6 Transformer-based Models

The transformer architecture is a novel neural network design introduced by Vaswani et al. in 2017. It revolutionized the field of natural language processing (NLP) by offering a mechanism to handle sequences without relying on recurrent layers. The core idea behind transformers is the attention mechanism, which allows the model to weigh the importance of different parts of the input data when producing an output(Vaswani et al., 2017a).

The transformer consists of an encoder and a decoder. The encoder processes the input sequence, while the decoder generates the output sequence. Both the encoder and decoder are composed of multiple identical layers, and each layer has two main components: a multi-head self-attention mechanism and a position-wise feed-forward network(Vaswani et al., 2017a).

The multi-head self-attention mechanism allows the model to focus on different parts of the input sequence simultaneously. It computes a weighted sum of input values (or "values") based on their relevance (or "attention scores") to a given input position (or "query"). The attention scores are determined by comparing the query with all other input positions (or "keys")(Vaswani et al., 2017a).

1. **Attention Score Calculation:**

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.13}$$

Where:

- Q is the matrix of queries.
- K is the matrix of keys.

- V is the matrix of values.
- d_k is the dimension of the key vectors.

2. **Multi-Head Attention:**

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O \tag{2.14}$$

Where:

- Each head_i is the result of the attention mechanism applied to linearly transformed versions of Q, K, and V.
- W_O is the output weight matrix.

3. **Position-wise Feed-Forward Network:**

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.15}$$

Where:

- W_1 and W_2 are weight matrices.
- b_1 and b_2 are bias vectors.

2.1.7 Causal Autoregressive Flows:

Causal Autoregressive Flows bridge the gap between two significant fields in machine learning: normalizing flows and causality(Khemakhem et al., 2021). At its core, this concept highlights a correspondence between a subset of autoregressive normalizing flows and identifiable causal models.

Normalizing Flows: These are models that aim to express the log-density of observations as an invertible and differentiable transformation of latent variables. The latent variables typically follow a simple base distribution, and the density of the observations can be derived using a change of variables formula(Papamakarios et al., 2019; Kobyzev et al., 2020; Khemakhem et al., 2021). Autoregressive flows, a subtype of normalizing flows,

are designed to simplify the computation of the Jacobian determinant by ensuring their Jacobian matrices are lower triangular(Papamakarios et al., 2018; Huang et al., 2018).

Causality: In the context of this work, autoregressive flow architectures define an ordering over variables, similar to a causal ordering. This makes them suitable for various causal inference tasks, such as causal discovery, interventional predictions, and counterfactual predictions(Khemakhem et al., 2021).

The Causal Autoregressive Flow (CAREFL) model is introduced as a method to leverage the properties of autoregressive flows for causal inference. This model is particularly powerful because it can identify the true direction of causal influence and provides a generalization of the additive noise model, which is a well-known model in causal discovery(Khemakhem et al., 2021).

1. The density of observations using normalizing flows:

$$p_x(x) = p_z(T^{-1}(x))|detJT^{-1}(x)|$$

   Where:

   - p_x is the density of observations.
   - p_z is the density of latent variables.
   - T is the transformation.
   - J is the Jacobian matrix.

2. The transformation in autoregressive flows:

$$x_j = \tau_j(z_j, x_{<\pi(j)})$$

   Where:

   - τ_j are called transformers.
   - π is a permutation that specifies an autoregressive structure on x.

3. For affine transformations in the autoregressive flow:

$$\tau_j(u, v) = e^{s_j(v)}u + t_j(v)$$

Where:

- s_j and t_j are functions defining the affine transformation.

4. The measure of causal direction is defined as:

$$R = E[logL_{\pi=(1,2)}(x_{test}; x_{train})] - E[logL_{\pi=(2,1)}(x_{test}; x_{train})]$$

Where:

- R is positive if x_1 is the causal variable and negative if x_2 is the causal variable.

2.2 Compilation

Authors	Scene Scenarios	Data Sources	Theoretical Frameworks
(D. Zhao & Liu, 2017)	Traffic scene	Primary data (large amounts of traffic scene videos and GPS data) and public data (KITTI dataset)	Architecture: GAN (Deep convolutional generative adversarial network) Developed by authors: No
(H. Avşar & Karacan, 2022)	Home/office scene	Primary data (Chess scene of 7-Scenes dataset)	Architecture: Generative adversarial networks Developed by authors: No
(P. Cai & Liu, 2020)	Driving scene	Public data (RobotCar dataset)	Architecture: LSTM network (VTGNet, an uncertainty-aware vision-based trajectory generation network) Developed by authors: Yes
(M. Qi & Luo, 2020)	Driving scene	Public data (public datasets, i.e., Cityscapes and CamVid)	Architecture: GAN: STC-GAN (Spatio-Temporally Coupled Generative Adversarial Networks) Developed by authors: Yes
(E. Pronovost & Roy, 2023)	Driving scene	Public data (dataset containing 6 million driving scenes from San Francisco, Las Vegas, Seattle, and the campus of the Stanford Linear Accelerator Center)	Architecture: Variational autoencoders (Scene diffusion) Developed by authors: Yes
(Periyasamy & Behnke, 2023)	Object scene	Public data (DeepCPD dataset)	Architecture: Not specified deep-learning technique Differentiable rendering (StilllebenDR, a lightweight, scalable differentiable renderer) Developed by authors: Yes
(A. Ghosh & Chowdhury, 2016)	Driving scene	Public data (game Road Rash)	Architecture: GAN Developed by authors: No
(W. Ding & Zhao, 2023)	Driving scene	Public data	Architecture: Not specified deep-learning technique Causal Autoregressive Flow (CausalAF) Developed by authors: Yes
(J. Devaranjan & Fidler, 2020)	Driving scene	Primary data (synthetic aerial imagery) and public data (MNIST dataset)	Architecture: No specified deep-learning technique Meta-Sim2 (a graph generation model of synthetic scenes) Developed by authors: Yes
(Kar et al., 2019)	Driving scene	Public data (MNIST-like data)	Architecture: No specified deep-learning technique Meta-Sim2 (a graph generation model of synthetic scenes) Developed by authors: Yes
(Tan et al., 2021)	Traffic scene	Public data (Argoverse and ATG4D)	Architecture: Neural autoregressive model (SceneGen) Developed by authors: Yes
(D. J. Fremont & Seshia, 2019)	Driving scene	Public data (Grand Theft Auto V)	Architecture: No specified deep-learning technique Scenic: A probabilistic programming language for scenario specification and scene generation Developed by authors: Yes
(Bergamini et al., 2021)	Driving scene	Public data (Lyft Motion Prediction Dataset)	Architecture: No specified deep-learning technique Scenic: A probabilistic programming language for scenario specification and scene generation Developed by authors: Yes
(Feng et al., 2023)	Traffic scene	Public data (Waymo Open Dataset)	Architecture: Transformer-based model (autoregressive neural generative model) TrafficGen: A traffic scenario generator Developed by authors: Yes

Table 2.1: Summary of scene scenarios, data sources, and theoretical frameworks

Table 2.1 presents a compilation of 14 relevant studies focusing on virtual scene applications within autonomous driving (AD). These studies cover various aspects, including research problems, data sources, theoretical frameworks, performance metrics, overall performance, and data processing/augmentation techniques. The approaches employed diverse scene generation methods such as GANs, causal autoregressive flow, and scene diffusion. For example, several papers utilized GAN principles or deep convolutional generative adversarial networks (DCGAN) to generate realistic traffic scenes, improving driving safety (Bergamini et al., 2021), (Feng et al., 2023), (D. J. Fremont & Seshia, 2019) (A. Ghosh & Chowdhury, 2016), (Tan et al., 2021), (D. Zhao & Liu, 2017).

One study introduced the STC-GAN, a novel GAN model, for predictive scene parsing (M. Qi & Luo, 2020) . Instead of generating safety-critical scenarios, another paper used a causality-based generative model to estimate risk situations within generated driving scenes, enhancing AV safety (W. Ding & Zhao, 2023) . Some studies focused on optimizing parameters of real images for synthetic data generation using techniques like Meta-Sim2 (J. Devaranjan & Fidler, 2020) and Meta-Sim (Kar et al., 2019) .

However, certain papers delved beyond driving scene generation. For instance, one study employed Pix2pixHD, a type of GAN, to convert night-time LIDAR or Kinect sensor images into daytime images with depth map data, yielding indistinguishably realistic outputs (H. Avşar & Karacan, 2022). In addressing the inefficiencies of traditional AD approaches to respond to varied driving scenarios, researchers proposed VTGNe, an uncertainty-aware vision-based generation network, to enhance vehicle navigation, safety measures, and recovery behaviors against driving errors (P. Cai & Liu, 2020). Another work generated complex driving scenes through diffusion and object detection in an end-to-end differentiable architecture (E. Pronovost & Roy, 2023) . Additionally, a study generated 2D images from 3D scene descriptions using an efficient scalable differentiable renderer called StilllebenDR (Periyasamy & Behnke, 2023) .

2.2.1 Scene Scenarios

The review identified five primary scene scenarios: driving scenes (9 papers), traffic scenes (3 papers), home/office scenes (1 paper), and object scenes (1 paper). These efforts aimed to simulate real-world scenes (Feng et al., 2023; A. Ghosh & Chowdhury, 2016; D. Zhao & Liu, 2017; P. Cai & Liu, 2020; E. Pronovost & Roy, 2023) , generate 2D images from 3D scene descriptions (H. Avşar & Karacan, 2022) (Periyasamy & Behnke, 2023), or predict scene parsing (M. Qi & Luo, 2020) .

2.2.2 Research Problems

All reviewed works directly or indirectly aimed to enhance AD or human safety. The most common problem addressed was the scarcity of rich real-world data for safe and efficient AD (10 papers). Many researchers highlighted insufficient driving/traffic scene data for training unmanned vehicles in specific or hazardous environments (Bergamini et al., 2021; Feng et al., 2023; D. J. Fremont & Seshia, 2019; A. Ghosh & Chowdhury, 2016; Tan et al., 2021; D. Zhao & Liu, 2017; W. Ding & Zhao, 2023; J. Devaranjan & Fidler, 2020; Kar et al., 2019; E. Pronovost & Roy, 2023) . Other addressed problems included converting night-time images to daytime images (H. Avşar & Karacan, 2022), AD vehicles' lack of imitation learning capability (P. Cai & Liu, 2020) , incompatibility of the standard rasterization and deep learning methods (Periyasamy & Behnke, 2023), and challenges in anticipating future scene parsing with limited annotated training data (M. Qi & Luo, 2020) .

2.2.3 Data Sources

The papers utilized primary data, public data, or both to tackle their research problems. Most studies used publicly available datasets such as KITTI datasets (D. Zhao & Liu, 2017) , RobotCar datasets (P. Cai & Liu, 2020), Cityscapes and CamVid (M. Qi & Luo, 2020) , DeepCPD datasets (Periyasamy & Behnke, 2023) , Waymo Open Datasets (Feng et al., 2023), and Lyft Motion Prediction Datasets (Bergamini et al., 2021). While

these datasets were predominantly real driving data, a few used data from popular racing games (D. J. Fremont & Seshia, 2019) (A. Ghosh & Chowdhury, 2016). Some authors employed both primary and public data for traffic scene generation (D. Zhao & Liu, 2017) , while others focused on non-traffic/driving scenes using a limited number of input images (H. Avşar & Karacan, 2022) (Periyasamy & Behnke, 2023) . The need for more extensive data was evident in research requiring comprehensive model training and testing.

2.2.4 Data Pre-processing/Augmentation

The vast majority of the reviewed studies (10 papers) employed some pre-processing or augmentation techniques before feeding their data into their model. The most common pre-processing technique was image resize (5 papers), followed by image categorization (4 papers), image vectorization (2 papers), and image labeling (2 papers). Studies resized images to 1247 x 384 (P. Cai & Liu, 2020) , 256 × 256 (M. Qi & Luo, 2020) (E. Pronovost & Roy, 2023), or 240 x 320 (Periyasamy & Behnke, 2023) . Image categorization was also used to categorize raw images into different traffic scenes [6], environmental conditions (P. Cai & Liu, 2020), and city conditions (J. Devaranjan & Fidler, 2020). This categorization is especially invaluable in predicting AD responses to different situations. Image data were also labeled prior to data training to provide the driving direction of vehicles (P. Cai & Liu, 2020) and enable predictive scene parsing (M. Qi & Luo, 2020) . Lastly, some authors converted scene images into a vector format, rasterizing high-definition semantic maps to create a bird's-eye view representation of the driving state (Bergamini et al., 2021) or vectorizing scenes into a set of vectors (Feng et al., 2023). However, not all the papers pre-processed or augmented their data. Some studies that fall under this category fed video-game data into their model without modifying them (D. J. Fremont & Seshia, 2019) (A. Ghosh & Chowdhury, 2016). Others required no data pre-processing as the cameras used were configured to capture ideal formats of images (H. Avşar & Karacan, 2022).

2.2.5 Theoretical Frameworks

Most of the reviewed works (9 papers) employed various deep learning techniques. The most used deep learning technique was GANs (A. Ghosh & Chowdhury, 2016) (D. Zhao & Liu, 2017) (M. Qi & Luo, 2020) (H. Avşar & Karacan, 2022) . Some authors modified their GAN models, using deep convolutional generative adversarial networks (D. Zhao & Liu, 2017) and spatiotemporally coupled generative adversarial networks (M. Qi & Luo, 2020) to train their data, while some studies leveraged the traditional GAN models to solve the image and video processing problems (A. Ghosh & Chowdhury, 2016) (H. Avşar & Karacan, 2022) . However, other related works tested their models using other deep learning techniques, such as LSTM (P. Cai & Liu, 2020), variational autoencoders (E. Pronovost & Roy, 2023), transformer-based models (Feng et al., 2023), and neural autoregressive networks (Tan et al., 2021) .

While many of the reviewed papers pointed out their deep learning approaches, others (6 papers) concentrated on research problems not directly linked to a specific deep learning technique. For instance, in (Periyasamy & Behnke, 2023) , the authors focused on equipping all deep learning methods with the capability of generating 2D images from 3D images using a differentiable render known as StilllebenDR. Similarly, one paper leveraged all learning networks to create a system capable of realistically simulating driving experiences (Bergamini et al., 2021). A few research works also concentrated on aspects such as causal autoregression flow (W. Ding & Zhao, 2023) , the appearance gap between synthetic and real-world images (J. Devaranjan & Fidler, 2020) (Kar et al., 2019) , and programming language for scene generation and specification [3]. Hence, these papers did not need to specify deep learning techniques used.

2.2.6 Performance Metrics

The reviewed studies employed diverse metrics to evaluate their solutions. Some studies (6 papers) used single metrics (D. Zhao & Liu, 2017) (W. Ding & Zhao, 2023) (J. Devaranjan & Fidler, 2020) (Kar et al., 2019) (H. Avşar & Karacan, 2022) (E. Pronovost & Roy, 2023) (Periyasamy & Behnke, 2023), while others used combinations of parameters to

assess their models (Bergamini et al., 2021) (Feng et al., 2023) (D. J. Fremont & Seshia, 2019), (Tan et al., 2021) (M. Qi & Luo, 2020) (P. Cai & Liu, 2020) . A few studies did not explicitly provide metric information but evaluated based on parameters like image appearance and training error (A. Ghosh & Chowdhury, 2016) (D. Zhao & Liu, 2017) .

The most used metric (3 papers) was mean maximum discrepancy (MMD). Authors leveraged this metric to evaluate the degree of similarity between created scenes (E. Pronovost & Roy, 2023) , scene structures (J. Devaranjan & Fidler, 2020) (Kar et al., 2019) ,and vehicle attribute distributions (Feng et al., 2023). Other metrics in the reviewed papers included loss trends (H. Avşar & Karacan, 2022) , pose estimation (Periyasamy & Behnke, 2023), collision rate (W. Ding & Zhao, 2023) , precision and recall (D. J. Fremont & Seshia, 2019), negative loglikelihood (Tan et al., 2021) , and other metric combinations in Table xx (Bergamini et al., 2021) (Feng et al., 2023) (M. Qi & Luo, 2020) (P. Cai & Liu, 2020) .

2.2.7 Overall Performance

Comparing performance across papers proved challenging due to the diverse metrics used for different tasks. While some studies (9 papers) compared their models with previous research (Feng et al., 2023) (Tan et al., 2021) (D. Zhao & Liu, 2017) (M. Qi & Luo, 2020) (W. Ding & Zhao, 2023) (J. Devaranjan & Fidler, 2020) (Kar et al., 2019) (P. Cai & Liu, 2020) (E. Pronovost & Roy, 2023) , others presented their results without comparisons (Bergamini et al., 2021) (D. J. Fremont & Seshia, 2019) (A. Ghosh & Chowdhury, 2016) (H. Avşar & Karacan, 2022) (Periyasamy & Behnke, 2023). Validation against different datasets or real-world scenarios was observed in the works of (J. Devaranjan & Fidler, 2020) , (Feng et al., 2023) (Kar et al., 2019) , and (Tan et al., 2021) . Most studies reported significant improvements, with focus areas ranging from generating realistic driving scenes (Tan et al., 2021) (D. Zhao & Liu, 2017) (J. Devaranjan & Fidler, 2020) (W. Ding & Zhao, 2023) (J. Devaranjan & Fidler, 2020) (Kar et al., 2019) (E. Pronovost & Roy, 2023) (Periyasamy & Behnke, 2023) to enhancing programming languages for scene generation (D. J. Fremont & Seshia, 2019) , traffic trajectories in different urban

areas (P. Cai & Liu, 2020) , semantic classification (M. Qi & Luo, 2020) , AD learning (A. Ghosh & Chowdhury, 2016) , AD safety (W. Ding & Zhao, 2023), and self-driving simulations from real-world observations (Bergamini et al., 2021). Notably, all solutions presented in these papers have limitations, paving the way for future research.

Chapter 3

Pipeline

3.1 Data and Preprocessing

The dataset under investigation is a structured collection in the form of a tibble, consisting of 136,784 records spanning seven distinct attributes. This dataset was obtained from the Orange County Government database in Florida, containing a full year of data. As it is open source, researchers have access to a comprehensive resource for analysis and modeling.

This extensive dataset serves as a critical foundation for further research and development, offering a wide-ranging perspective on the domain under study.

- **DATE**: POSIXct vector representing the date of the emergency call, with a standard format highlighting the specificity of each event occurrence.
- **TIME**: POSIXct vector indicating the time of day the call was logged, formatted to capture the precise moment of each emergency situation.
- **Call TYPE**: Character vector that succinctly classifies the nature of each call using an internal shorthand code.
- **CALL TYPE DESC**: Descriptive character vector providing a more detailed explanation of the call type, such as "MEDICAL-DELTA" for urgent medical situations or "FIRE ALARM" for potential fire-related emergencies.

- **XCOORD** and **YCOORD**: Character vectors representing the longitudinal and latitudinal coordinates respectively, pinpointing the exact location of the incident.
- **ADDRESS**: Character vector providing the physical address associated with each call, further aiding in the geographical pinpointing of emergencies.

	A	B	C	D	E	F	G
1	***Sep 18, 2022 to Sep 18, 2023***						
2	**DATE**	**TIME**	**Call TYPE**	**CALL TYPE DESC**	**XCOORD**	**YCOORD**	**ADDRESS**
3	2022-09-18	6:08:24 PM	EMDD	MEDICAL-DELTA	-81.396331	28.605885	1051 LEE RD, ORG
4	2022-09-18	7:30:14 PM	EMDD	MEDICAL-DELTA	-81.414938	28.631289	8207 FOREST CITY RD, ORG
5	2022-09-20	3:29:47 PM	EMDC	MEDICAL-CHARLIE	-81.406569	28.610950	1803 MOSHER DR, ORG
6	2022-09-20	4:41:13 PM	AFA	FIRE ALARM	-81.414938	28.631289	8207 FOREST CITY RD, ORG
7	2022-09-20	10:58:09 PM	EMDB	MEDICAL-BRAVO	-81.419669	28.625538	3240 RIVERSIDE PARK RD, ORG
8	2022-09-21	12:03:30 PM	EMDC	MEDICAL-CHARLIE	-81.393994	28.606610	933 LEE RD, ORG
9	2022-09-21	5:38:08 PM	EMDA	MEDICAL-ALPHA	-81.393411	28.604347	5135 ADANSON ST, ORG
10	2022-09-22	1:49:45 AM	EMDD	MEDICAL-DELTA	-81.393114	28.607525	5300 ADANSON ST, ORG
11	2022-09-23	8:19:31 AM	AFA	FIRE ALARM	-81.412833	28.618888	2724 ELMHURST CIR, ORG
12	2022-09-23	11:08:46 PM	EMDB	MEDICAL-BRAVO	-81.414938	28.631289	8207 FOREST CITY RD, ORG
13	2022-09-24	4:00:49 PM	AA	VEH ACC	-81.411930	28.618432	W KENNEDY BLVD/DAVAR AV
14	2022-09-25	10:00:09 AM	EMDD	MEDICAL-DELTA	-81.388312	28.607147	606 LEE RD, ORG
15	2022-09-25	10:32:13 AM	EMDC	MEDICAL-CHARLIE	-81.383384	28.601101	420 OLOLU DR, ORG
16	2022-09-25	10:35:58 AM	EMDD	MEDICAL-DELTA	-81.396331	28.605885	1051 LEE RD, ORG
17	2022-09-26	2:06:47 AM	EMDD	MEDICAL-DELTA	-81.412266	28.629695	2513 FALKNER RD, ORG
18	2022-09-27	1:49:26 AM	EMDA	MEDICAL-ALPHA	-81.414938	28.631289	8207 FOREST CITY RD, ORG
19	2022-09-27	10:22:30 AM	AA	VEH ACC	-81.415007	28.629622	FOREST CITY RD/FALKNER RD
20	2022-09-27	2:01:03 PM	ELECK	ELECTRICAL CHECK	-81.423633	28.633298	3617 PEMBROOK DR, ORG
21	2022-09-28	1:18:58 AM	EMDB	MEDICAL-BRAVO	-81.388312	28.607147	606 LEE RD, ORG
22	2022-09-28	12:54:55 PM	AFA	FIRE ALARM	-81.411024	28.614705	2315 LAKE WESTON DR, ORG
23	2022-09-28	4:27:23 PM	AFA	FIRE ALARM	-81.415043	28.621255	7126 FOREST CITY RD, ORG

Figure 3.1: Head of 911 Records Dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	dt	dt_iso	timezo	city_name	lat	lon	temp	visibility	dew_point	feels_like	temp_min	temp_max	pressure	sea_	gr	hum	wind_spe	wind_deg	wir	rain_1h	ra	s	s	clouds_al	weather_i	weather_n	weather_c	weather_i
2	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.15	10000	22.95	25.05	23.98	25.04	1017			93	2.57	80		1.27				100	501	Rain	moderate	10n
3	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.12	10000	22.38	24.94	23.83	24.42	1017			90	2.57	170		0.44				40	500	Rain	light rain	10n
4	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.02	10000	22.28	24.83	23.83	24.41	1017			90	0	0		0.27				40	500	Rain	light rain	10n
5	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.99	10000	22.25	24.79	23.83	24.03	1017			90	1.54	220		0.15				75	500	Rain	light rain	10n
6	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.88	10000	22.32	24.7	23.27	24.03	1016			91	1.54	230						75	803	Clouds	broken cl	04n
7	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.86	10000	22.3	24.68	22.98	24.03	1016			91	0	0						75	803	Clouds	broken cl	04n
8	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.3	10000	22.28	24.14	22.98	23.37	1016			94	0	0						100	804	Clouds	overcast c	04n
9	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.13	10000	22.11	23.95	22.72	23.98	1015			94	1.54	230						40	802	Clouds	scattered	03n
10	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	22.97	10000	21.95	23.78	22.76	23.27	1015			94	2.57	220						0	800	Clear	sky is clea	01n
11	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	22.99	10000	21.97	23.8	22.72	23.3	1014			94	2.06	150						0	800	Clear	sky is clea	01n
12	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	22.96	10000	21.94	23.77	22.72	23.03	1014			94	2.57	230						0	800	Clear	sky is clea	01n
13	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23	10000	22.16	23.84	22.72	23.03	1014			95	2.06	260						0	800	Clear	sky is clea	01n
14	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23	10000	21.98	23.81	22.72	23.3	1015			94	1.54	300						75	803	Clouds	broken cl	04d
15	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.96	10000	22.4	24.79	23.27	25.04	1015			91	1.54	230						0	800	Clear	sky is clea	01d
16	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	25.63	10000	23.11	26.49	23.83	26.08	1016			86	0	0						20	801	Clouds	few cloud	02d
17	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	26.84	10000	22.68	29.22	24.38	27.82	1016			78	1.54	90		0.15				75	500	Rain	light rain	10d
18	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	27.01	10000	22.2	29.3	26.65	28.37	1016			75	2.06	160		0.38				75	500	Rain	light rain	10d
19	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	28.02	10000	22.27	30.84	27.76	29.98	1015			71	4.12	160		0.29				75	500	Rain	light rain	10d
20	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	29.25	10000	22.24	32.51	28.27	29.98	1015			66	4.12	180		0.29				40	500	Rain	light rain	10d
21	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	29.98	10000	22.43	33.62	28.83	30.98	1014			64	3.09	150						75	803	Clouds	broken cl	04d
22	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	31.01	10000	22.06	34.62	29.38	31.09	1014			59	4.12	210						40	802	Clouds	scattered	03d
23	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	30.95	10000	21.44	34.02	30.5	31.15	1013			57	4.12	210						0	800	Clear	sky is clea	01d
24	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	30.92	10000	20.83	33.52	30.53	31.15	1013			55	2.57	190						0	800	Clear	sky is clea	01d
25	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	29	10000	20.18	30.88	28.86	29.98	1013			59	1.54	230						0	800	Clear	sky is clea	01d
26	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	27.97	10000	23.12	31.29	27.75	28.27	1014			75	4.63	120						20	801	Clouds	few cloud	02n
27	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	27.02	10000	23.27	29.78	25.98	27.2	1014			80	4.12	120						0	800	Clear	sky is clea	01n
28	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	25.28	10000	21.99	26	24.98	26.61	1014			82	3.09	130						0	800	Clear	sky is clea	01n
29	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.24	10000	21.36	24.91	22.98	26.05	1014			84	2.57	130						0	800	Clear	sky is clea	01n
30	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.19	10000	21.89	24.94	23.98	25.5	1014			87	1.54	140						0	800	Clear	sky is clea	01n
31	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.07	10000	21.96	24.83	23.87	24.97	1014			88	1.54	140						0	800	Clear	sky is clea	01n
32	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	24.04	10000	22.11	24.82	23.87	24.94	1013			89	2.57	130						0	800	Clear	sky is clea	01n
33	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.9	10000	22.16	24.7	22.98	24.48	1013			90	0	0						0	800	Clear	sky is clea	01n
34	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.15	10000	21.96	23.95	23.03	24.38	1013			93	0	0						0	800	Clear	sky is clea	01n
35	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.83	10000	22.09	24.62	22.82	24.03	1013			90	1.54	80						0	800	Clear	sky is clea	01n
36	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.11	10000	21.92	23.9	23.03	23.98	1013			93	1.54	60						100	804	Clouds	overcast c	04n
37	1.66E+09	2022-09-1	-14400	Orlando	28.53838	-81.3789	23.73	10000	21.99	24.51	22.76	24.03	1013			90	1.54	40						100	804	Clouds	overcast c	04n

Figure 3.2: Head of Weather Dataset.

Preprocessing steps undertaken include the conversion of coordinate data from character to numeric format to facilitate spatial analysis, the parsing of date-time information into a consistent POSIXct format, and the cleansing of any anomalies or inconsistencies within the call type descriptions.

Recognizing the limitations of the initial feature set for robust model construction, we embarked on a comprehensive data enrichment process. The first step involved the enhancement of the primary dataset with precise postal codes for each address. This was accomplished by leveraging a geocoding API, which allows for the conversion of geographic coordinates into postal codes. The geocoding API, which translates geographic coordinates into postal codes, was critical for this enhancement. The function `get_postcode` was used to retrieve postal codes from latitude and longitude data, with a throttle to ensure compliance with usage policies. This addition provided a vital categorical variable to the dataset, bolstering its utility for further analysis and modeling.

In addition to this, we incorporated demographic data, hypothesizing that such information could offer valuable insights for our classification efforts. This demographic dataset, rich in population statistics associated with postal codes, was seamlessly integrated into our main dataset through an inner join operation, ensuring consistency by the shared postal code attribute.

Further bolstering our feature set, we integrated a third dataset detailing the nearest fire station locations, utilizing postal codes as the relational key. This dataset (`data2`) contained information on the spatial coordinates (`X_station`, `Y_station`), address (`ADDR_station`), city, state (`STATE`), the jurisdiction (`JURIS`), and the unique operational number (`OC_NUM`) for each fire station.

The primary goal of this project is to generate synthetic images from real-world data, so merging various datasets helps create a more realistic context for the scenarios we're trying to model. By incorporating the fire station locations, we add depth to the scene-generation process, ensuring that our synthetic data closely mimics real-life situations.

Additionally, by combining multiple datasets, we can examine the relationships between different features. Understanding these connections can lead to insights that im-

prove the accuracy and reliability of our generated scenarios. This step is crucial for developing robust models and simulations that reflect the complexity of real-world emergency responses.

Thus, the original dataset was enriched with two additional datasets via inner joins, predicated on the postal codes obtained through the geocoding API. The new features—postal code, demographic profiles, and proximity to emergency services—were assimilated to provide a multidimensional view of the emergency calls, enabling a more nuanced analysis and modeling.

```
#API for postcodes
def get_postcode(lat, lon):
    time.sleep(1)  # To avoid overloading the server
    url = f"https://nominatim.openstreetmap.org/reverse?format=json
    &lat={lat}&lon={lon}"
    response = requests.get(url)
    if response.status_code == 200:
        content = response.json()
        if 'address' in content and 'postcode' in content['address']:
            return content['address']['postcode']
    return None
```

The resulting enriched dataset now stands as a testament to the power of data integration, promising a more informed and effective classification model.

3.1.1 Missing Values

During the preprocessing phase, dealing with missing values is crucial to maintain the integrity of the dataset and ensure accurate modeling. To address this, we employed a statistical method using the mode, which represents the most common value within a particular column.

The Python function `fill_with_mode` leverages the `mode` function from the `scipy.stats` module to ascertain the most frequently occurring value, or mode, in a specified column.

Once the mode is determined, the function replaces all missing (NaN) entries in that column with the mode value. This method is particularly effective for categorical data where the mode represents the most typical category.

We applied this function to the 'Day' and 'Month' columns of our dataset, which are critical temporal features for our analysis. By filling the missing values with the mode, we preserve the distributional characteristics of our dataset and avoid the introduction of bias that could result from other imputation methods, such as mean or median replacement, which are less suitable for categorical data.

```
from scipy.stats import mode

# Function to fill missing values with mode
def fill_with_mode(df, column_name):
    mode_value = df[column_name].mode()[0]
    df[column_name].fillna(mode_value, inplace=True)

# Fill missing values in 'Day' and 'Month' with their respective modes
fill_with_mode(data, 'Day')
fill_with_mode(data, 'Month')
```

This approach ensures that the temporal aspect of the dataset remains as accurate as possible, reflecting the true nature of the emergency call occurrences.

3.1.2 Preprocceing the type of features

In the preprocessing phase of our project, we performed several critical operations on our dataset to prepare it for further analysis. A key aspect of this preparation involved the manipulation of date and time data, as well as the conversion of coordinate data into a suitable numeric format.

```
from datetime import datetime
```

```
# Load your data
# Replace 'data.csv' with the path to your dataset
data = pd.read_csv('data.csv')

# Convert the DATE and TIME columns to datetime objects
data['DATE'] = pd.to_datetime(data['DATE'], format='%Y-%m-%d')
data['TIME'] = pd.to_datetime(data['TIME']).dt.time

# Combine the DATE and TIME columns to create a new DateTime column
data['DateTime'] = data.apply(lambda row: datetime.combine(row
['DATE'], row['TIME']), axis=1)

# Convert XCOORD and YCOORD columns to numeric type
data['XCOORD'] = pd.to_numeric(data['XCOORD'], errors='coerce')
data['YCOORD'] = pd.to_numeric(data['YCOORD'], errors='coerce')

# Check for missing data and remove rows with missing data
print(data.isna().sum())
data = data.dropna()

# Extract day, month, year, hour, and minute information from the
DateTime column
data['Day'] = data['DateTime'].dt.day
data['Month'] = data['DateTime'].dt.month
data['Year'] = data['DateTime'].dt.year
data['Hour'] = data['DateTime'].dt.hour
data['Minute'] = data['DateTime'].dt.minute
```

The process involved several steps:

1. **Data Loading:** The dataset, stored in a CSV file, is loaded into a DataFrame

using Pandas.

2. **Date and Time Conversion:** The 'DATE' and 'TIME' columns, initially in string format, are converted into Python datetime objects for more efficient manipulation.

3. **Combining Date and Time:** A new column, 'DateTime', is created by combining the 'DATE' and 'TIME' columns. This unified column facilitates temporal analysis by providing precise timestamps.

4. **Coordinate Conversion:** The 'XCOORD' and 'YCOORD' columns, representing geographical coordinates, are converted from strings to numeric values. This conversion is necessary for any spatial analysis that might be conducted.

5. **Handling Missing Data:** We conducted a check for missing values in the dataset and removed rows containing any missing data, ensuring the cleanliness and reliability of our dataset.

6. **Extraction of Temporal Features:** From the 'DateTime' column, detailed temporal information such as day, month, year, hour, and minute is extracted, allowing for a granular temporal analysis of the emergency calls.

These preprocessing steps were essential in transforming the raw data into a structured format, ripe for analysis and modeling.

3.1.3 Outliers

In our data analysis, we conducted two significant statistical investigations to understand the temporal distribution of emergency calls.

Firstly, we examined the dates with the highest number of calls:

```
# Group by 'DATE' and count the occurrences
top_dates = data.groupby('DATE').size().reset_index(name='Count')

# Sort by 'Count' in descending order and get the top 10
top_dates = top_dates.sort_values(by='Count', ascending=False).head(10)
```

In this analysis, the dataset was grouped by the 'DATE' column, and the occurrences of calls on each date were counted. We then sorted these dates in descending order based

on the number of calls and extracted the top 10 dates. This provided us with insights into specific days with unusually high emergency call volumes.

Secondly, we analyzed the dataset to identify outliers in terms of call volumes:

```
data_grouped = data.groupby('DATE').size().reset_index(name='num_calls')

# Calculate the first and third quartiles (Q1 and Q3)
Q1 = data_grouped['num_calls'].quantile(0.25)
Q3 = data_grouped['num_calls'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = data_grouped[(data_grouped['num_calls'] < lower_bound) |
                        (data_grouped['num_calls'] > upper_bound)]

print(outliers)
```

This segment of the analysis focused on identifying days with an atypical number of calls. By calculating the interquartile range (IQR) and defining bounds for outliers, we could pinpoint days where the number of calls significantly deviated from the norm. This helped in identifying anomalies or potential areas of concern that may need further investigation. In our analysis, we conducted a detailed exploration of the frequency of emergency calls in relation to postal codes. This exploration aimed to identify postal codes with high emergency call volumes and detect any outliers that significantly deviate from the typical call frequency pattern.

Analysis of Top Postal Codes by Call Frequency:

```
# Group by 'postcode' and count the occurrences
top_postcodes = data.groupby('postcode').size().reset_index(name='Count')

# Sort by 'Count' in descending order and get the top 10
top_postcodes = top_postcodes.sort_values(by='Count', ascending=False).head(10)

print(top_postcodes)
```

In this segment, the dataset was grouped by the 'postcode' column, and the number of calls for each postal code was calculated. These counts were then sorted in descending order to identify the top 10 postcodes with the highest call volumes. This analysis helped pinpoint areas with potentially higher needs for emergency services.

Identification of Outliers in Call Frequencies by Postal Code:

```
# Group by 'postcode' and count the occurrences
data_grouped2 = data.groupby('postcode').size().reset_index(name='num_calls')

# Calculate the first and third quartiles (Q1 and Q3)
Q1 = data_grouped2['num_calls'].quantile(0.25)
Q3 = data_grouped2['num_calls'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
```

```
outliers = data_grouped2[(data_grouped2['num_calls'] < lower_bound) |
                         (data_grouped2['num_calls'] > upper_bound)]

print(outliers)
```

This analysis focused on detecting outliers in the number of calls per postcode. Using the interquartile range (IQR) and quartiles, we defined boundaries to identify postal codes with unusually high or low call frequencies. This approach is instrumental in highlighting postal codes that may require special attention or are behaving anomalously compared to the majority.

Both these analytical approaches are vital for understanding the spatial distribution of emergency calls, which is crucial for effective resource allocation and planning in emergency response strategies.

3.1.3.1 Handling of Outliers

During our analysis, a deeper examination of the outliers, particularly with respect to dates, revealed a significant insight. The outlier dates coincided with the occurrence of Hurricane Ian, a major event that naturally led to an abnormal spike in emergency calls. To maintain the integrity of our analysis and avoid skewing the data with this anomaly, we made the decision to exclude the dates pertaining to Hurricane Ian from our dataset.

Additionally, our investigation into postal code-related outliers identified a single postcode with a disproportionately high number of emergency calls. Recognizing that outright removal of this postcode could lead to the loss of valuable information, we instead opted for a more nuanced approach. We reduced the number of entries for this particular postcode by a small margin, akin to an undersampling technique. This adjustment aimed to mitigate the impact of this outlier on our overall analysis without entirely eliminating the valuable data it contained.

These strategic decisions in handling outliers were crucial to ensure that our analysis accurately reflected typical emergency call patterns, thereby providing more reliable insights for emergency response planning and resource allocation.

3.1.4 Feature Engineering

To enhance the predictive capability of our models, we conducted extensive feature engineering on the dataset. This process involved creating new, meaningful features from the existing data, which provided deeper insights and improved the model's accuracy.

1. **Distance from City Center:** We calculated the Manhattan distance of each emergency call location from a predefined city center point. This was achieved by converting the X and Y coordinate columns to numeric types and then applying the Manhattan distance formula. The Manhattan distance, also known as the taxicab or city block distance, is calculated as the sum of the absolute differences of their Cartesian coordinates; specifically, it is $|x_1 - x_2| + |y_1 - y_2|$ where (x_1, y_1) and (x_2, y_2) are the Cartesian coordinates of the emergency call location and the city center, respectively.

2. **Categorization of Call Types:** We categorized the types of emergency calls into distinct groups such as 'Ambulance', 'Fire Brigade', 'Police', 'Rescue Vehicles', and 'Combined Services'. This categorization was based on the nature of the emergency as indicated in the 'Call TYPE' column.

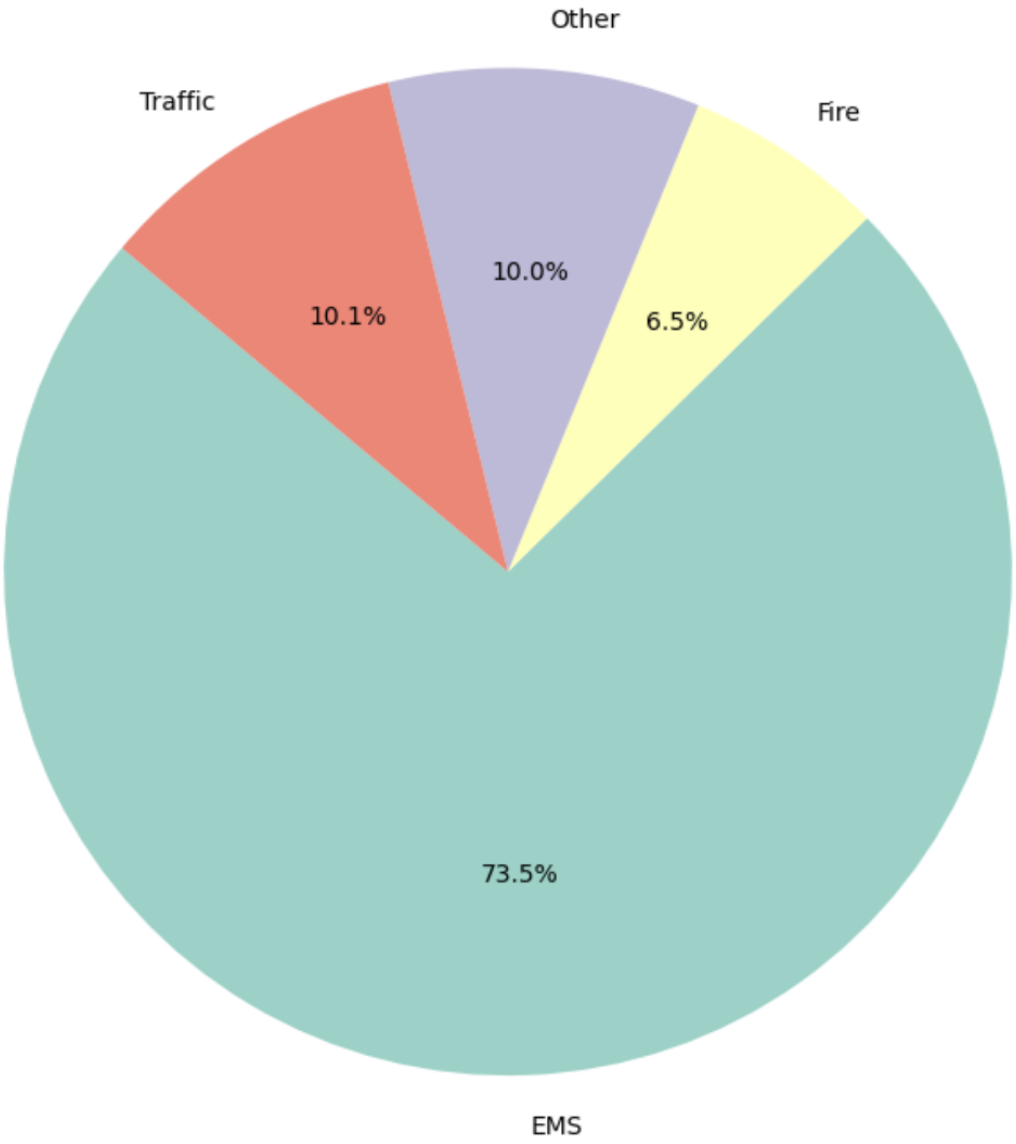

Figure 3.3: Distribution of emergency calls by category after Categorization.

3. **Further Categorization of Call Descriptions:** We applied a similar categorization process to the 'CALL TYPE DESC' column, grouping the calls into 'EMS', 'Traffic', 'Fire', and 'Other'. This classification relied on the detailed descriptions provided and was aimed at refining the categorization process.

4. **Time of Day and Season Extraction:** The dataset was further enriched by deriving the 'TimeOfDay' and 'Season' from the 'Hour' and 'Month' columns, respectively. The 'TimeOfDay' was categorized into 'Night', 'Morning', 'Day', and 'Evening', while the 'Season' was classified into 'Winter', 'Spring', 'Summer', and 'Fall'. These features were expected to reveal temporal patterns in emergency call

frequencies.

These newly engineered features significantly enhanced our dataset, providing a multifaceted view of the emergency calls. This enriched dataset was expected to offer a more accurate and comprehensive basis for our subsequent data modeling and analysis efforts.

3.1.5 Seasonality and Distribution Analysis

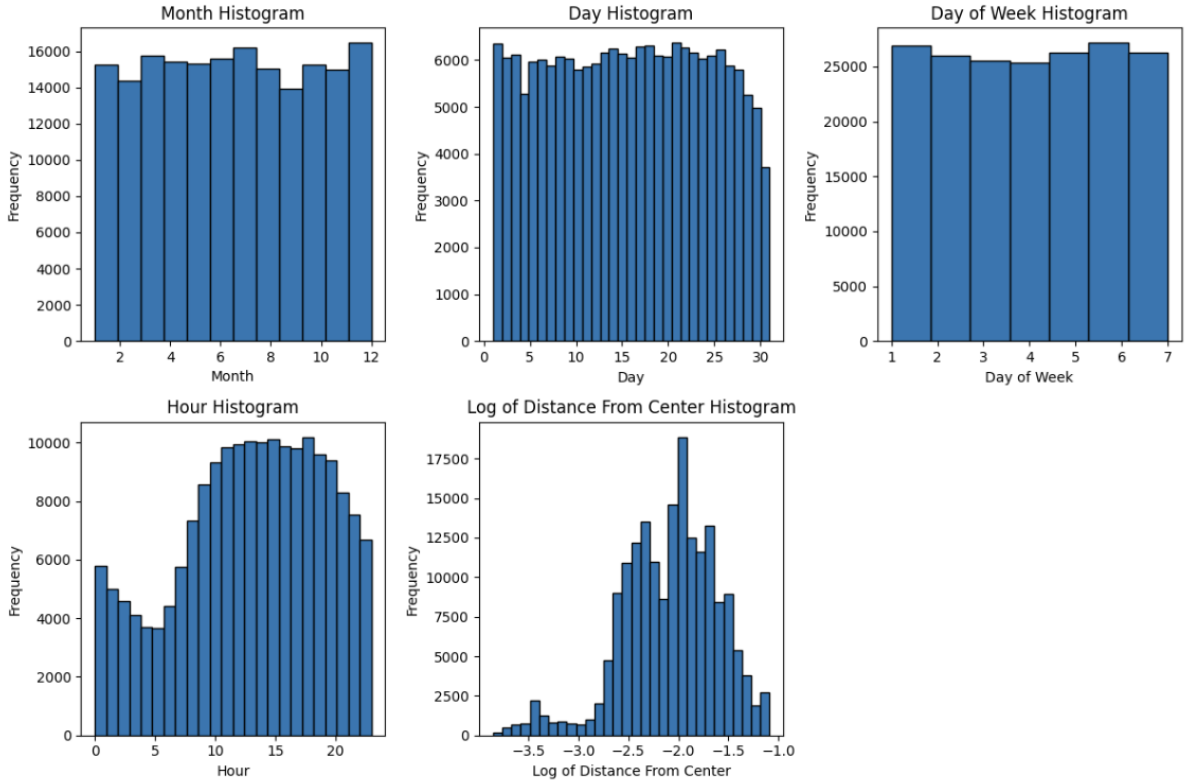

Figure 3.4: Histograms

We assessed the seasonality and distribution of emergency calls through a series of histograms representing various temporal aspects.

- The **Month Histogram** suggests a moderate seasonality with a slight uptick in calls during the middle of the year, hinting at possible influences of seasonal factors.
- The **Day Histogram** displays a consistent call distribution across the month, indicating the absence of day-specific trends in the frequency of calls.

- In the **Day of Week Histogram**, the distribution of calls is relatively even, suggesting that emergency incidents are fairly distributed throughout the week.
- The **Hour Histogram** shows a pronounced diurnal trend, with calls increasing through the day and peaking in the evening, then declining during the night hours.
- The **Log of Distance From Center Histogram** reveals a concentrated frequency of calls at certain distances from the city center, likely reflecting the denser areas of population and activity.

These visualizations are crucial for understanding the temporal dynamics of emergency calls and guiding the allocation of resources and services.

3.1.6 Feature Engineering and Model Tranings for Feature Importance

In the pursuit of understanding the influence of various factors on performance outcomes, I chose to utilize the CatBoost classifier due to its exceptional capability in handling and analyzing complex datasets, particularly those rich in categorical data. The motivation behind employing this advanced machine learning algorithm was to meticulously dissect and discern the relative importance and impact of diverse variables such as time (e.g., specific months), geographical location (e.g., distance from a central point), and environmental conditions (e.g., weather patterns) on the performance metrics in question.

The CatBoost classifier, renowned for its proficiency with categorical data, allows for a nuanced exploration of how each factor contributes to the overall outcomes. For instance, by examining whether certain months bear more significance than others, or assessing the role of distance and weather conditions, I aimed to unveil underlying patterns and correlations that might otherwise remain obscured. This methodological approach facilitated a deeper understanding of the dynamics at play, enabling the identification of key drivers of performance and areas where targeted interventions could yield substantial improvements.

Furthermore, the use of CatBoost's feature importance tool provided an invaluable lens through which to evaluate the contribution of each variable to the model's predictive accuracy. This analysis is crucial for prioritizing resources and strategies in scenarios where enhancing performance is paramount. In essence, the application of the CatBoost classifier served not only to illuminate the effects of various factors on performance but also to inform decision-making processes with data-driven insights, underscoring the algorithm's robustness and versatility in analytical endeavors.

```
    import pandas as pd
from catboost import CatBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
selected_features = ['Month', 'Day', 'Hour', 'postcode']
X = data[selected_features]
y = data['Category']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_stat
cat_features = [i for i, col in enumerate(X_train.columns) if X_train[col].dtype == '
catboost_model = CatBoostClassifier(
    iterations=1000,
    learning_rate=0.1,
    depth=6,
    cat_features=cat_features,
    verbose=200,
    random_state=123
)
catboost_model.fit(X_train, y_train)
y_pred = catboost_model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

In this code snippet, we utilize the CatBoost model to identify and visualize the

importance of various features in our dataset. Firstly, we extract the feature importances from our trained CatBoost model using the get future importance method. This method returns a list of importance scores, where each score corresponds to a feature in our dataset, indicating the relative importance of each feature in the model's predictions.

Next, we set up a matplotlib figure with a specified size of 12x6 inches to provide a clear and spacious visualization. We then create a bar chart using plt.bar(), where the x-axis represents the indices of our features (ranging from 0 to the number of features), and the y-axis represents the importance scores of these features.

3.1.7 Seasonality and Distribution Analysis

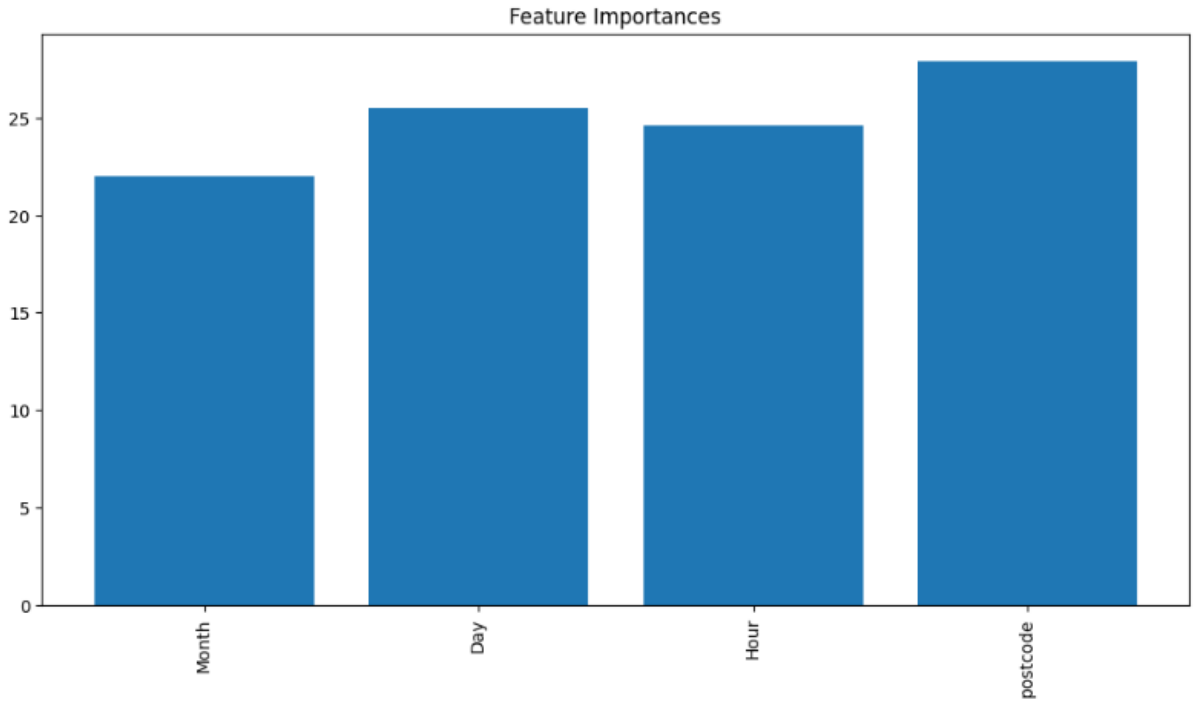

Figure 3.5: Feature Importance 1

3.1.8 Seasonality and Distribution Analysis

In conclusion, our comprehensive analysis utilizing the CatBoost classifier has led us to ascertain that proximity to the city center is a significant determinant of performance outcomes. This finding underscores the pivotal role that geographical location plays in influencing the metrics under consideration. The nuanced examination facilitated by the

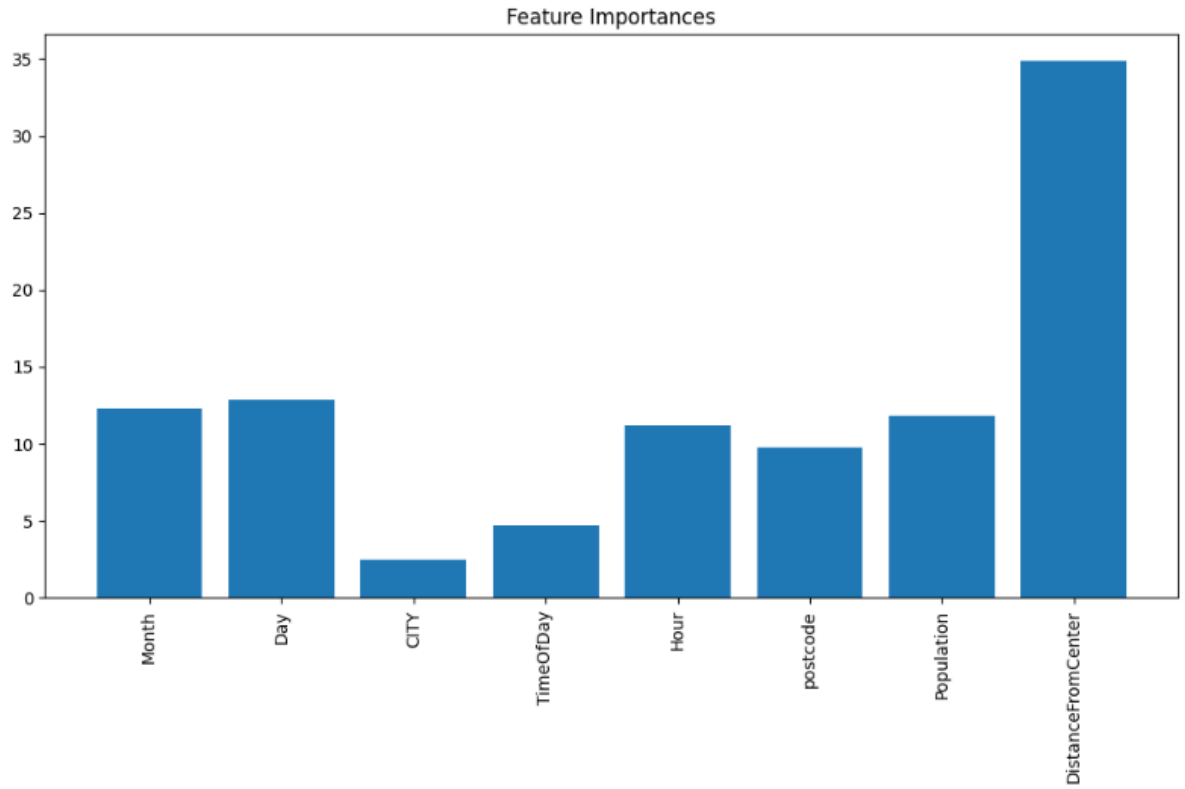

Figure 3.6: Feature Importance 2

CatBoost's feature importance tool has enabled us to pinpoint the centrality of location as a key factor, surpassing other variables we initially hypothesized to have considerable impact, such as specific time periods and environmental conditions.

However, it's also crucial to acknowledge a notable limitation encountered in our dataset - the underrepresentation of data from rural areas. This scarcity has implications for the breadth and applicability of our findings, particularly in extending our analysis to scenarios beyond urban contexts. After careful deliberation and in light of the data imbalance, we have decided to refrain from drawing extensive conclusions regarding environmental factors. This decision stems from a commitment to ensuring the reliability and validity of our insights, acknowledging that the current dataset does not furnish a comprehensive view of the rural landscape.

Moving forward, it's imperative that future investigations seek to incorporate a more balanced representation of both urban and rural data. This would not only enrich the robustness of the analysis but also enable a more holistic understanding of the factors affecting performance across diverse geographical settings. Our study highlights the im-

portance of geographical proximity in urban contexts, yet it also opens avenues for further research into the multifaceted interactions between location, environmental conditions, and other variables in shaping outcomes across different settings.

Chapter 4

Reverse Engineering

4.1 Diffusion Models

In our thesis, we made a key decision to use diffusion models for synthetic image generation, motivated by their remarkable ability to create realistic and diverse images from textual descriptions. This choice was driven by the need to produce a wide range of images that accurately represented the detailed requirements of our research, where conventional data collection methods were either inadequate or impractical.

To harness the full potential of diffusion models, we conducted an in-depth study of their underlying architecture and operational mechanisms. Diffusion models represent a significant advancement in generative AI technology, bridging the fields of natural language processing (NLP) and computer vision to interpret and visualize complex textual inputs as detailed images.

The core of diffusion models' architecture is a variant of the Transformer model, originally made famous for its outstanding performance in NLP tasks. The Transformer's ability to process sequences of data and capture long-range relationships makes it a perfect fit for diffusion models, allowing them to understand and contextualize textual descriptions with impressive accuracy. Moreover, diffusion models use a modified version of a large language model to process textual inputs, which guides the process of image generation.

The innovative aspect of diffusion models also lies in a technique called VQ-VAE-2 (Vector Quantized-Variational AutoEncoder 2), which aids in compressing and decompressing images into a more manageable size. This process enables the creation of high-quality images that closely align with the given textual descriptions, effectively bridging the gap between textual and visual data.

By utilizing the architecture of diffusion models, we aimed to generate synthetic images that could significantly improve the quality and variety of data for our thesis. The capacity to produce custom images tailored to specific research needs opened new possibilities for exploration and analysis, enabling the investigation of scenarios that would be difficult to replicate in the real world.

In summary, our decision to use diffusion models was strategic, driven by the goal to push the boundaries of synthetic image generation in academic research. By delving into the architecture and capabilities of diffusion models, we not only enriched the dataset for our thesis, but also contributed to a deeper understanding of how AI-driven image synthesis can be used in various research fields.

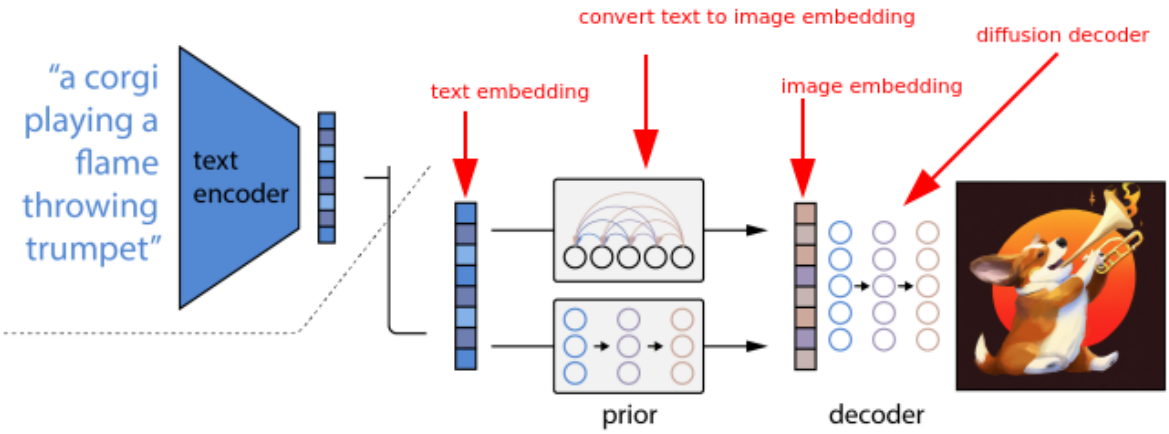

Figure 4.1: Diffusion Model Architecture

4.1.1 Text Description

In a dedicated subsection of our thesis, we delve into the reverse engineering process of utilizing large language models to describe synthetic images in textual form. This

investigative approach was predicated on our strategic decision to employ diffusion models for the generation of synthetic scene images within our research. Understanding how diffusion models interpret images as textual descriptions was imperative, as it directly informs our methodology for synthesizing scene images that are coherent and contextually relevant to our study's objectives.

The rationale behind this reverse engineering effort stems from the necessity to bridge the gap between the visual and textual representations of data. By inputting synthetic images into a large language model and analyzing its textual descriptions, we aimed to decipher the underlying patterns, themes, and characteristics that diffusion models prioritize when converting textual prompts into images. This process involved feeding a large language model a series of synthetic images generated by us and meticulously documenting how it articulated these images in text, identifying key descriptors, attributes, and nuances it associated with each image.

This reverse engineering task was crucial for several reasons. Firstly, it provided us with valuable insights into the interpretative mechanisms of diffusion models, allowing us to tailor our textual prompts more effectively to generate desired images. Secondly, by understanding the textual language that large language models use to describe images, we were able to refine our approach to synthetic image generation, ensuring that the resulting images align closely with the intended themes and elements of our research.

Ultimately, this subsection emphasizes the interdisciplinary nature of our research, bridging AI's capabilities in understanding and generating both textual and visual content. By reverse engineering the process of describing synthetic images using large language models, we gained a deeper comprehension of the conversational models' perceptual frameworks, which, in turn, significantly enhanced the precision and relevancy of the synthetic scenes generated using diffusion models for our study. This methodology underscores the iterative and explorative nature of working with AI tools in research, where understanding the input-output dynamics of one tool (large language models) can profoundly influence the effective utilization of another (diffusion models).

4.1.2 Deciphering Diffusion Models: A Test of Image Understanding

In this chapter, we delve into the intricate workings of diffusion models' image perception and descriptive capabilities, unveiling the nuances of how this advanced AI interprets visual data. Diffusion models, renowned for their ability to generate images from textual descriptions, also possess the intriguing capability to "see" and describe images with a level of nuance that approaches human-like understanding. To explore this, we present two real-life scenarios that demonstrate the interpretative skills of diffusion models. The first scenario involves an image of a bustling urban street scene, replete with the vibrancy of city life—a test of the diffusion model's ability to identify and articulate the complexity of everyday urban existence. The second scenario is a contrasting image of serenity: a lone oak tree in a sun-drenched meadow, which probes the diffusion model's capacity to capture and convey the tranquil essence of nature. These scenarios were carefully selected not only to test the range of the diffusion model's image comprehension but also to assess its proficiency in switching contexts and encapsulating varied thematic elements within its descriptions. This analysis not only provides insight into the current state of AI in image recognition and description but also propels us to consider the potential applications and implications of such technology in our increasingly visual world.

4.1.2.1 Test-1

- **Primary Scene:** Intersection in a suburban area, potentially in Florida given the greenery and light.
- **Time:** Daytime, as evidenced by the bright sky and shadows cast on the road.
- **Weather:** Clear, with no signs of precipitation or overcast conditions.
- **Foreground:**
  - Interior of a vehicle, likely captured through a dashboard camera.
  - Dashboard is partially visible at the bottom of the frame.

Figure 4.2: Diffusion Model Understanding Test-1

  - Windshield provides a clear view to the outside scene.

- **Midground:** Street View
  - Road: Paved, with traffic lanes and road markings clearly visible.
  - Vehicles:
    * Police Car: Positioned on the road, bearing typical blue and white police color scheme and markings.
  - Surroundings: Suburban, with manicured grass and traffic infrastructure.
- **Background:** Suburban Environment
  - Vegetation: Mature trees lining the street.
  - Sky: Daylight with some clouds, but predominantly clear.
- **Relationships:**
  - The viewer (through the camera's perspective) is stationary behind the police car.
  - The police car is ahead on the road, positioned across traffic lanes, possibly indicating an active response to an incident or an emergency situation.

– The roadside vegetation and traffic lights provide a context of a regulated traffic environment within a suburban setting.

4.1.2.2 Test-2

Figure 4.3: Diffusion Model Understanding Test-2

- **Primary Scene:** Rural or semi-rural road, likely in a region similar to Florida given the dense vegetation and Spanish moss.
- **Time:** Daytime, as indicated by the natural light and shadows visible on the road.
- **Weather:** Overcast or partly cloudy, with diffused lighting and no direct sunlight observed.
- **Foreground:** Inside of a vehicle, possibly a car.
  - The dashboard is in clear view, occupying the lower portion of the image.
  - The windshield serves as a transparent barrier, offering an unobstructed view of the exterior.
- **Midground:** Roadway

  - The road is paved and appears to be a two-lane rural road with double yellow lines, indicating no passing.
  - Vehicles:
    * Police Car: Parked or moving slowly on the right side of the road, featuring the standard black and white color scheme with identifiable police markings.
  - Surroundings: Lush, with a natural and less manicured look compared to urban areas.

- **Background:** Rural Landscape
  - Vegetation: Dense greenery, trees draped with Spanish moss, and wild underbrush.
  - Sky: Visible through the canopy, appears mostly cloudy with soft light filtering through.

- **Relationships:**
  - The perspective is from inside a following car, capturing the scene through the windshield.
  - The positioning of the police car suggests it may have pulled over or is attending to a matter off the road.
  - The surrounding foliage and road setup provide a sense of seclusion, typical of less populated areas.

4.1.3 Results

Through a detailed analysis of the results from our proprietary dataset, we've been able to identify with considerable precision the types of emergency vehicles involved, the traffic conditions, the weather status, and the time of day. These insights have helped us define key inputs for our language model, which is set to accurately interpret and contextualize a variety of scenarios encountered on the road.

In the current stage of our research, we've chosen to exclude road type information from the model's inputs. This decision is based on our analysis, which shows that most 911 emergency calls are made near urban centers. As a result, the likelihood of calls from rural areas is relatively low. Additionally, given the typical challenges faced by autonomous vehicles in city settings, we've decided that our model's distribution of road types should follow a normal distribution for now.

This strategic choice allows us to streamline the model's focus, reducing complexity without compromising its ability to predict accurately. By incorporating emergency vehicle types, traffic and weather conditions, along with time-of-day variables, we're equipping the model to perform with improved relevance and precision in the complex environment of urban transportation.

It's important to note that this decision isn't set in stone but reflects our iterative research process. As autonomous vehicles become more capable of handling a broader range of terrains, and as our dataset expands to include more varied scenarios, we may adjust the model's parameters to encompass a wider array of road types.

In summary, our current approach to defining the language model's inputs reflects a careful balance between empirical evidence and practical considerations, designed to drive our research forward in a way that is methodologically sound and in line with real-world applications.

Chapter 5

Language Models for Text Generation

In the realm of text generation, the efficacy of a model is predominantly dictated by the richness and diversity of its training dataset. Given the limited scope of input data initially available for my large language model, manual augmentation of input-output pairs was deemed inefficient and time-consuming. To circumvent this constraint and enrich the training dataset, the Anyway Conditional Tabular Generative Adversarial Network (ACTGAN) was employed. The Gretel ACTGAN model API was leveraged to access a generative data model specifically designed for tabular data. This innovative approach supports advanced functionalities, including conditional data generation, proving particularly adept at handling datasets characterized by a predominance of numeric data, extensive column counts, and highly unique categorical fields.

ACTGAN's utility in enhancing the training dataset for large language models cannot be overstated. However, it is imperative to acknowledge that this model is trained exclusively on the examples provided in the training dataset. Consequently, it inherently captures and is likely to perpetuate any biases present within the training set. To mitigate this, and ensure the augmented dataset's suitability for the large language model, a meticulous manual review and re-labeling process was undertaken. This step was critical to refining the dataset, ensuring the generated text is not only diverse and rich but also

devoid of inherent biases that could compromise the model's objectivity and fairness.

By integrating ACTGAN's capabilities with the large language model, this study has significantly advanced the frontier in text generation. The augmented dataset has not only expanded the model's exposure to varied linguistic structures and semantics but has also enhanced its ability to generate coherent and contextually relevant text. This synthesis of generative adversarial networks with large language models presents a novel methodology in the field of natural language processing, demonstrating the potential for hybrid approaches to overcome limitations inherent in traditional model training processes.

The generation of synthetic data was thus a strategic move, designed to bridge any gaps in the dataset that could potentially hinder the model's performance. By augmenting the dataset with synthetic entries, we aimed to create a comprehensive training environment that would enable large language models to capture the nuances and varied nature of emergency calls, ultimately leading to more accurate and reliable text generation that reflects the critical information conveyed during 911 emergencies.

Here is an example:

Input

- Night
- Rain, moderate rain
- Ambulance, Police Car
- Normal Traffic

Output

- **Primary Scene:** An urban street during a rainy night in a simulated environment, hinting at a driving or emergency response simulation.
- **Time:** Night within the simulation, as indicated by the simulated darkness and the artificial lights reflecting on wet surfaces.
- **Weather:** Simulated moderate rainfall, with effects such as glistening roads and

raindrops visible in the streetlights' glow.

- **Interior:** The interior of a simulated emergency vehicle, likely an ambulance or police car, with a focus on the view through the windshield, providing a first-person perspective of the rainy street ahead.
- **Road:** The road is wet and reflective, with traffic lanes visibly marked and puddles forming at the pedestrian crossings.
- **Vehicles:** Various simulated cars, including an ambulance and a police car, are depicted with their emergency lights on, navigating through the normal traffic that is moving cautiously due to the rain.
- **Buildings:** The buildings that line the street are typical urban structures, their windows shimmering with the reflection of rain and street lights.
- **Sky:** The sky is dark, filled with rain clouds, enhancing the nocturnal urban atmosphere of the simulation.
- **Relationships:** The arrangement of the vehicles and the simulation's weather conditions create a dynamic scene where the emergency services are actively responding within a typical traffic pattern, providing a realistic depiction of urban operations during inclement weather.

5.1 Models for Improving Dataset

5.1.1 ACTGAN

The network structure of the CTGAN (Xu et al., 2019) model is designed to capture all possible correlations between columns in tabular data. It utilizes fully-connected networks in both the generator and critic components. The structure of the CTGAN network is meticulously crafted to encapsulate the potential correlations among the columns of tabular data. The model employs fully-connected layers in both the generator and critic to ensure comprehensive feature integration.

The generator within CTGAN is composed of dual fully-connected hidden layers that utilize batch normalization and ReLU activation functions to foster the generation of

synthetic row representations. This synthesis is facilitated through a diverse set of activation functions: tanh for scalar values, and gumbel softmax for both mode indicators and discrete values.

The critic, pivotal in assessing the fidelity of the synthetic data, incorporates the leaky ReLU function and implements dropout across its hidden layers to enhance model robustness. Furthermore, CTGAN adopts the PacGAN framework, specifically deploying 10 samples per pac to avert the issue of mode collapse.

The architecture underpinning the conditional generator and critic is delineated with precision in the referenced document, explicating the intricate operations and transformations pivotal for synthetic data generation and the subsequent evaluation by the critic. The discourse encompasses an array of innovative techniques such as mode-specific normalization, conditional generator, and training-by-sampling—all integral in addressing the complexities of tabular data synthesis.

The key components of the CTGAN model include:

Mode-Specific Normalization: Utilizing a variational Gaussian mixture model, CTGAN normalizes continuous columns, thus translating continuous values across varying ranges into a bounded vector form, aptly suited for neural network processing.

Conditional Generator and Training-by-Sampling: To tackle the imbalance present in discrete columns, CTGAN integrates a conditional generator and a training-by-sampling methodology. The conditional generator is particularly advantageous for synthesizing data corresponding to specific discrete values, an asset for data augmentation. Training-by-sampling proves indispensable in managing datasets characterized by significant imbalances, such as those found in credit-related data.

Network Architecture: The utilization of fully-connected networks within the generator and critic is key to capturing inter-column correlations. The generator benefits from batch normalization and ReLU activations, whereas the critic is fortified by leaky ReLU functions and dropout techniques.

Collectively, these components empower CTGAN to adeptly model complex column distributions and produce synthetic tabular data that is both realistic and highly repre-

sentative of the original datasets.

Given two discrete columns D_1 and D_2, the process begins by selecting a column, say D_2, and a category within that column, say category 1. Then, a row from the training data T_{train} with $D_2 = 1$ is picked.

A noise vector z is sampled from a multivariate normal distribution, $z \sim \mathcal{N}(0, I)$, and is fed to the generator $G(\cdot)$ along with the selected category encoded as a one-hot vector. For example, if D_2 is selected and category 1 is chosen, the one-hot vector would be $[0, 1]$ where the second position corresponds to category 1. This vector is concatenated with z.

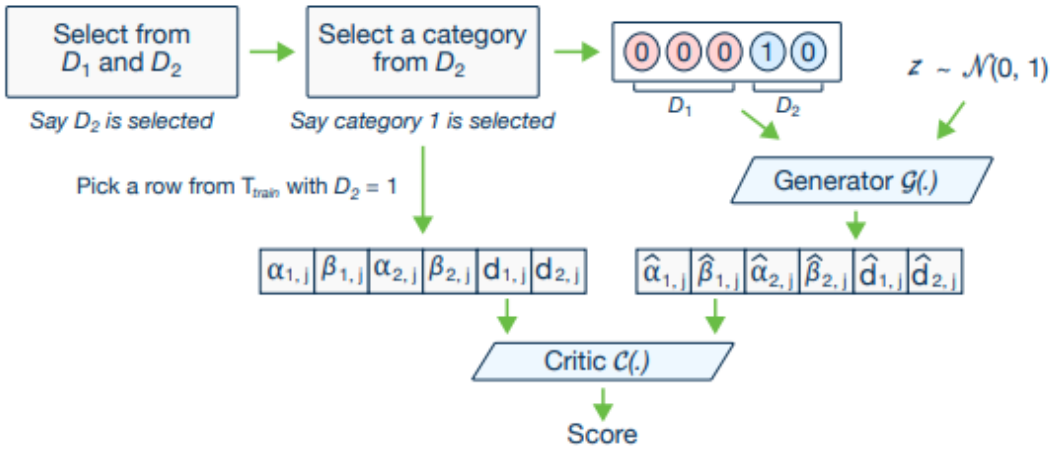

Figure 5.1: CTGAN Algorithm (Xu et al., 2019)

The generator $G(\cdot)$ then produces a synthetic row $\hat{r} = \{\hat{\alpha}_1, \hat{\beta}_1, \ldots, \hat{\alpha}_2, \hat{\beta}_2, \hat{d}_1, \hat{d}_2\}$, which is a vector that mimics the real data's structure. Here, $\hat{\alpha}_i$ represents the synthetic continuous values and $\hat{\beta}_i$ are the mode indicators for continuous columns, while $\hat{d}_i$ are the synthetic discrete values.

This synthetic row is then passed to the critic $C(\cdot)$, which evaluates the 'realness' of the generated data and produces a score reflecting how well the generator is performing. The score is used to update the generator and critic during training, with the objective of improving the generator's ability to create realistic synthetic data.

The CTGAN model confronts the complexities of simulating tabular datasets with a series of inventive solutions. Central to its design is the mode-specific normalization technique, which adeptly manages the intricate distributions of continuous data, trans-

forming continuous variables into a normalized vector format that neural networks can process efficiently. Moreover, CTGAN incorporates a conditional generator combined with a training-by-sampling strategy to tackle the prevalent issue of data imbalance, particularly within discrete columns. This approach is instrumental in generating synthetic data for discrete values and maintaining equilibrium across disparate datasets. CTGAN leverages fully-connected networks, along with contemporary training methods, to cultivate a model capable of accurately capturing and reproducing the intricate distributions inherent in tabular data.

In my thesis, ACTGAN, an abbreviation for Anyway Conditional Tabular Generative Adversarial Network, emerges as a sophisticated tool engineered to generate synthetic data at scale, adept at processing the high-dimensional datasets characteristic of industries like advertising, cybersecurity, finance, and life sciences. As an evolution of CTGAN, ACTGAN is designed to address and surmount scalability challenges presented by larger datasets.

ACTGAN stands out with its several pivotal enhancements. It markedly diminishes memory consumption on both CPU and GPU fronts, thus streamlining the training workflow and facilitating the management of extensive datasets without resorting to high-memory GPUs or enduring protracted training periods. This achievement stems from the integration of a novel Binary Encoder that functions in concert with conventional one-hot encoding, optimizing the internal data representation for efficiency.

Additionally, ACTGAN streamlines the identification and transformation of datetime fields, adopting a novel sampling approach that reflects the actual distributions found in the original data, rather than categorically treating such variables. This feature assures that the synthetic data generated bears a closer resemblance to the original data's temporal trends.

A critical feature of ACTGAN is the enhancement of its conditional vector sampling mechanism, significantly heightening the precision of conditional data generation. This is particularly beneficial for creating accurately labeled instances within machine learning datasets, thereby refining the utility and relevance of the synthetic data.

5.1.2 Implementation

In our recent implementation, we employed the Gretel ACTGAN model, a variant of the Generative Adversarial Network (GAN) framework specifically tailored for generating synthetic tabular data that closely mimics authentic datasets. The ACTGAN model is designed to operate with discrete and continuous variables, handling complex multi-modal distributions and relationships within the data.

The configuration of the model was meticulously set to optimize its performance. The `embedding_dim` was established at 128 to allow for a robust representation of the data. The `generator_dim` and `discriminator_dim` were both set to two layers of 1024 dimensions, providing the model with the necessary capacity to generate and discriminate data effectively. The learning rates for the generator and discriminator were configured at 0.0001 and 0.00033, respectively, with a slight decay of 0.000001 to fine-tune the training process over time. The batch size was set to `auto`, allowing the model to determine the optimal number dynamically based on the dataset size.

5.1.3 Model Configuration Parameters

[Python code for Gretel ACTGAN model configuration]

```
embedding_dim: 128
generator_dim:
  - 1024
  - 1024
discriminator_dim:
  - 1024
  - 1024
generator_lr: 0.0001
generator_decay: 0.000001
discriminator_lr: 0.00033
discriminator_decay: 0.000001
batch_size: auto
```

```
discriminator_steps: 1
binary_encoder_cutoff: 150
binary_encoder_nan_handler: mode
auto_transform_datetimes: false
log_frequency: true
cbn_sample_size: 250000
epochs: auto
pac: 10
data_upsample_limit: 100
conditional_vector_type: single_discrete
reconstruction_loss_coef: 1
force_conditioning: auto
```

Here are also the last modified model parameters.

```
  - actgan:
      privacy_filters:
        outliers: medium
        similarity: high
        max_iterations: 10
      params:
        embedding_dim: 128
        generator_dim:
          - 512
          - 512
        discriminator_dim:
          - 512
          - 512
        generator_lr: 0.0002
        generator_decay: 0.00001
        discriminator_lr: 0.0004
```

```
        discriminator_decay: 0.00001
        batch_size: 64
        discriminator_steps: 1
        binary_encoder_cutoff: 150
        binary_encoder_nan_handler: mode
        auto_transform_datetimes: true
        log_frequency: true
        cbn_sample_size: 250000
        epochs: 100
        pac: 64
        data_upsample_limit: 100
        conditional_vector_type: single_discrete
        conditional_select_column_prob: 0.5
        conditional_select_mean_columns: null
        reconstruction_loss_coef: 1
        force_conditioning: true
      generate:
        num_records: 10000
        max_invalid: 500
      evaluate:
```

In the pursuit of optimizing the performance of the Anyway Conditional Tabular Generative Adversarial Network (ACTGAN) utilized in this study, a series of strategic adjustments were implemented to the model parameters. These modifications were informed by preliminary results and aimed to enhance the generative capabilities of the model, as well as to adapt to the specific characteristics of the dataset in use. Below, we delineate the rationale behind each alteration:

Generator and Discriminator Dimensions: The dimensions for both the generator and discriminator were escalated from 512 to 1024 units. This change aims to increase the model's capacity to capture and synthesize the complex patterns present in the data.

A higher-dimensional space allows for a more nuanced representation of data features, thereby improving the fidelity of the generated samples.

Learning Rates and Decay: Adjustments to the learning rates and decay parameters were made to fine-tune the training dynamics. The generator's learning rate was reduced from 0.0002 to 0.0001, and the discriminator's from 0.0004 to 0.00033, with both decays adjusted to 0.000001. These changes are intended to facilitate more stable and gradual learning, reducing the risk of overshooting minima in the optimization landscape. Batch Size: The batch size was set to `auto`, allowing the training process to dynamically adjust the number of samples processed simultaneously. This decision was based on the premise that an adaptable batch size can optimize memory usage and computational efficiency, potentially leading to improved model performance by adjusting to the optimal batch size for the given computational environment.

Epochs and PAC (Perturbation of Actor-Critic): The number of epochs was set to `auto`, and the PAC parameter was adjusted from 64 to 10. Allowing the model to determine the optimal number of training epochs can lead to more efficient training cycles, stopping the training process when additional epochs no longer contribute to significant improvements. Reducing the PAC parameter aims to decrease the granularity at which the discriminator evaluates batches, potentially leading to more robust discrimination and generation by focusing on broader patterns rather than overfitting to specific sample peculiarities.

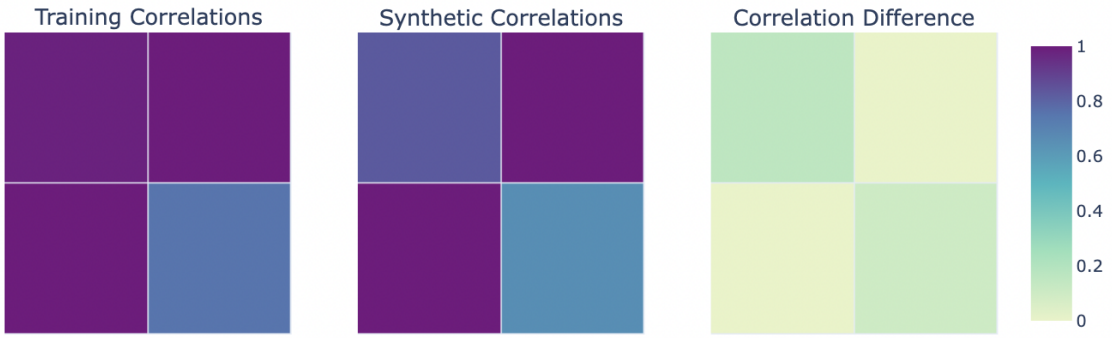

Figure 5.2: First Model Training and Synthetic Data Correlation

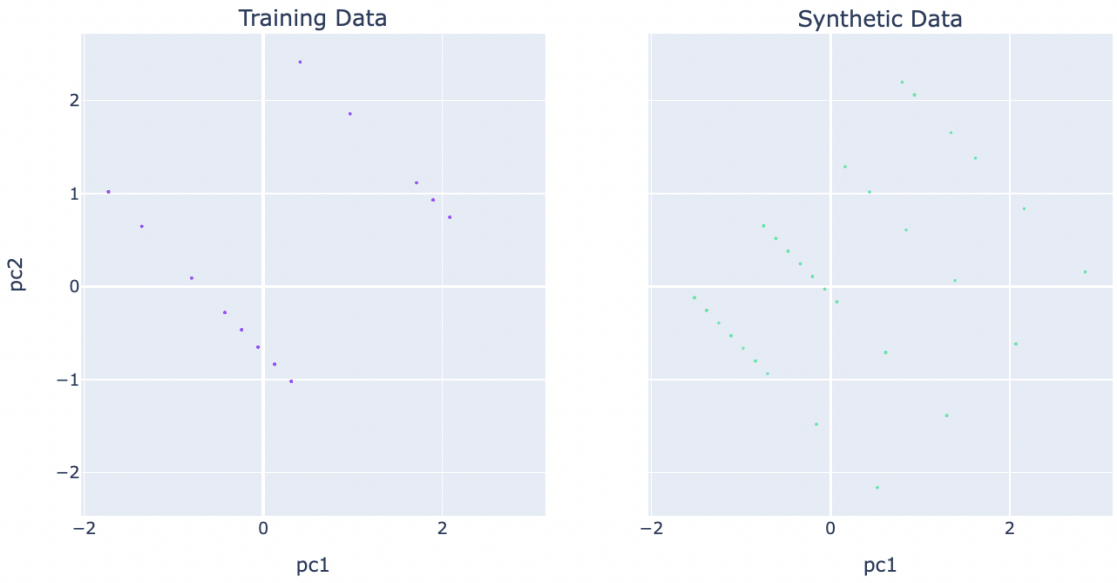

Figure 5.3: First Model Training and Synthetic Data PCA Results

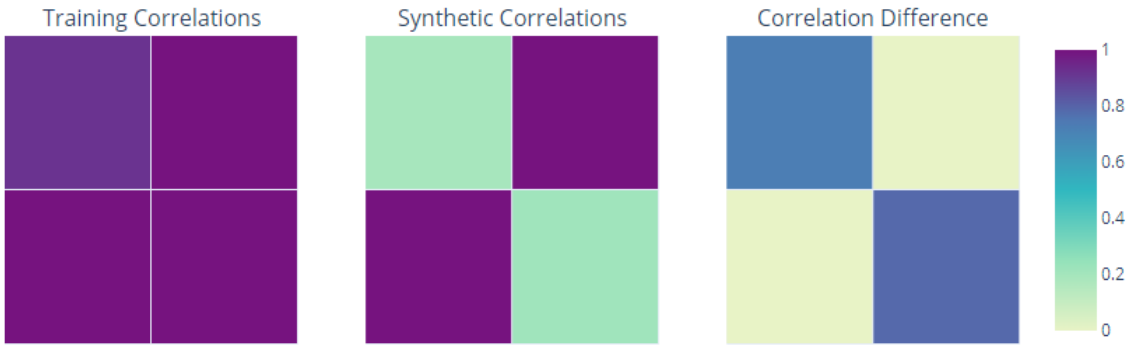

Figure 5.4: Last Model Training and Synthetic Data Correlation

The PCA visualizations serve as a tool to compare the dimensionality-reduced representations of both the real training dataset and the synthetically generated dataset. For the training data, the PCA plot shows a more focused distribution, indicating that the real-world scenarios, as captured by the 911 call logs, possess a narrower range of variance. This is expected as the original data likely contains consistent and specific patterns of emergency situations and their corresponding descriptive elements.

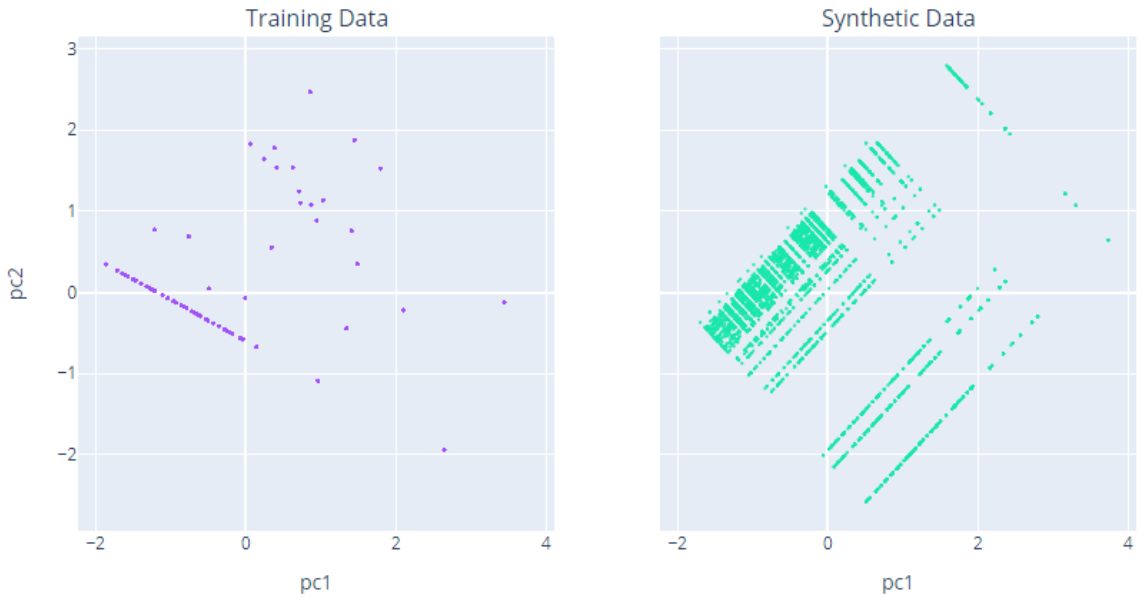

Figure 5.5: Last Model Training and Synthetic Data PCA Results

In contrast, the synthetic data plot reveals a broader spread across the principal component axes. This dispersion suggests that the synthetic generator has successfully introduced a wider variety of contextual elements and scenarios, which, while grounded in the reality of the original data, extends beyond it to encapsulate a richer set of potential emergencies and environmental conditions. The clusters within the synthetic data scatter plot point to the formation of subgroups that could represent unique or less frequent scenarios, adding to the robustness of the dataset. The PCA analysis validates that the synthetic data is not an exact replica of the training data but rather an expansion, aiming to enhance the language model's ability to generalize across a spectrum of possible emergency scenarios.

5.1.4 Example of Result

Input

- daytime
- clear, sky is clear
- Ambulance, Police Car
- heavy traffic

Output

- **Primary Scene:** A daytime urban setting under a clear sky, depicted in a simulation environment, featuring an ambulance and a police car navigating through heavy traffic conditions.
- **Time:** The scene is set during the day, with bright sunlight and a clear blue sky creating a bustling atmosphere in the city.
- **Weather:** The weather is perfectly clear, providing optimal visibility and a sense of warmth under the clear sky.
- **Interior:** View from inside a simulated emergency vehicle, focusing on the road and the challenging heavy traffic conditions, illustrating the complexities of navigating through congested streets under clear skies.
- **Road:** The road is congested with heavy traffic, characteristic of a busy daytime period in the city, with vehicles moving slowly due to the congestion.
- **Vehicles:** Both an ambulance and a police car are part of the traffic, showcasing their presence and the need for their services during busy times.
- **Buildings:** The urban buildings are bathed in sunlight, adding to the vibrancy of the cityscape under the clear sky.
- **Sky:** The sky is clear and blue, emphasizing the brightness and warmth of the day.
- **Relationships:** This scenario captures the dynamics of emergency vehicles operating in heavy traffic during a clear day, highlighting the challenges and importance of their roles in maintaining order during busy hours.

5.2 Summary of Generating Synthetic Data for Language Model

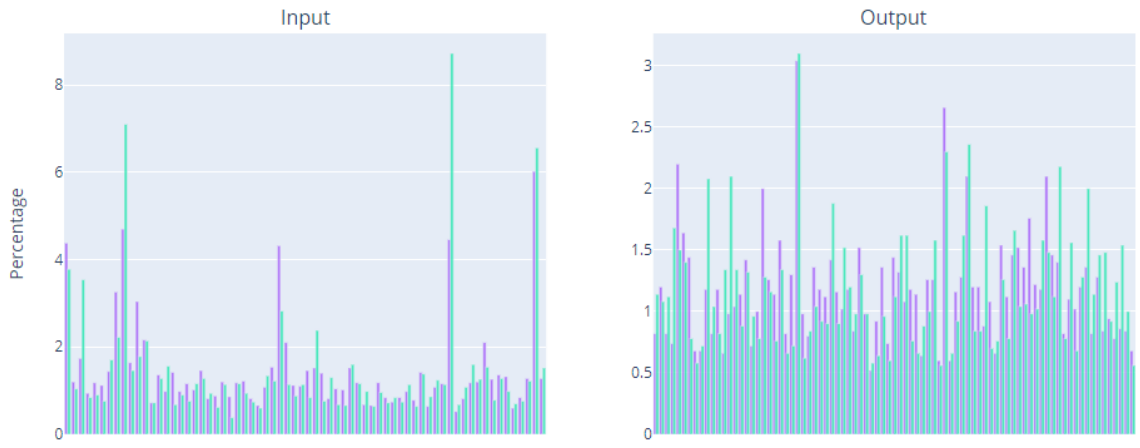

Figure 5.6: Field Distribution Comparisons

In the "Field Distribution Comparisons" graph, we observe two sets of data represented by color-coded histograms: the original training data in purple and the synthetically generated data in blue. These datasets are analyzed in terms of their 'Input' and 'Output' distributions, providing insights into the data engineered to serve as the training set for a GPT (Generative Pre-trained Transformer) language model.

The 'Input' histogram reveals a multitude of input scenarios. The synthetic dataset, shown in blue, exhibits a notably even distribution across a broad range of inputs, in contrast to the more variable frequency distribution of the original training data in purple. The consistency in the synthetic dataset's distribution is intentional, aimed at providing a comprehensive range of scenarios to prevent the model from overfitting to the less diverse original data.

The 'Output' histogram also displays the distribution of outcomes the GPT model is expected to generate. Here again, the blue bars of the synthetic data present a smooth and broad distribution, compared to the original data's purple bars, which indicate a narrower range of outputs. This broad coverage in the synthetic data is critical for the GPT model's ability to generalize across various outputs when generating text, ensuring

that it learns from a dataset that mirrors the complexity and diversity of real-world data.

The strategic generation of the synthetic data, as highlighted by the blue bars, is thus a deliberate effort to enhance the GPT model's training process. By including a wide array of synthesized scenarios, the model is encouraged to learn a more comprehensive representation of possible situations. This training methodology ensures that the model is not only trained on a rich dataset but also reflects a realistic variability that could be encountered in practical applications.

The graph underscores the synthesis of a robust training set, where the synthetic data (blue) is tailored to complement the original training data (purple), providing a well-rounded foundation for the GPT model's training. This preparation is instrumental for developing a model that can accurately simulate human-like text generation across a vast spectrum of topics and contexts.

In the process of enhancing the dataset for my analysis, I employed ACTGAN to generate synthetic data, thereby significantly augmenting the size of my original dataset. The expansion is reflected in a DataComPy comparison, which provides a detailed account of the structural integrity and augmentation success. The original dataset consisted of a modest 200 rows across two columns, which, through the application of synthetic data generation techniques, was expanded to an impressive 9,453 rows, while maintaining the original two columns. This careful expansion ensures that the dimensionality of the dataset remains constant, preserving the features for consistent comparison and analysis. The comparison was executed with a match on 'input' and 'output' columns, indicative of the datasets' relational attributes. Notably, the process introduced no duplicates in terms of matched values, and the data was verified with an absolute tolerance of 0.0001, ensuring high precision in the synthetic data. Despite the massive increase in data volume, there was no commonality in rows between the original and new datasets, which underscores the diversity and uniqueness of the generated records. Moreover, all compared columns were found to be equal when accounting for the defined tolerance, demonstrating that the values' range and characteristics were faithfully reproduced in the synthetic set. This extensive augmentation from the original 200 rows to 9,453 in the new dataset illustrates

a successful data generation endeavor, which provides a rich and varied foundation for training more robust predictive models, facilitating a deeper and more comprehensive analysis.

5.3 Large Language Model for Scene Description

In the domain of computational linguistics and artificial intelligence, Large Language Models (LLMs) represent a significant advancement in the ability of machines to process, understand, and generate human language (Goodfellow et al., 2016). These models leverage deep learning architectures, predominantly transformer models introduced by Vaswani et al. (Vaswani et al., 2017b), designed to handle sequential data and capture long-range dependencies within text. The "large" in LLMs refers to the substantial scale of the neural networks, which often contain billions or even trillions of parameters, enabling them to learn nuanced patterns of language from extensive corpora (Brown et al., 2020).

The transformer model, central to the operation of Large Language Models leverages a self-attention mechanism and positional encoding to process sequences of tokens. Below are key mathematical formulations that define its operation:

The self-attention mechanism allows each position in a sequence to attend to all positions within the same sequence, which is critical for understanding the contextual relationship between words or subwords:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.1}$$

where Q, K, and V represent the matrices for queries, keys, and values, respectively, and d_k is the dimensionality of the keys. This formulation helps stabilize the gradients by scaling the dot products.

To imbue the model with a sense of word order, or the position of tokens in a sequence, positional encodings are added to the input embeddings:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{5.2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{5.3}$$

where pos is the position in the sequence, i is the dimension, and d_{model} is the dimensionality of the token embeddings.

. The training of these models involves a two-phase process: unsupervised pre-training on a vast dataset to learn general language patterns, followed by fine-tuning on smaller, task-specific datasets to optimize performance for particular tasks (Radford et al., 2019). The training of LLMs involves minimizing the loss function, commonly the cross-entropy loss, to predict the next word in a sequence:

$$\mathcal{L}(\theta) = -\sum_{t} \log P(w_t | w_{<t}; \theta) \tag{5.4}$$

where w_t is the target word at time t, $w_{<t}$ denotes all preceding words, and θ represents the model parameters.

LLMs like GPT undergo pre-training on a vast corpus of text to learn general language patterns and are then fine-tuned on task-specific datasets:

- **Pre-training:** Learning from large, unlabeled text corpora.
- **Fine-tuning:** Adjusting model parameters θ on labeled datasets for specific tasks.

5.3.1 Implementation

In our effort to build a robust large language model despite limited authentic training data, we turned to synthetic data as a solution. To generate the synthetic datasets, we utilized Generative Adversarial Networks (ACTGANs). These datasets became an additional training resource to offset the shortage of real-world data.

Once the synthetic data was created, we merged these sequences and distributions with our original dataset, forming a comprehensive base for the large language model's

training phase. This approach not only increased the volume of our training data but also added a diverse range of synthetic examples that closely resembled real-world scenarios.

With the large language model equipped with this enriched dataset, we moved on to the deployment phase. At this stage, the model was tested against the entire set of actual data, incorporating every genuine instance. The success of this endeavor was evident in the large language model's consistent ability to generate coherent and contextually relevant text in response to new, unseen synthetic visual stimuli.

This achievement highlights two key outcomes. First, it demonstrates the large language model's skill in navigating and understanding synthetic data landscapes that closely mimic real-world conditions. Second, it confirms the effectiveness of synthetic data in supplementing real data shortages, offering a practical approach to data augmentation without compromising the quality of the learned representations.

In summary, our exploration into the use of synthetic data for large language model training has shown its considerable value. By tapping into the power of synthetic data, we've unlocked the large language model's potential to generate accurate textual interpretations for a continuous flow of synthetic imagery, effectively bridging the gap between limited data resources and the demand for extensive, high-quality training material.

```
training_configuration = {
    'batch_size': 4,
    'epochs': 3.0,
    'weight_decay': 0.01,
    'warmup_steps': 100,
    'lr_scheduler': 'linear',
    'learning_rate': 0.0002,
    'max_tokens': 512,
}

generation_configuration = {
    'num_records': 5000,
    'maximum_text_length': 3200,
```

```
}
```

```
def create_finetune_dataset(dataset_path: str) -> pd.DataFrame:

    records = []

    try:
        df = pd.read_csv(dataset_path, on_bad_lines='skip')
        df[LABEL_AND_TEXT_COLUMN] = df[LABEL_COLUMN] + SEPARATOR
           + df[TEXT_COLUMN]
        return df
    except FileNotFoundError:
        print(f"Error: File not found at '{dataset_path}'")
        return None

LABEL_AND_TEXT_COLUMN = 'label_and_text'
SEPARATOR = ':'

df = create_finetune_dataset(DATASET_PATH)
```

Listing 5.1: Python function to create a fine-tuning dataset.

```

def generate_synthetic_data(model: Model, prompt_df: pd.DataFrame
   ):
    """
    Generate synthetic data based on a prompt using an AI model.

    Args:
        model: The LLM used for generating synthetic data.
        prompt_df: A single-column dataframe containing the
```

```
            prompts.
9
10      Returns:
11          df: A dataframe containing the synthetic data generated
                by the model.
12      """
13
14      # Create a response handler object
15      response_handler = model.create_record_handler_obj(
16          params={"maximum_text_length": 1800, "temperature": 0.7},
17          data_source=prompt_df
18      )
19      response_handler.submit_cloud()
20      poll(response_handler, verbose=False)
21
22      # Read the response into a dataframe
23      df = pd.read_csv(response_handler.get_artifact_link("data"),
            compression='gzip')
24
25      return df
26
27  synthetic_data = generate_synthetic_data(model, prompt_df)
28  synthetic_data
```

Function Overview The function `create_finetune_dataset` takes the path to a dataset file in CSV format as its input and returns a pandas DataFrame. This DataFrame includes a new column named `label_and_text` which concatenates the class label and the associated text for each record, using a predefined separator.

Error Handling Within the function, error handling is in place to manage instances where the file is not found at the specified path, thereby ensuring that the function call does not result in a termination of the program but rather a graceful exit with an error message.

Usage The snippet ends with an example usage of the function, which would be tailored to the specific path of the dataset intended for use.

	Input	Output	label_and_text
0	Evening, Clear, sky is clear, Ambulance, Poli...	Primary Scene: A rain-drenched urban avenue a...	Evening, Clear, sky is clear, Ambulance, Poli...
1	Night, Rain, light rain, Rescue Vehicle, Norm...	Primary Scene: A daytime urban setting under ...	Night, Rain, light rain, Rescue Vehicle, Norm...
2	Night, Thunderstorm, thunderstorm with light ...	Primary Scene: An evening urban street scene ...	Night, Thunderstorm, thunderstorm with light ...
3	Evening, Clear, sky is clear, Fire Truck, Nor...	Primary Scene: A morning urban setting under ...	Evening, Clear, sky is clear, Fire Truck, Nor...
4	Morning, Clouds, broken clouds, Police Car, H...	Primary Scene: A clear morning in an urban se...	Morning, Clouds, broken clouds, Police Car, H...
...	...	...	...
4995	Evening, Clouds, few clouds, Ambulance, Polic...	Primary Scene: An evening urban setting with ...	Evening, Clouds, few clouds, Ambulance, Polic...
4996	Morning, Clear, sky is clear, Ambulance, Poli...	Primary Scene: A clear day urban setting with...	Morning, Clear, sky is clear, Ambulance, Poli...
4997	Evening, Thunderstorm, thunderstorm with ligh...	Primary Scene: An evening urban street scene ...	Evening, Thunderstorm, thunderstorm with ligh...
4998	Evening, Rain, light rain, Ambulance, Police ...	Primary Scene: A simulated urban street scene...	Evening, Rain, light rain, Ambulance, Police ...
4999	Evening, Rain, light rain, Ambulance, Police ...	Primary Scene: An evening urban setting under...	Evening, Rain, light rain, Ambulance, Police ...

5000 rows × 3 columns

Figure 5.7: Example of Training Dataset

5.3.1.1 Example of Result

Input Description Evening, Clear, sky is clear, Fire Truck, Heavy Traffic.

Output Description

- **Primary Scene:** An evening urban setting with clear skies, depicted in a simulation environment, featuring a fire truck navigating through heavy traffic conditions.
- **Time:** The scene is set during the evening, with clear skies and abundant stars, creating a serene and pleasant atmosphere over the city streets.
- **Weather:** The weather is clear with no signs of clouds or rain, making it an ideal and visually appealing driving environment.

- **Interior:** View from within a simulated fire truck, focusing on the road and the surrounding heavy traffic, illustrating an intense evening in the city.
- **Road:** The road shows heavy traffic, with vehicles moving slowly in the congested streets, reflecting an intense urban environment.
- **Vehicles:** A fire truck navigates through the city's heavy traffic, highlighting its role in responding to emergencies during busy evenings.
- **Buildings:** The city buildings stand tall under the clear skies, creating a visually engaging urban landscape during the evening.
- **Sky:** The clear sky adds to the overall sense of tranquility and beauty in the scene, with the city enjoying a pleasant evening.
- **Relationships:** This scenario captures the intense and visually appealing mood of an evening in the city under clear skies, with a fire truck responding to emergencies in heavy traffic conditions.

5.3.2 Model Accuracy

The Text Semantic Similarity Score, which ranges from 0 to 100, indicates how closely the meanings of real and synthetic texts align across the entire dataset. To understand this score, let's break down how it's calculated in our model:

Firstly, we use an embedding model to convert the text into a simpler, one-dimensional representation, with each text transformed into a vector of length 512. Then, we compute the cosine similarity between the average vectors of the training and synthetic texts. This similarity measurement tells us how similar the meanings of the texts are. For example, a score of 94 indicates that the synthetic texts generated by our model closely match the meaning of the original texts. In simpler terms, it means that the synthetic texts are almost indistinguishable from the real ones in terms of their meaning and context.

Moving on to the Text Structure Similarity Score, this metric assesses how closely the structure of synthetic texts resembles that of the original dataset. It focuses on aspects

like sentence length, average words per sentence, and characters per word distributions. But how do we determine this score?

We use something called Jensen-Shannon divergence, which is a statistical method that measures the difference between two probability distributions. In our case, it helps us compare the distribution of structural elements (like sentence length and word counts) between the real and synthetic texts. A score of 88 suggests that the synthetic data closely mirrors the original in terms of its structural makeup. This means that our synthetic texts maintain similar patterns in sentence length, word usage, and overall text structure compared to the real dataset. Essentially, it shows that our model has successfully captured the nuances of how text is constructed in the original dataset.

Following the training phase, the large language model underwent a rigorous validation process. A subset of 100 examples was selected from the validation dataset, which had not been seen by the model during training. The large language model then generated 100 textual outputs corresponding to these unseen examples. To evaluate the quality of the generated text, BERTScore, a metric for assessing the semantic similarity between two pieces of text, was employed.

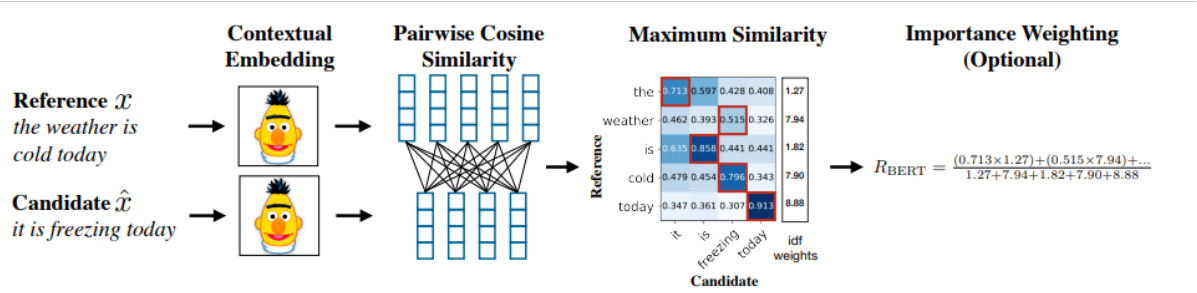

Figure 5.8: BERT Score Explanation (Zhang et al., 2019)

BERTScore is an automatic evaluation metric for text generation that computes a similarity score for each token in the candidate sentence with each token in the reference sentence using contextual embeddings.(Zhang et al., 2019)

It is utilized for evaluating generated text against gold standard references, applicable in domains such as machine translation and caption generation tasks. The computation

of BERTScore involves summing the cosine similarities between the tokens' embeddings of two sentences. This metric circumvents the limitations of n-gram-based approaches by leveraging contextualized token embeddings, thereby enhancing the evaluation of semantic equivalence.

Recall (R), Precision (P), and F1 Score (F1) Calculation:

- **Recall (R)**: The average of the maximum dot product between each word in the reference sentence x and the candidate sentence $\hat{x}$.
- **Precision (P)**: The average of the maximum dot product between each word in the candidate sentence $\hat{x}$ and the reference sentence x.
- **F1 Score (F1)**: Computed using the harmonic mean of Precision and Recall:

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

These scores collectively provide a nuanced evaluation of the similarity between reference and candidate sentences through the use of BERTScore.

The evaluation of the generated text using BERTScore yielded the following average scores across the dataset:

- Average Precision (P): 0.8342
- Average Recall (R): 0.8314
- Average F1 Score (F1): 0.8327

These scores represent the model's ability to generate text that aligns with the expected output. Precision reflects the proportion of relevant instances among the instances selected by the model. Recall indicates the proportion of relevant instances that were retrieved over the total amount of relevant instances. The F1 Score is the harmonic mean of precision and recall, providing a single score that balances the two metrics. The BERTScore outcomes demonstrate the model's high proficiency in generating semantically rich and contextually accurate descriptions of emergency scenarios across varied

environments. These metrics indicate not only the model's ability to replicate specific details and contexts accurately but also its success in maintaining the integrity of the emergency scenarios' complex and dynamic nature. The high Precision score reflects the generated text's relevance and alignment with the diverse scenarios depicted in the reference dataset, while the Recall score confirms that the model effectively encapsulates the critical elements of these scenarios. The balanced F1 score further affirms the model's adeptness at providing detailed, comprehensive narratives that closely mirror the reference texts.

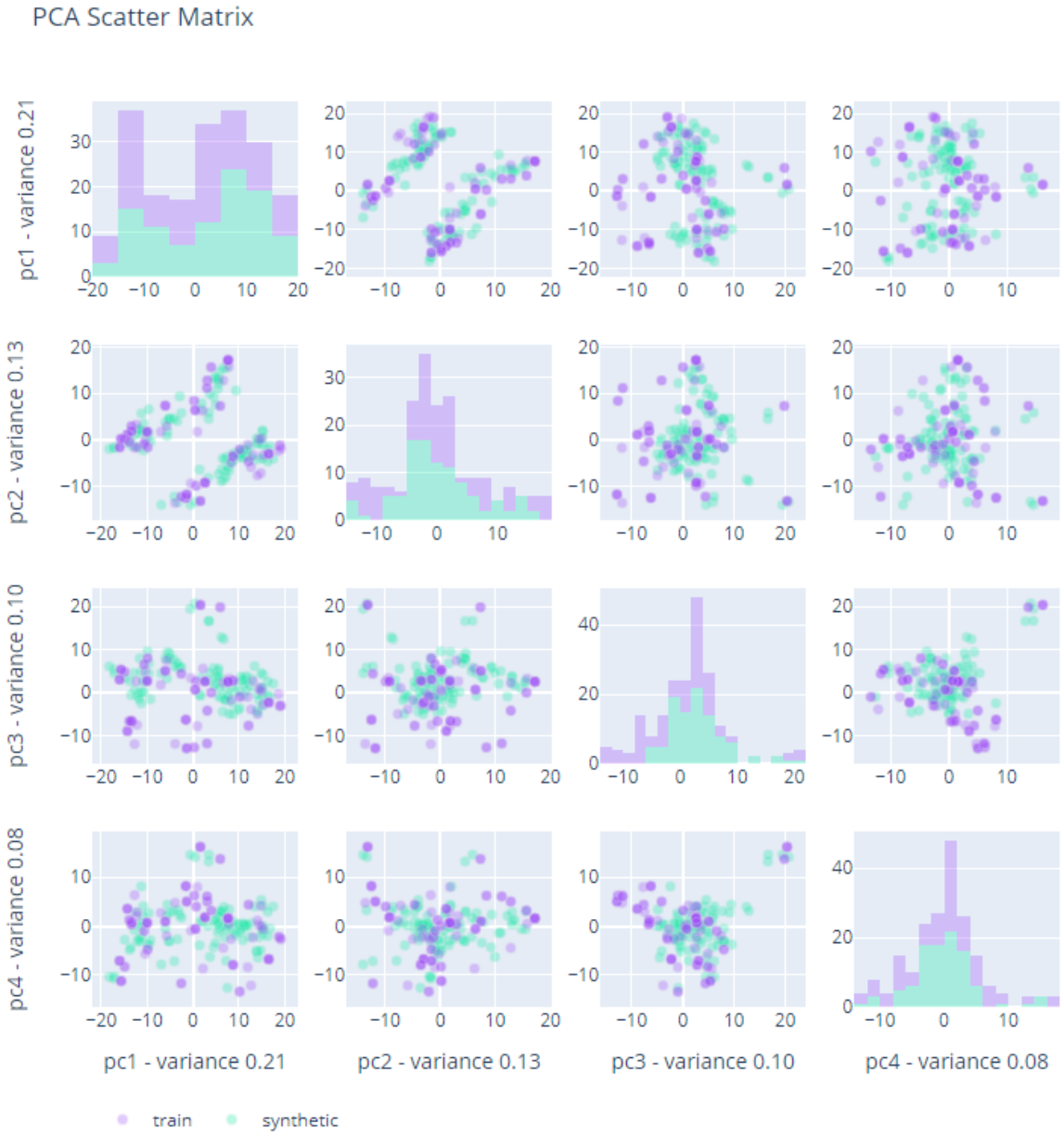

Figure 5.9: Principal Component Analysis Results

This image shows a scatter plot matrix generated through Principal Component Analysis (PCA). PCA is a dimensionality reduction technique used to visualize and analyze large datasets by projecting them onto a lower-dimensional space defined by principal components (PCs). Here are some observations:

- **Multiple Subplots**: The matrix contains multiple scatter plots. Each subplot compares two principal components, indicating how data points in the reduced-dimension space are distributed.
- **Histogram Diagonal**: The diagonal plots are histograms showing the distribution of data along individual principal components. These histograms help understand the spread and variance for each component.
- **Color Coding**: The color coding distinguishes between two datasets: a training dataset (in purple) and a synthetic dataset (in green). This distinction helps analyze how well the synthetic data aligns with the training data.
- **Similarity Between Datasets**: In many plots, the scatter points and histogram distributions for both datasets overlap significantly. This overlap suggests that the synthetic dataset closely resembles the training dataset in the PCA-reduced space, indicating successful replication of the training data's characteristics.
- **Variances of Principal Components**: Below each subplot, there is a mention of the variance explained by each principal component. The first component (PC1) explains the most variance (0.21), with subsequent components explaining lesser variance. This information indicates the proportion of the dataset's total variance captured by each principal component.

Overall, this scatter plot matrix suggests that the synthetic dataset aligns well with the training dataset, showing similar patterns in the reduced-dimensionality space. This alignment indicates that the synthetic data generation process has successfully replicated the key features and distributions of the original training data.

Chapter 6

Image Generation

In Chapter 4, the choice to use a diffusion model for image generation represented a significant advancement in our research. The decision was driven by the diffusion model's innovative capability to construct images from textual descriptions, enabling a new form of synthetic media creation.

Building on this foundation, we created a fine-tuned dataset to train a large language model. The dataset was synthesized using Anyway Conditional Tabular Generative Adversarial Networks (ACTGANs), chosen for their ability to produce high-fidelity synthetic data that closely resembles the complexity of real-world distributions.

The primary goal of synthesizing this data was to enhance the existing training set, especially when the original data was limited or lacked comprehensiveness. This approach not only expanded the dataset but also ensured that the linguistic patterns and intricacies that the large language model was to learn encompassed a wide range of variations. Such diversity is crucial for training the model to generate text prompts rich in context and varied in content.

Once developed, the large language model became central to a synthetic generation process aimed at producing descriptive texts. These texts are specifically crafted to be processed by a diffusion model, which creates images from textual descriptions. The goal behind developing the large language model went beyond text generation; it also involved creating precise and detailed descriptions that could guide the diffusion model

in synthesizing images that are coherent and contextually relevant.

This interaction between text generation and image synthesis forms the core of an end-to-end pipeline that starts with synthetic data generation by ACTGAN and ends with synthetic images created by a diffusion model. The images generated by the diffusion model are then evaluated to determine their quality and the extent to which they accurately represent the textual descriptions generated by the large language model.

In conclusion, the development of the large language model represents a critical step in a broader effort to generate synthetic images through a diffusion model. By leveraging the unique capabilities of ACTGAN and the large language model, we have established a comprehensive method for creating synthetic media. This innovative approach not only enhances the generation of artificial imagery but also opens up new possibilities for its application across various sectors of the digital media landscape.

6.0.1 Example of Generated Images

In this section, we demonstrate the input provided to the diffusion model and the output it generated. The input was created using a large language model (LLM), showcasing the versatility of LLM-generated prompts for guiding the diffusion process.

The following is a description of the input to the diffusion model:

Table 6.1: Input to Diffusion Model 1

Category	**Description**
Primary Scene	Rural or semi-rural road, likely in a region similar to Florida given the dense vegetation and Spanish moss.
Time	Daytime, as indicated by the natural light and shadows visible on the road.
Weather	Overcast or partly cloudy, with diffused lighting and no direct sunlight observed.
Foreground	Inside of a vehicle, possibly a car, with the dashboard in clear view, occupying the lower portion of the image, and the windshield serving as a transparent barrier, offering an unobstructed view of the exterior.
Midground	Roadway, with a paved two-lane rural road, featuring double yellow lines indicating no passing. A police car is parked or moving slowly on the right side of the road, and the surroundings are lush with a natural, less manicured look compared to urban areas.
Background	Rural Landscape, featuring dense greenery, trees draped with Spanish moss, wild underbrush, and a sky visible through the canopy, appearing mostly cloudy with soft light filtering through.
Relationships	The perspective is from inside a following car, capturing the scene through the windshield. The positioning of the police car suggests it may have pulled over or is attending to a matter off the road, and the surrounding foliage and road setup provide a sense of seclusion, typical of less populated areas.

Figure 6.1: (Output of Diffusion Model 1) Generated Image Example 1

Table 6.2: Input to Diffusion Model 2

Category	**Description**
Primary Scene	An evening urban setting with light rain, depicted in a simulation environment, featuring both an ambulance and a police car navigating through heavy traffic conditions.
Time	The scene is set during the evening, with light rain creating wet and challenging conditions over the city streets.
Weather	The weather features light rain, making it a demanding and potentially hazardous driving environment.
Interior	View from within a simulated emergency vehicle, with the focus on the road and the challenging driving conditions caused by the rain.
Road	The road shows heavy traffic, with vehicles moving cautiously in the rain, reflecting a challenging and intense urban environment.
Vehicles	Both an ambulance and a police car navigate the city under the rain, emphasizing their roles in maintaining safety.
Buildings	The city buildings stand tall under the rain, creating a dramatic and intense urban landscape during the evening.
Sky	The rain intensifies the atmosphere, with wet streets and reflections enhancing the city's mood.
Relationships	This scenario captures the intense mood of an evening in the city with light rain, with emergency vehicles dealing with heavy traffic.

Figure 6.2: (Output of Diffusion Model 2)Generated Image Example 2

Table 6.3: Input to Diffusion Model 3

Category	**Description**
Primary Scene	A simulated urban street scene at night under a clear sky, designed for a driving simulation or emergency response training.
Time	The setting is nighttime, highlighted by the darkness and the visibility of stars in the clear sky, along with artificial street lighting.
Weather	Weather conditions are simulated as clear, with a star-filled sky providing a serene backdrop to the urban environment.
Interior	View from within a simulated emergency vehicle, focusing on the clear view ahead through the windshield.
Road	The road shows a normal traffic flow, reflective of a calm night in the city.
Vehicles	An ambulance and a police car are present, blending in with the normal traffic, indicating a routine patrol or standby.
Buildings	Urban buildings flank the street, their lights twinkling under the clear night sky.
Sky	The sky is clearly simulated to represent a typical clear night.
Relationships	The scenario illustrates a typical night in the city, with emergency vehicles as part of the regular traffic.

Figure 6.3: (Output of Diffusion Model 3)Generated Image Example 3

Table 6.4: Input to Diffusion Model 4

Category	**Description**
Primary Scene	An evening urban setting with clear skies, depicted in a simulation environment, featuring a fire truck navigating through heavy traffic conditions.
Time	The scene is set during the evening, with clear skies and abundant stars, creating a serene and pleasant atmosphere over the city streets.
Weather	The weather is clear with no signs of clouds or rain, making it an ideal and visually appealing driving environment.
Interior	View from within a simulated fire truck, with the focus on the road and the surrounding heavy traffic, illustrating an intense evening in the city.
Road	The road shows heavy traffic, with vehicles moving slowly in the congested streets, reflecting an intense urban environment.
Vehicles	A fire truck navigates through the city's heavy traffic, highlighting its role in responding to emergencies during busy evenings.
Buildings	The city buildings stand tall under the clear skies, creating a visually engaging urban landscape during the evening.
Sky	The clear sky adds to the overall sense of tranquility and beauty in the scene, with the city enjoying a pleasant evening.
Relationships	This scenario captures the intense and visually appealing mood of an evening in the city under clear skies, with a fire truck responding to emergencies in heavy traffic conditions.

Figure 6.4: (Output of Diffusion Model 4)Generated Image Example 4

The examples of generated images, as illustrated in the thesis, highlight certain challenges in artificial intelligence recognition capabilities. Notably, in Image 6.1, identifying the presence of a police car poses a significant challenge for models such as YOLO (You Only Look Once), primarily because it appears indistinguishable from a standard vehicle at first glance. Similarly, the flashing lights of emergency vehicles, such as ambulances and police cars, during night-time emergency situations, present an additional challenge. Artificial intelligence models may confuse these emergency lights with standard traffic lights, leading to decreased accuracy scores. This confusion underscores the necessity for sophisticated recognition algorithms that can differentiate between various light sources, a task that can benefit from advanced activation functions in neural networks to enhance feature discrimination.

The inception of this research project was partly due to the insufficient quantity of emergency vehicle images available for training purposes. By augmenting datasets with labeled synthetic images that replicate emergency scenarios, the robustness of these datasets can be significantly enhanced. The synthetic images generated for this project were based on scenarios constructed from real-world data analysis of 911 call logs, ensuring relevance and applicability to actual emergency situations.

Incorporating synthetic images into training datasets strengthens the model's ability to recognize and differentiate between complex features in real-world scenarios.

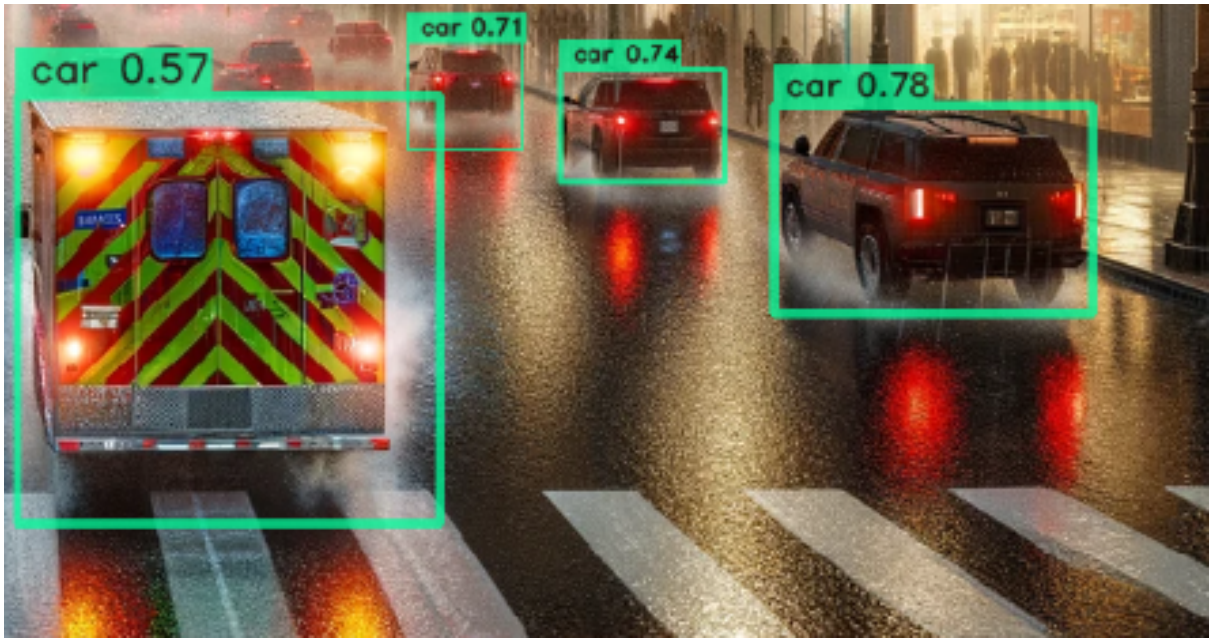

Figure 6.5: Generated Image in YOLO Model

Consistent with the introductory discussion, the challenge of object detection and classification models in accurately recognizing emergency vehicles is a recognized constraint within the field. This synthetic image underscores the issue: the fire truck, despite being prominently placed and closer to the viewpoint than other vehicles, is assigned a lower confidence score and erroneously classified as a 'car'. Such misclassification is particularly troubling given the distinct and critical nature of emergency vehicles in traffic scenes.

To demonstrate the consistency of this limitation, an image generated through object detection on a real-world scenario is compared with the synthetic image (Figure 1.1). The confidence scores from both instances align, indicating that these models often struggle with similar misclassification patterns when dealing with emergency vehicles.

The image further illustrates the limitations of current models, which can struggle with complex visual cues presented in real-world scenarios, such as reflective wet surfaces, varying light conditions, and unique vehicle designs. The fire truck, with its conspicuous reflective patterns and distinctive structure, should ostensibly be more recognizable; yet, the model's inability to accurately identify it suggests a need for enriched training protocols.

To remedy these deficiencies, an expansion of the training data is essential. This

expansion must include a substantial volume of accurately labeled, synthetic images that mirror the diverse conditions under which emergency vehicles operate. By synthesizing images from a corpus of real-world data and incorporating them into the training set, the models can be better equipped to recognize the array of features unique to emergency response vehicles.

The incorporation of these synthetic images, with precise annotations, is poised to enhance the model's performance significantly. This methodological enhancement will be vital in advancing object detection systems' reliability, as detailed in the forthcoming section on future works. There, strategies for data augmentation and model retraining will be outlined, providing a roadmap for overcoming the current limitations and advancing the state of the art in visual object recognition.

Figure 6.6: Generated Image Example 5

As shown in the image (Figure 6.6: Generated Image Example 5) , a police car is positioned in a far-right lane, which would typically be restricted or not used for regular traffic flow in a real-world setting. This positioning indicates a situation that would generally be improbable in standard traffic environments, suggesting it might be part of a training exercise, simulation, or designed to create more challenging conditions for object detection models.

While this might not align with typical road rules, such an arrangement can be valu-

Figure 6.7: Generated Image Example 6

able in several contexts:

- **Challenging Scenarios for Object Detection Models**: By placing the police car in an unconventional lane, models are presented with a scenario that requires higher-level inference to understand why a vehicle is in a seemingly incorrect position. This deviation from the norm can help object detection algorithms learn to manage more complex and unexpected situations.

- **Simulation and Training Environments**: Simulations often need to include unusual situations to prepare emergency response personnel for a wide range of events. By incorporating scenarios like this one, simulation developers can create more realistic and diverse training environments that reflect potential edge cases or emergency response conditions.

- **Testing Model Robustness**: Situations where vehicles are out of their expected positions can serve as a test for model robustness. If a model can accurately identify and classify a police car in such a setting, it suggests that the model might be capable of handling complex and unconventional arrangements in real-world scenarios.

This scene can serve as a useful resource for enhancing object detection models and creating more varied and challenging simulation environments. While it might seem unconventional, these scenarios are vital for advancing the capabilities of both object detection technology and emergency response training simulations.

Chapter 7

Conclusion

In this thesis, a critical observation was made regarding the challenges faced by artificial intelligence models, particularly in recognizing emergency vehicles within the generated images. As illustrated in Image 6.1, distinguishing a police car becomes notably difficult for AI models, such as those based on convolutional neural networks (CNNs) utilizing common activation functions like ReLU or Sigmoid. This difficulty arises because, in certain generated scenarios, the police car appears indistinguishable from a standard vehicle, lacking the distinctive visual cues typically relied upon for identification.

Further complicating model accuracy, emergency vehicles such as ambulances and police cars, when depicted during nighttime scenarios with activated emergency lights, present a unique challenge. AI models may erroneously interpret these lights as traffic signals due to their similar luminous characteristics. This confusion potentially leads to a decrease in the accuracy of such models when tasked with identifying emergency situations or distinguishing emergency vehicles from other traffic participants.

The genesis of this research project was, in part, the recognition of a paucity in the variety and quantity of emergency vehicle images within existing datasets. This shortage hampers the ability of AI models to learn the nuanced visual signatures of these vehicles, particularly in diverse operational contexts such as nighttime or emergency conditions where lighting plays a pivotal role.

By synthesizing and subsequently labeling a broader array of emergency vehicle im-

ages, including those with challenging lighting conditions, and integrating these into existing datasets, we can substantially enhance the robustness and accuracy of AI models. This approach not only addresses the immediate challenge of vehicle recognition in synthetic images but also contributes to the broader field of computer vision by providing richer datasets for training.

Incorporating activation functions that excel in handling nonlinearities and complex patterns, such as Leaky ReLU or ELU, could further augment the model's performance in distinguishing nuanced visual cues. Activation functions play a crucial role in neural networks by introducing nonlinear properties, enabling the network to learn complex patterns such as the subtle differences between an emergency vehicle and a regular vehicle under various lighting conditions.

In conclusion, the augmentation of datasets with a diverse range of labeled synthetic images of emergency vehicles represents a significant step forward in improving the perceptual accuracy of AI models. This endeavor not only aids in overcoming current limitations but also sets a foundation for future research in enhancing the cognitive capabilities of AI systems in recognizing and interpreting complex visual environments.

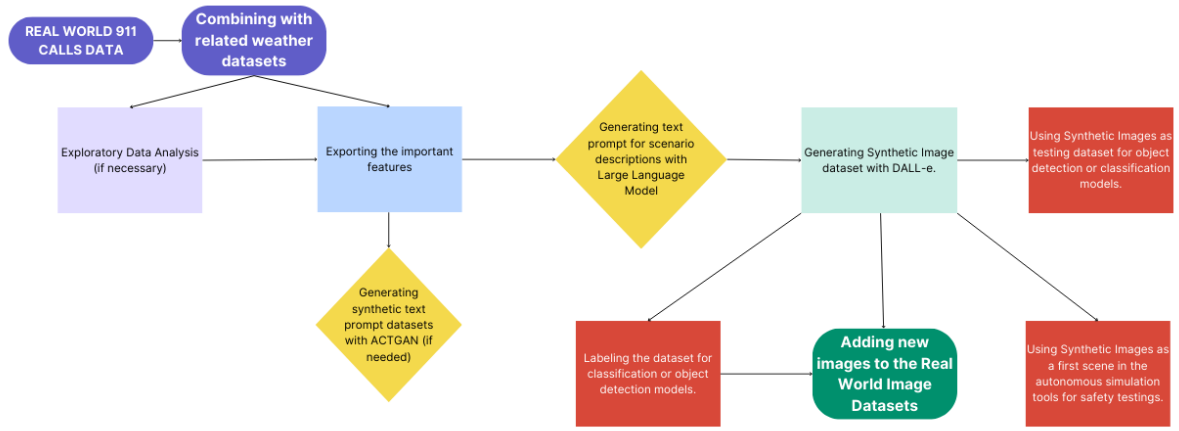

Figure 7.1: Thesis Diagram

In this thesis, the methodological approach for data analysis was paramount to understanding and extracting significant features from real-world 911 call data. The generation of synthetic text prompts was instrumental in supporting the training data for a large language model, given the text-to-image nature of the subsequent image generation with

a diffusion model. It is highlighted that for data beyond Orange County or for new datasets of Orange County 911 calls, there may not be a need to conduct additional Exploratory Data Analysis (EDA) or generate synthetic ACTGAN prompts. The structure elucidated herein delineates a workflow where the initial step involves amalgamating the 911 call data with pertinent datasets and extracting crucial features. Subsequently, the necessary large language model (LLM) and diffusion models convert real-world data into synthetic image datasets.

The subsequent phases of utilizing the created synthetic image datasets may pivot to various research directions. Although the inception of this project was driven by the absence of adequate emergency vehicle imagery in datasets, impeding the identification capabilities of object detection and classification models used in autonomous driving, it is imperative to recognize the broader applications of these synthetic datasets. They hold potential utility across diverse domains, underscoring the versatility and extendibility of the generated synthetic image data beyond the initial scope of improving emergency vehicle recognition in autonomous navigation systems.

This workflow affirms the innovative use of AI in addressing data gaps within critical application fields and opens up avenues for leveraging synthetic imagery for multipurpose applications, thereby contributing to the advancement of AI in practical scenarios.

7.1 Future Work

The research trajectory has embraced an exploratory and experimental methodology, with the initial findings indicating promising benefits. The following avenues are proposed for future investigation:

1. **Large-Scale Real-Synthetic Dataset Compilation:** The assembly of an extensive dataset that melds real and synthetic imagery is imperative. This repository will underpin the development of advanced object detection and classification models, enhancing the diversity and realism of training data.

2. **Benchmarking with Real-World Data Models:** It is essential to evaluate the

performance of models trained on the hybrid dataset against those trained solely on real-world data. Comparative analysis will elucidate the value added by synthetic data in terms of model performance.

3. **Integration into Simulation Testing:** Synthetic images should be utilized as test data within simulated environments. Such simulations will provide a controlled testing platform, especially beneficial in data-constrained scenarios.

4. **Iterative Development and Evaluation:** Adopting a cyclic approach to model development, encompassing training, testing, and refinement, with the continual incorporation of new data, is recommended for progressive enhancement of model accuracy.

5. **Exploration of Cross-Domain Applicability:** The potential applications of synthetic data sets extend beyond the initial scope of autonomous vehicle systems. Future research should investigate the applicability in other domains reliant on precise object detection.

6. **Assessment of AI Trustworthiness:** Future studies should examine the impact of synthetic data on the trustworthiness of AI systems, assessing how such data affects model confidence and public perceptions of AI reliability.

7. **Creation of More Challenging Scenarios:** Developing complex and challenging scenarios for AI models is crucial. This approach involves incorporating edge cases, unexpected situations, and high-complexity environments to test and improve model robustness. By simulating these scenarios, models can better adapt to real-world unpredictability.

8. **Adaptive Connectivity in Simulations:** To ensure AI models can interact effectively with other systems, future work should focus on adaptive connectivity within simulations. This includes enabling AI models to communicate and adapt in response to varying conditions and contexts, enhancing their ability to operate in dynamic environments.

These initiatives are anticipated to make substantial contributions to AI research, notably in refining object detection capabilities and in broadening our understanding of synthetic data integration within machine learning models.

References

A. Dosovitskiy, F. C. A. L., G. Ros, & Koltun, V. (2017). *Carla: An open urban driving simulator.* Proceedings of the 1st Annual Conference on Robot Learning.

A. Ghosh, B. B., & Chowdhury, S. B. R. (2016). *Sad-gan: Synthetic autonomous driving using generative adversarial networks.* arXiv preprint arXiv:1611.08788.

Bergamini, L., Ye, Y., Scheel, O., Chen, L., Hu, C., Del Pero, L., ... Ondruska, P. (2021). *Simnet: Learning reactive self-driving simulations from real-world observations.*

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... others (2020). Language models are few-shot learners. In *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901).

Ciresan, D., Giusti, A., Gambardella, L., & Schmidhuber, J. (2013). *Mitosis detection in breast cancer histology images with deep neural networks.* Springer.

Ciresan, D., Meier, U., Masci, J., Maria Gambardella, L., & Schmidhuber, J. (2011). *Flexible, high performance convolutional neural networks for image classification* (Vol. 22).

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). *Multi-column deep neural networks for image classification.*

D. J. Fremont, S. G. X. Y. A. L. S.-V., T. Dreossi, & Seshia, S. A. (2019). *Scenic: a language for scenario specification and scene generation.* Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation.

Doersch, C. (2016). *Tutorial on variational autoencoders.* (Carnegie Mellon, UC Berkeley)

D. Zhao, J. W., & Liu, Y. (2017). *Generating traffic scene with deep convolutional*

generative adversarial networks. 2017 Chinese Automation Congress (CAC).

E. Pronovost, K. W., & Roy, N. (2023). *Generating driving scenes with diffusion.* arXiv preprint arXiv:2305.18452.

Feng, L., Li, Q., Peng, Z., Tan, S., & Zhou, B. (2023). *Trafficgen: Learning to generate diverse and realistic traffic scenarios.*

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). *Generative adversarial nets* (Vol. 27).

Graves, A. (2008). *Supervised sequence labelling with recurrent neural networks.*

Graves, A. (2013). *Generating sequences with recurrent neural networks.* arXiv preprint arXiv:1308.0850.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., & Wierstra, D. (2015). *Draw: A recurrent neural network for image generation.*

H. Avşar, M. S., & Karacan, L. (2022). *Scene construction from depth map using image-to-image translation model* (Vol. 5) (No. 1).

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory* (Vol. 9) (No. 8).

Huang, C.-W., Krueger, D., Lacoste, A., & Courville, A. (2018). *Neural autoregressive flows.* arXiv:1804.00779 [cs, stat].

J. Devaranjan, A. K., & Fidler, S. (2020). *Meta-sim2: Unsupervised learning of scene structure for synthetic data generation.* Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII.

Kar, A., Prakash, A., Liu, M.-Y., Cameracci, E., Yuan, J., Rusiniak, M., ... Fidler, S. (2019). *Meta-sim: Learning to generate synthetic datasets.*

Khemakhem, I., Monti, R., Leech, R., & Hyvarinen, A. (2021). *Causal autoregressive flows.*

Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). *Semi-supervised learning with deep generative models.*

Kingma, D. P., & Welling, M. (2014). *Auto-encoding variational bayes.*

Kobyzev, I., Prince, S., & Brubaker, M. (2020). *Normalizing flows: An introduction and*

review of current methods.

Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). *Deep convolutional inverse graphics network.*

Lin, H. W., & Tegmark, M. (2017). *Criticality in formal languages and statistical physics* (Vol. 19) (No. 7).

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., ... Wießner, E. (2018). *Microscopic traffic simulation using sumo.*

Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., ... Urtasun, R. (2020). *Lidarsim: Realistic lidar simulation by leveraging the real world. 2020 ieee.*

Ming, Z., & Jun, L. (2020). *Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving.*

M. Qi, A. L., Y. Wang, & Luo, J. (2020). *Stc-gan: Spatio-temporally coupled generative adversarial networks for predictive scene parsing* (Vol. 29).

O'Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks.* arXiv preprint arXiv:1511.08458.

Papamakarios, G., Nalisnick, E., Rezende, D., Mohamed, S., & Lakshminarayanan, B. (2019). *Normalizing flows for probabilistic modeling and inference.* arXiv preprint arXiv:1912.02762.

Papamakarios, G., Pavlakou, T., & Murray, I. (2018). *Masked autoregressive flow for density estimation.* arXiv:1705.07057 [cs, stat].

Pascanu, R. (2014). *On recurrent and deep neural networks.*

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). *On the difficulty of training recurrent neural networks.*

P. Cai, H. W., Y. Sun, & Liu, M. (2020). *Vtgnet: A vision-based trajectory generation network for autonomous vehicles in urban environments* (Vol. 6) (No. 3).

Periyasamy, A. S., & Behnke, S. (2023). *Towards 3d scene understanding using differentiable rendering* (Vol. 4) (No. 3).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language

models are unsupervised multitask learners. URL https://cdn.openai.com/better-language-models/language$_m odels_a re_u nsupervised_m ultitask_l earners.pdf$.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). *Stochastic backpropagation and approximate inference in deep generative models.*

Salimans, T., Kingma, D., & Welling, M. (2015). *Markov chain monte carlo and variational inference: Bridging the gap.*

Sherstinsky, A. (2020). *Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network* (Vol. 404).

Sohn, K., Lee, H., & Yan, X. (2015). *Learning structured output representation using deep conditional generative models.*

Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., & Urtasun, R. (2021). *Scenegen: Learning to generate realistic traffic scenes.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017a). *Attention is all you need* (Vol. 30).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017b). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Walker, J., Doersch, C., Gupta, A., & Hebert, M. (2016). *An uncertain future: Forecasting from static images using variational autoencoders.*

W. Ding, B. L., H. Lin, & Zhao, D. (2023). *Causalaf: Causal autoregressive flow for safety-critical driving scenario generation.*

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling tabular data using conditional gan.*

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. In *International conference on learning representations.*