POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Applications of Generative AI via Latent Diffusion for Explainability in Geolocalization

Supervisors

Dr. Carlo MASONE

Gabriele BERTON

Gabriele TRIVIGNO

Prof. Barbara CAPUTO

Candidate

Emanuele PINNA

October 2024

Summary

This thesis explores the integration of conditional generative models, specifically diffusion models, with Cosplace, a model designed for visual geolocation tasks. The primary objective is to enhance the explainability of geolocation models by generating images that visualize how these models interpret spatial features. Leveraging the Pittsburgh dataset, the conditional model was trained to generate images based on embeddings produced by Cosplace, offering valuable insights into the internal decision-making process of geolocation networks.
Throughout the study, we demonstrate that the conditional generative approach yields impressive results, even at early stages of training, with the generated images closely resembling the original scenes. However, certain limitations in the model's ability to capture dynamic elements, such as moving vehicles and pedestrians, as well as finer architectural details, were observed. These challenges were largely attributed to the constraints of the Cosplace embeddings and the computational limitations encountered during training.
Key challenges included the need to reduce the model complexity due to hardware restrictions and limitations on image resolution and batch size. Despite these constraints, the integration of latent diffusion models for explainability proved to be highly innovative and effective, offering a novel method to visualize and interpret the workings of complex AI models.
This research not only advances the explainability of AI-driven geolocation systems but also paves the way for broader applications of generative AI across a wide range of disciplines, offering new opportunities for interpreting complex data and enhancing model transparency. The findings underscore the potential of diffusion models to provide interpretable, human-understandable outputs in a variety of AI-driven tasks, offering valuable insights across multiple domains.

Acknowledgements

I would like to express my heartfelt gratitude to my parents, whose unwavering support and wise counsel have been my guiding light through every decision I have made. Their belief in me has instilled a sense of confidence that has propelled me forward.
To my brother, thank you for lifting my spirits and filling my days with laughter; your encouragement has motivated me to strive for excellence in creating a better future.
I am profoundly grateful to my grandparents, whose steadfast faith in my abilities continues to inspire me; their legacy lives on through my achievements.
To the rest of my family, your love and encouragement have provided a foundation of strength that has sustained me through challenges and triumphs alike.
I extend my appreciation to my past and present roommates, who have brightened my daily life with their camaraderie; the myriad of cherished moments we shared have made these years truly unforgettable.
I am deeply thankful to my supervisors Carlo Masone, Gabriele Berton, Gabriele Trivigno, and Barbara Caputo, for their invaluable guidance and support throughout the thesis process. Your insights have enriched my knowledge and equipped me with essential skills for the future.
To the Vandal Lab, thank you for welcoming me with open arms and fostering an environment of collaboration and friendship from the very beginning.
Lastly, to all my friends, though I cannot name you individually, your presence has made this journey less lonely, and your willingness to lend a helping hand whenever needed has meant the world to me. Each of you has contributed to my growth and success in ways I will forever cherish.

"The real risk with AI isn't malice but incompetence."
— Elon Musk

Table of Contents

List of Figures

Acronyms

AI
: Artificial Intelligence

LDM
: Latent Diffusion Model

GAN
: Generative Adversarial Networks

VAE
: Variational Auto-Encoder

DDPM
: Denoising Diffusion Process Method

DDIM
: Denoising Diffuison Implicit Models

Chapter 1

Introduction to Artificial Intelligence

1.1 Overview of Artificial Intelligence

1.1.1 Definition and historical background of AI

Artificial Intelligence refers to the ability of machines, particularly computers, to perform tasks that traditionally require human intelligence. These tasks include reasoning, problem-solving, learning, perception, and natural language understanding. AI is broadly categorized into two types: **narrow AI** (or weak AI), which is designed for specific tasks such as image recognition or playing chess, and **general**

AI (or strong AI), which aspires to achieve a level of intelligence that matches or surpasses human capabilities across a wide range of tasks.
The roots of AI date back to the 1950s when computer scientists first began exploring the idea of creating machines that could simulate human thought. The term **"Artificial Intelligence"** was coined in 1956 by John McCarthy at the Dartmouth Conference, which is widely considered the birth of AI as an academic discipline. Early efforts in AI focused on symbolic reasoning and logic-based systems. Notable milestones include Alan Turing's development of the **Turing Test** in 1950 to evaluate a machine's ability to exhibit intelligent behavior, and the creation of the first AI programs such as *the Logic Theorist (1955)* and the *General Problem Solver (1957)*.
Over the decades, AI has experienced periods of both rapid progress and stagnation, commonly referred to as **"AI winters"**. These periods of low funding and interest occurred due to the gap between high expectations and the actual capabilities of AI systems. However, the field witnessed a resurgence in the 1990s and 2000s, primarily due to advancements in computational power, the availability of large datasets, and new approaches such as machine learning and neural networks. Today, AI plays an integral role in various sectors, driving innovations in automation, intelligent decision-making, and human-computer interaction.

1.1.2 Key concepts: agents, machine learning, and decision-making

AI is grounded in several foundational concepts, three of which are particularly central to its modern applications: intelligent agents, machine learning, and decision-making systems.

Agents

In AI, an agent refers to any system that perceives its environment and acts upon it to achieve specific goals. Agents can be as simple as a thermostat or as complex as an autonomous vehicle. They operate based on three components: perception, which allows them to observe their surroundings through sensors; processing, where they use algorithms to make sense of the information they perceive; and action, which is the execution of tasks based on the decisions they make.
Agents are categorized as either **reactive** (responding directly to stimuli without reasoning about the future) or **deliberative** (planning actions by reasoning about future states). Advanced AI agents often incorporate elements of both, allowing them to act reflexively in some situations while engaging in more complex, long-term planning in others.

Machine Learning

Machine learning (ML) is a subset of AI focused on developing algorithms that allow computers to learn from data. Instead of being explicitly programmed for each task, ML algorithms detect patterns and make predictions or decisions based on data inputs. This approach enables AI systems to improve their performance over time without human intervention.
Machine learning is typically divided into three types:

- **Supervised learning**, where the system is trained on labeled datasets and learns to map inputs to specific outputs (e.g., image classification).
- **Unsupervised learning**, where the system analyzes data without predefined labels to find hidden patterns or relationships (e.g., clustering).
- **Reinforcement learning**, where an agent learns by interacting with its environment and receiving feedback through rewards or penalties, refining its actions to maximize long-term success (e.g., game playing or robotic control).

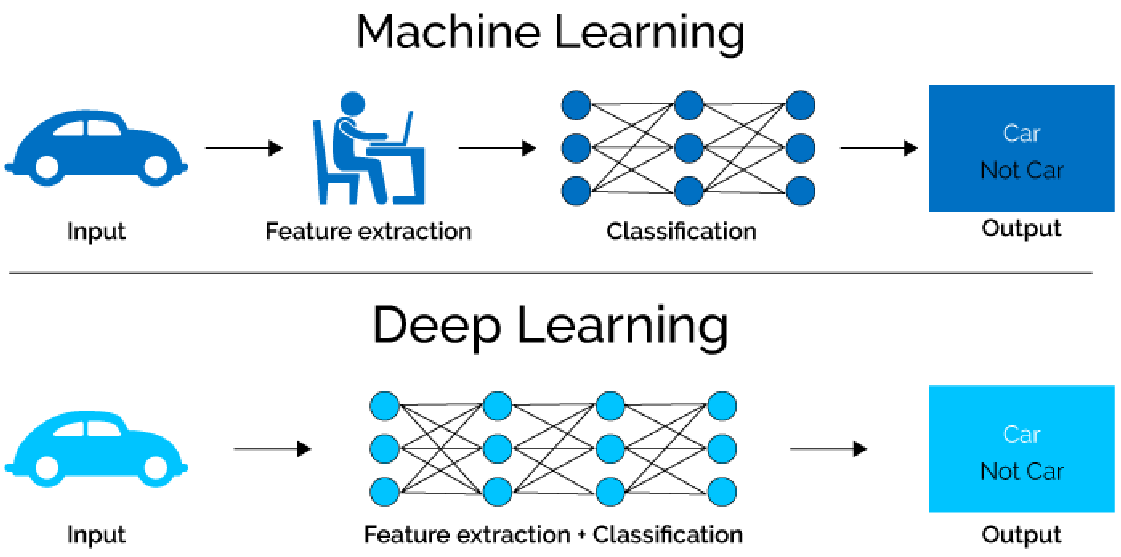
The growth of machine learning, particularly deep learning, has revolutionized AI in recent years. Deep learning models, which mimic the structure of the human brain using artificial neural networks, have achieved remarkable results in complex tasks such as language translation, image recognition, and even autonomous driving.

Decision-Making

Decision-making in AI involves selecting the optimal course of action from a set of alternatives based on available data and objectives. AI systems use various decision-making algorithms, including optimization techniques, rule-based systems, and probabilistic reasoning, to choose actions that maximize outcomes or minimize risks. In many cases, decision-making systems work in uncertain environments, requiring them to make predictions about future states or account for incomplete information.
The process of decision-making can be broadly categorized into deterministic and stochastic methods. Deterministic decision-making assumes certainty in outcomes and operates under fixed rules, while stochastic decision-making incorporates elements of randomness and probability, making it more suitable for real-world environments where uncertainty and variability are common.
In complex AI systems, such as autonomous robots or self-driving cars, decision-making involves balancing multiple objectives, often under real-time constraints. These systems must consider factors such as safety, efficiency, and ethical implications, making decision-making one of the most critical and challenging areas in AI research.

1.2 Shallow Learning vs Deep Learning

In the realm of machine learning, a distinction is made between shallow learning and deep learning methods. Shallow learning refers to traditional machine learning algorithms that often rely on handcrafted features and simpler structures. In contrast, deep learning involves complex, multilayered neural networks that automatically extract features from raw data. This section explores the key differences between these approaches, focusing on the respective methodologies, architectures, and real-world impacts.



1.2.1 Overview of Shallow Learning methods (SVMs, Decision Trees, etc.)

Shallow learning methods are traditional **machine learning techniques** that typically rely on a limited number of layers or decision processes. These algorithms excel in scenarios with smaller datasets and well-defined features, where computational efficiency is crucial. Common shallow learning algorithms include:

Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a powerful supervised learning algorithm used for classification and regression tasks. SVMs work by finding the optimal hyperplane that separates different classes in the data. In a high-dimensional space, this hyperplane maximizes the margin between the nearest data points from each class, known as support vectors. SVMs are particularly effective in binary classification problems and are known for their ability to handle high-dimensional feature spaces. However, they require careful tuning of parameters and feature selection to achieve optimal performance.

Decision Trees

Decision Trees are another popular shallow learning method used for both classification and regression tasks. A decision tree builds a model by recursively splitting the data into subsets based on the value of input features, creating a tree-like structure. Each node in the tree represents a decision based on a feature, and each leaf node corresponds to a predicted output or category. Decision Trees are easy to interpret and implement, making them ideal for tasks where transparency and explainability are important. However, they can be prone to overfitting, especially when the tree becomes too deep.

k-Nearest Neighbors (k-NN)

The **k-Nearest Neighbors (k-NN)** algorithm is a simple, instance-based learning method used for classification and regression. In this approach, predictions are made by finding the "k" closest data points (neighbors) to a given query point and determining the majority class or average value of those neighbors. k-NN is intuitive and easy to implement, but it becomes computationally expensive with large datasets, as it requires calculating distances between the query and all other points.

Naïve Bayes

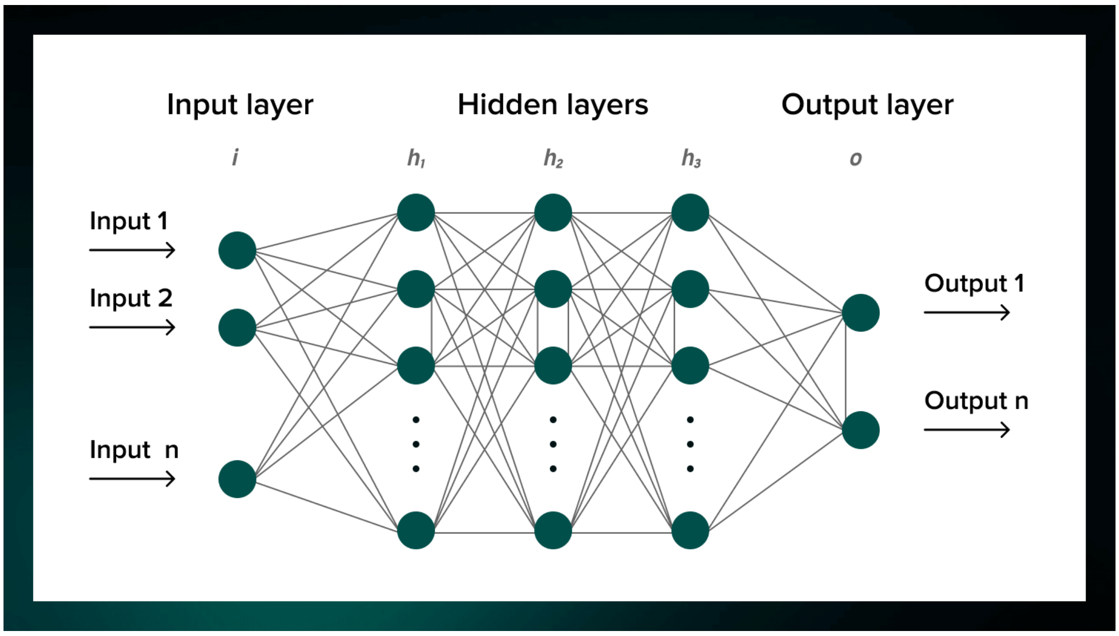
Naïve Bayes[1] is a probabilistic classifier based on Bayes' Theorem, with the assumption that features are conditionally independent given the class label. Despite the "naïve" assumption, this method often performs well on real-world data, particularly for text classification tasks like spam detection or sentiment analysis. Naïve Bayes is highly efficient and can be used with large datasets, but its performance may degrade if features are highly correlated.
Shallow learning methods are effective for many applications, particularly when the feature space is well-understood and smaller datasets are available. However, they often struggle with high-dimensional data and complex feature interactions, which is where deep learning approaches come into play.

1.2.2 Deep Learning advancements: Neural Networks and their architectures

Deep learning, a subset of machine learning, uses artificial neural networks (ANNs) with multiple layers to automatically extract features from raw data and make predictions. The "depth" in deep learning refers to the number of layers in the neural network, which allows it to model complex relationships and learn hierarchical representations of data. Over the past decade, deep learning has led to breakthroughs in areas such as computer vision, natural language processing, and reinforcement learning.

Neural Networks

At the core of deep learning are **artificial neural networks (ANNs)**[2], which are computational models inspired by the structure and function of the human brain. A neural network consists of interconnected nodes (neurons), organized into layers: an input layer, one or more hidden layers, and an output layer. Each neuron receives inputs, processes them through an activation function, and sends the output to the next layer. The network learns by adjusting the weights of the connections between neurons through a process called backpropagation.



Architectures in Deep Learning

Several specialized neural network architectures have emerged to address specific types of data and tasks:

- **Convolutional Neural Networks (CNNs)**: CNNs are designed to process grid-like data structures, such as images. They use convolutional layers to automatically detect spatial hierarchies of features, making them highly effective for image recognition, object detection, and video analysis.
- **Recurrent Neural Networks (RNNs)**: RNNs are used for sequential data, such as time series or natural language. They maintain a memory of previous inputs through feedback loops, allowing them to capture temporal dependencies in data. Long Short-Term Memory (LSTM) networks are a type of RNN specifically designed to address the problem of vanishing gradients in long sequences.
- **Transformer Networks**: Transformers have revolutionized natural language processing (NLP) tasks. Unlike RNNs, transformers do not process data sequentially but rely on an attention mechanism to focus on different parts of the input. This allows them to process text and other sequential data more efficiently, and models like BERT and GPT have set new benchmarks in NLP.

Deep learning's ability to automatically learn features from vast amounts of data, along with the availability of large datasets and powerful hardware (e.g., GPUs), has driven its success in solving complex, high-dimensional problems.

1.3 Importance of AI in Modern Applications

The advancements in both shallow and deep learning have cemented AI's role in modern applications across various domains. AI's ability to process and analyze large volumes of data, identify patterns, and make informed decisions is reshaping industries and enabling new possibilities.

1.3.1 AI's impact across industries and real-world applications

AI's influence extends across multiple industries, providing innovative solutions to long-standing challenges and opening new avenues for growth.

- **Healthcare**: In healthcare, AI systems are improving diagnostic accuracy by analyzing medical images, patient records, and genetic data. Deep learning models are used for tasks like detecting cancerous tumors in radiology scans or predicting patient outcomes. AI is also accelerating drug discovery by analyzing vast chemical datasets and predicting potential drug compounds faster than traditional methods.

- **Finance**: In finance, AI enhances fraud detection systems by identifying unusual patterns in transaction data. Machine learning models predict stock market trends, optimize portfolio management, and automate trading. AI-powered chatbots and virtual financial advisors offer personalized services, improving customer engagement and satisfaction.
- **Transportation and Autonomous Systems**: The transportation industry is leveraging AI for autonomous vehicles, where deep learning models process visual and sensor data to enable self-driving cars. AI algorithms control navigation, detect obstacles, and make real-time decisions, enhancing the safety and efficiency of transport systems. AI is also used in smart traffic management to reduce congestion and improve urban mobility.
- **Entertainment and Media**: In the entertainment industry, AI has transformed content creation, recommendation systems, and user experience. Streaming platforms use AI-driven recommendation engines to curate personalized content for users based on their viewing habits. AI is also used in video game development to create intelligent, adaptive virtual environments.

Chapter 2

Generative Artificial Intelligence

Midjourney Version Comparison
V1
Released February 2022
V2
Released April 12, 2022
V3
Released July 25, 2022
V4
Released November 5, 2022
V5
Released March 15, 2022
V5.1
Released May 3, 2022

Generative Artificial Intelligence (Generative AI) is a rapidly evolving field within AI that focuses on creating models capable of generating new data similar to the data they were trained on. Unlike traditional AI, which often centers around recognizing patterns or making decisions based on existing data, generative AI can produce novel outputs, including text, images, audio, and even video. This section explores the core principles of generative AI, including key concepts of generative models and how they differ from discriminative AI.

2.1 What is Generative AI?

Generative AI refers to a class of artificial intelligence models that can learn from training data to create new, similar data that did not exist in the original dataset. These models capture the underlying distribution of the input data, enabling them to generate new instances that share key characteristics of the training data. Applications of generative AI range from generating realistic images, synthesizing human-like text, creating music, and even simulating complex environments for use in scientific research or virtual worlds.

2.1.1 Definition and key concepts of generative models

Generative models are a subset of machine learning models designed to generate new data points. These models do not simply learn to classify or label data; instead, they learn the probabilistic distribution of the data and use that knowledge to produce new instances that resemble the training data. Key concepts in generative models include:

- **Probability Distribution**: Generative models learn the joint probability distribution $P(X, Y)$ of input features X and their corresponding labels Y. By understanding this distribution, the model can generate new data points, predict missing information, or simulate various possible outcomes.

- **Latent Space**: In many generative models, data is often mapped to a lower-dimensional latent space, which encodes the essential features and characteristics of the data. The model then generates new data by sampling from this latent space and transforming the sample back to the original high-dimensional space. This approach is commonly seen in models like Variational Autoencoders (VAEs).

- **Training through** Generative Adversarial Networks (GANs): One of the most prominent approaches in generative AI is through Generative Adversarial Networks (GANs). GANs consist of two neural networks: a generator that creates new data and a discriminator that tries to distinguish between real data from the training set and the data produced by the generator. These two networks are trained in tandem, with the generator improving its output to fool the discriminator, resulting in increasingly realistic generated data.

- **Autoregressive Models**: These models, such as **GPT (Generative Pre-trained Transformer)**, generate data by predicting the next element in a sequence based on previously generated elements. Autoregressive models are widely used in natural language processing tasks like text generation and language translation.

2.1.2 Differences between Discriminative and Generative AI

The distinction between discriminative and generative AI lies in their approach to learning and utilizing data.

- **Discriminative AI** focuses on learning the boundary between different classes in the data. A discriminative model learns to predict the label Y given the features X, modeling the conditional probability $P(Y|X)$. Examples of discriminative models include logistic regression, support vector machines (SVMs), and neural networks used for classification. These models are primarily concerned with distinguishing between different categories or outcomes based on existing data. Discriminative Models are optimized for tasks like classification, where the goal is to predict the correct label for a given input. They excel at drawing clear boundaries between classes, but they cannot generate new data. Discriminative models focus on mapping inputs to outputs without capturing the overall distribution of the data.

- **Generative AI**, on the other hand, models the joint probability distribution $P(X, Y)$. It not only classifies data but also learns how to generate new instances of the data. By learning the underlying structure and distribution of the input data, generative models can create new data that fits the same distribution. For example, a generative model trained on images of faces could generate new, realistic-looking faces. Generative Models are designed to capture the data's internal structure and distribution, allowing them to generate new samples. They can be used not only for classification (by using the joint distribution) but also for tasks like data generation, anomaly detection, and unsupervised learning.

Key Differences:

- **Objective**: Discriminative models classify or label existing data, while generative models focus on generating new data that resembles the training set.

- **Probabilistic Approach**: Discriminative models estimate $P(Y|X)$, whereas generative models estimate $P(X, Y)$, which allows them to generate new instances.

- **Capabilities**: Discriminative models are limited to tasks like classification and regression, whereas generative models can perform additional tasks like data augmentation, unsupervised learning, and generating new, unseen examples.

Discriminative models are often simpler and more efficient for tasks like image classification or sentiment analysis. However, generative models open up new possibilities for creativity, simulation, and unsupervised learning, making them central to many cutting-edge applications in AI, including image synthesis, text generation, and virtual environment creation.

2.2 Applications of Generative AI

Generative AI has rapidly expanded across various fields, showcasing its versatility in creating novel content, improving efficiencies, and offering innovative solutions in areas ranging from art to scientific research. By leveraging its ability to generate realistic and creative data, Generative AI is reshaping industries and inspiring new ways of thinking about content creation, problem-solving, and human-computer interaction. This section explores some of the most prominent applications of generative AI, focusing on examples in art, image synthesis, natural language processing, and more.

2.2.1 Examples in art, image synthesis, natural language processing, and more

Generative AI is being applied in numerous domains, where it has unlocked new capabilities for creativity, automation, and problem-solving. Below are key examples of how it is utilized across various fields:

Art and Creativity

Generative AI has made a significant impact on the world of art, giving rise to new forms of creative expression. AI-powered tools, such as deep learning-based image generators and style transfer algorithms, allow artists to produce artwork that merges their own styles with those learned from vast datasets of existing art.

- **AI-Generated Art**: Tools like **DeepArt** and **Artbreeder** enable users to create entirely new visual artwork by blending styles or generating images from scratch. These platforms utilize generative models, such as **Generative Adversarial Networks (GANs)**, to create images that are not only visually compelling but also novel. Artists have begun to collaborate with AI to co-create artwork, blurring the lines between human and machine creativity.

- **Music and Audio Synthesis**: Generative AI is also transforming the music industry. Models like OpenAI's Jukebox can generate original songs by learning from vast datasets of music, producing lyrics, melodies, and harmonies. AI-based music composition tools provide musicians with a new medium for generating inspiration or automating background scores for films and video games.

Image Synthesis

In the domain of image generation, GANs have emerged as a revolutionary technology, allowing for the creation of highly realistic images from scratch. Image synthesis has several key applications:

- **Deepfakes**: GANs can generate hyper-realistic images and videos by learning from existing visual data. Deepfake technology, while controversial, demonstrates the capability of generative AI to synthesize human faces and voices with remarkable accuracy. Although it poses ethical challenges, it also holds promise for applications like virtual actors, digital avatars, and film production.

- **Super-Resolution and Image Restoration**: Generative AI is also applied to enhance the quality of images. Models such as **SRGAN (Super-Resolution GAN)**[3] can take low-resolution images and generate high-resolution versions by filling in missing details. This technology is widely used in areas like medical imaging, satellite imagery, and improving archival photos and videos.
- **Text-to-Image Generation**: Tools like DALL · E and MidJourney use deep generative models to create images from textual descriptions. Users can input a phrase or sentence, and the AI will generate a highly detailed image based on the description. This has opened up new opportunities in content creation, design, advertising, and entertainment.

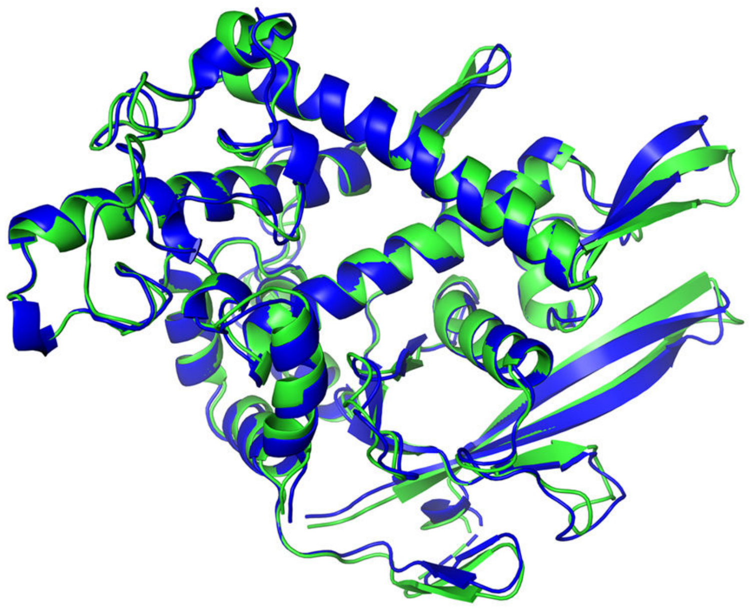
Natural Language Processing (NLP)

Generative AI has made tremendous strides in **Natural Language Processing (NLP)**, particularly in the field of text generation. Large language models, such as GPT (Generative Pre-trained Transformer), have transformed the way machines understand and generate human language, enabling a wide range of applications.

- **Text Generation and Writing Assistance**: GPT models and similar language models can generate coherent and contextually relevant text based on user prompts. These models are used to create blog posts, articles, creative writing, and even code. AI-powered writing tools, such as **Grammarly** and **Copy.ai**, assist users in drafting emails, essays, and social media posts by suggesting improvements and generating alternative content.

- **Conversational Agents and Chatbots**: Virtual assistants like **OpenAI's ChatGPT** and **Google's Bard** utilize generative language models to conduct meaningful, human-like conversations. These chatbots are used in customer service, education, and personal productivity applications, helping users answer questions, schedule appointments, and perform various tasks through natural language interactions.
- **Machine Translation and Summarization**: Generative AI models power sophisticated translation services, enabling seamless communication across languages. Tools like Google Translate and DeepL use deep learning models to generate accurate translations. Additionally, AI is used to automatically summarize lengthy documents, making it easier for users to process large volumes of text efficiently.

Scientific Research and Simulation



Generative AI is also making significant contributions to scientific research by helping researchers simulate complex processes, generate synthetic data, and accelerate discovery:

- **Molecular and Drug Discovery**: In the pharmaceutical and chemical industries, generative models are used to predict the structure of new molecules and simulate their behavior. Generative models like **DeepMind's AlphaFold**[4] can predict protein folding structures, aiding in drug discovery and the understanding of biological systems. This has the potential to drastically reduce the time and cost required for developing new treatments.

- **Climate Modeling and Data Augmentation**: In environmental science, generative models are used to simulate climate scenarios, predict weather patterns, and model ecosystems. These models help researchers fill in missing data or simulate hypothetical scenarios that can aid in disaster preparedness and environmental conservation efforts.

Gaming and Virtual Environments

Generative AI is transforming the gaming industry by enabling the creation of realistic and dynamic virtual environments. Procedural content generation powered by AI allows developers to create immersive game worlds that adapt to player interactions:

- **Procedural Content Generation**: AI models can autonomously generate game levels, characters, and storylines based on predefined rules or datasets. This allows for more personalized and engaging gaming experiences, as well as reducing the manual labor required to design complex virtual environments.

- **NPCs and Interactive Dialogue**: Generative AI is used to create more lifelike non-playable characters (NPCs) with dynamic dialogue and behavior. By using natural language models, NPCs can respond more naturally to player actions, creating a more immersive and interactive gaming experience.

2.3 Ethical and Societal Implications

As generative AI continues to advance, it brings with it profound ethical and societal challenges. While the ability of AI to generate novel content offers tremendous opportunities, it also raises critical questions about its responsible use, potential misuse, and long-term impacts on society. This section explores the ethical and societal concerns surrounding generative AI, focusing on the challenges of bias, misinformation, and deepfakes, as well as the broader consequences for privacy, employment, and trust in digital content.

2.3.1 Challenges and concerns in the use of generative AI

Generative AI introduces several complex ethical issues that require careful consideration, particularly in terms of bias, authenticity, and misuse. Below are some of the key challenges and concerns in the use of this technology:

Bias and Fairness in Generative AI

Bias in AI models is a significant ethical concern, especially in generative AI, which learns from large datasets that may contain underlying biases. These biases can be unintentionally propagated, or even amplified, by AI models, leading to unfair or discriminatory outcomes.

- **Training Data Bias**: Generative models, such as those used in natural language processing or image generation, are trained on vast amounts of data sourced from the internet or other repositories. If the training data reflects societal biases—such as gender, racial, or cultural stereotypes—the AI can generate outputs that reinforce these biases. For instance, AI-generated art might disproportionately represent certain ethnicities or genders in particular roles, and language models could produce biased or harmful text reflecting prejudiced attitudes present in the data.
- **Algorithmic Amplification of Bias**: Once trained on biased data, generative AI can unintentionally amplify these biases when creating new content. For example, language models might perpetuate biased narratives, and image generators could produce outputs that disproportionately exclude or misrepresent marginalized groups. Addressing this issue requires not only more diverse and representative datasets but also the development of techniques to actively mitigate bias during model training and generation.

Bias in generative AI poses a risk in many domains, including automated content generation, media creation, and decision-making systems. Ensuring fairness, transparency, and inclusivity in AI systems is crucial to avoid perpetuating existing inequalities.

Misinformation and Deepfakes

One of the most alarming applications of generative AI is its potential for creating highly realistic yet fake content, especially in the form of deepfakes. Deepfakes are AI-generated images, videos, or audio recordings that convincingly imitate real people, often with malicious intent.

- **Misinformation and Disinformation**: Deepfakes can be used to spread misinformation by fabricating events, statements, or actions involving public figures. This has serious implications for politics, media, and public trust. For example, a deepfake video of a politician making inflammatory remarks could spark social unrest, undermine democratic processes, or manipulate public opinion.

- **Erosion of Trust**: As generative AI becomes more adept at creating indistinguishable fake content, it becomes harder to discern what is real and what is fabricated. This erosion of trust in digital media can have far-reaching effects on society, including diminishing confidence in journalism, public discourse, and even legal evidence.

- **Misuse for Personal Harm**: Beyond public figures, deepfakes can also be weaponized to harm individuals through non-consensual content, such as deepfake pornography or maliciously altered videos intended to damage someone's reputation. These unethical uses of generative AI pose significant risks to personal privacy and safety.

To mitigate the risks associated with deepfakes and misinformation, researchers are developing deepfake detection algorithms and advocating for stricter regulatory frameworks. However, the rapid pace of AI advancement means that countermeasures often lag behind the technology, leaving society vulnerable to new forms of digital manipulation.

Intellectual Property and Ownership

Generative AI blurs the lines between human creativity and machine output, raising complex legal and ethical questions around intellectual property (IP) and content ownership.

- **Authorship and Attribution**: When AI models generate art, music, text, or code, it is often unclear who should be credited as the author. Should the creator of the model, the user who inputted the prompts, or the AI itself receive recognition? This ambiguity in authorship creates challenges in determining copyright and IP rights, especially when generative AI is used commercially.

- **Training on Copyrighted Data**: Many generative models are trained on datasets that include copyrighted materials, such as images, music, or text. This raises concerns about the legality of using such models to generate new content that may resemble or incorporate elements of the original works. Legal frameworks have yet to fully address the implications of using copyrighted material in AI training processes, creating uncertainty around the use and ownership of AI-generated content.

Privacy Concerns

Generative AI also introduces concerns related to privacy and data security, particularly in the context of generating personal data or imitating individuals without consent.

- **Synthetic Data Generation**: AI can generate realistic synthetic data, including personal details such as faces, voices, or other identifiable information. This raises privacy concerns when AI-generated content is used to impersonate individuals or when personal data is synthesized without explicit permission.
- **Data Collection for Model Training**: Generative AI models often require large datasets for training, which may include personal information scraped from public and private sources. If the data is not properly anonymized, individuals' privacy could be compromised, especially in models designed to simulate personal identities or behaviors.

Societal and Economic Impact

Generative AI could also have broad societal and economic implications, affecting employment, the creative economy, and social interactions:

- **Job Displacement**: As generative AI becomes more capable of creating content autonomously, there is growing concern about its impact on industries such as journalism, art, design, and even programming. AI systems that can generate articles, design graphics, or write code could displace human workers in these fields, leading to potential job losses and economic disruption.
- **Redefinition of Creativity and Human Roles**: Generative AI challenges traditional notions of creativity and originality, raising questions about what it means to be a creator in the age of AI. While some see AI as a tool for augmenting human creativity, others worry that it may diminish the value of human contribution in creative processes, shifting societal perceptions of art, literature, and other forms of expression.

Chapter 3

Introduction to Generative Models

Generative models are a cornerstone of modern artificial intelligence, focusing on learning the underlying patterns of data to create new, realistic outputs. Unlike discriminative models, which aim to classify data points, generative models capture the probability distribution of a dataset, allowing them to generate new data points similar to those in the original dataset. This section provides an overview of generative models and delves into the most popular types, including Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion models.

3.1 Overview of Generative Models

Generative models are designed to model the distribution of data $P(X)$, where X represents the features of the dataset. By learning this distribution, these models can generate new data points that are statistically similar to the original training data. This is in contrast to discriminative models, which learn $P(Y|X)$, the probability of a label Y given the input features X.
The ability of generative models to create novel data has made them pivotal in various applications, including image synthesis, text generation, and even drug discovery. Key applications include:

- **Image Generation**: Creating realistic images from random noise or based on text descriptions.

- **Data Augmentation**: Generating additional training data to improve machine learning models.

- **Text Generation**: Producing coherent and contextually accurate text in natural language processing tasks.

Several types of generative models have emerged, each with unique architectures and mechanisms for generating data. The most prominent among them are Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models.

3.1.1 Types of generative models: GANs, VAEs, and Diffusion models

Generative models come in various forms, with each type having distinct architectures and methods for generating data. Below, we explore three of the most commonly used generative models in modern AI: GANs, VAEs, and Diffusion Models.

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)[5], introduced by Ian Goodfellow and his colleagues in 2014, represent a class of generative models that use two neural networks—a generator and a discriminator—trained together in an adversarial setup.

- **Architecture**: GANs consist of two components:
  - The **Generator** learns to produce realistic data by mapping random noise (usually sampled from a Gaussian distribution) to the data distribution.
  - The **Discriminator** is a classifier that attempts to distinguish between real data (from the training set) and fake data (generated by the generator).

  These two models are trained in a competitive game: the generator improves by learning to "fool" the discriminator, while the discriminator improves by better detecting fake data. Over time, the generator becomes proficient at producing high-quality, realistic data.

- **Applications**: GANs have been revolutionary in areas such as:
  - **Image synthesis**: Creating realistic faces, landscapes, and objects.
  - **Deepfakes**: Generating realistic yet artificially created videos or images of individuals.
  - **Data augmentation**: Increasing the size of training datasets with synthetic data.

- **Challenges**: While GANs produce highly realistic outputs, they are notorious for instability during training and issues like mode collapse, where the generator produces limited types of outputs.

Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs)[6], introduced in 2013 by Kingma and Welling, are a type of generative model that uses an encoder-decoder architecture to learn a latent space representation of the input data, from which new data points can be generated.

- **Architecture**: VAEs consist of two main components:
  - The encoder maps input data to a probability distribution in a latent space, typically parameterized as a multivariate Gaussian distribution. Each input is encoded into a mean and variance, which define the distribution of the latent variable.
  - The decoder then samples from this latent space and reconstructs the input data.

  The key innovation in VAEs is the use of variational inference, a technique that forces the learned latent space to follow a known distribution (usually Gaussian), enabling smooth interpolation and sampling from the latent space.
- **Applications**: VAEs are widely used in tasks like:
  - Data compression: Compressing data into a lower-dimensional latent representation.
  - Image generation: Generating images by sampling from the learned latent space.
  - Anomaly detection: Detecting outliers by analyzing how well the model can reconstruct data points.
- **Challenges**: While VAEs are more stable and easier to train than GANs, they often produce blurrier outputs due to the smoothness imposed on the latent space, making them less ideal for tasks requiring high fidelity.

Diffusion Models

Diffusion models are a relatively newer class of generative models that have gained popularity for their ability to generate high-quality images. These models are based on a **probabilistic diffusion process**, where data is gradually transformed into noise, and the model learns to reverse this process to generate new samples.

- **Architecture**: Diffusion models work by slowly adding noise to training data in a step-by-step manner, eventually converting the data into pure noise. The model is then trained to reverse this process, learning how to gradually "denoise" the noisy data to recover realistic samples.
  - This process of learning to denoise makes diffusion models highly effective at capturing fine details in the generated data, often outperforming GANs in terms of image quality.
- **Applications**: Diffusion models are used in:
  - **Image synthesis**: Producing high-resolution images with fine details.
  - **Text-to-image generation**: Models like **DALL · E 2** and **Stable Diffusion**[7] use this approach to generate images from textual descriptions.
- **Advantages and Challenges**: Diffusion models often outperform GANs in generating photorealistic images and are more stable during training. However, they tend to be computationally expensive and slow during the generation process, as the denoising process requires multiple steps.

3.2 GANs (Generative Adversarial Networks)

Generative Adversarial Networks (GANs)[5] have emerged as one of the most influential types of generative models in AI, excelling in tasks such as image synthesis, data augmentation, and content generation. Introduced by Ian Goodfellow and his collaborators in 2014, GANs are designed around the concept of adversarial training, where two neural networks—the generator and the discriminator—compete against each other. This section delves into the architecture of GANs and explores the challenges associated with training them, including mode collapse and instability.

3.2.1 Architecture: Generator vs. Discriminator

At the heart of every GAN is a **dual-network architecture**, consisting of two neural networks: the generator and the discriminator. These networks play a zero-sum game, where the success of one depends on the failure of the other, leading to a dynamic and competitive learning process.

- Generator: The generator's primary objective is to create realistic data that can fool the discriminator. It takes in a random vector, usually sampled from a simple probability distribution (such as Gaussian noise), and maps it to the data space, aiming to generate outputs indistinguishable from real data.

  - **Input**: A random noise vector (e.g., from a Gaussian distribution).
  - **Output**: Synthetic data resembling real examples from the training set (e.g., images, text, or other structured data).
  - **Objective**: Minimize the discriminator's ability to correctly classify generated data as fake, effectively improving its capacity to generate realistic outputs.

- **Discriminator**: The discriminator is a classifier that attempts to distinguish between real data (from the training set) and fake data (generated by the generator). Its role is to improve its ability to detect generated (fake) data while recognizing authentic (real) data.
  - **Input**: A mix of real data and synthetic data produced by the generator.
  - **Output**: A probability score indicating whether the input is real or fake (usually between 0 and 1).
  - **Objective**: Maximize the probability of correctly classifying real data as real and fake data as fake, essentially making it harder for the generator to fool it.

- **Adversarial Training Process**: GANs rely on an adversarial learning process where both the generator and the discriminator improve iteratively:
  - The generator learns to generate increasingly realistic data by trying to "trick" the discriminator.
  - The discriminator learns to improve its ability to distinguish between real and generated data.

  Mathematically, the training process can be represented as a minimax game, where the generator tries to minimize the discriminator's ability to classify fake data, and the discriminator aims to maximize its success in detecting fakes.
  This can be expressed by the following objective function:

  $$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim\mathbb{P}_{data}(x)[\log D(x)]} + \mathbb{E}_{z\sim\mathbb{P}_z(z)}[log\,(1-D(G(z)))]$$

  Here, G represents the generator, D is the discriminator, $\mathbb{P}_{data}(x)$ is the distribution of real data, and $\mathbb{P}_z(z)$ is the distribution of the noise input.

3.2.2 Challenges with GANs (mode collapse, training instability)

Despite their groundbreaking success, GANs are notoriously difficult to train, often facing several key challenges that can hinder their performance. Among these, mode collapse and training instability are the most prominent issues.

Mode Collapse

Mode collapse occurs when the generator fails to capture the full diversity of the data distribution, producing only a limited set of outputs. Instead of generating a wide range of data points, the generator repeatedly outputs very similar or identical samples, which reduces the overall quality and variety of generated content.

- **Cause**: Mode collapse typically arises when the generator finds a way to consistently fool the discriminator by producing a small subset of realistic outputs. Although these outputs may initially deceive the discriminator, they lack diversity, failing to represent the entire distribution of the data.
- **Example**: In the case of image generation, mode collapse might result in the generator producing images of the same object or person repeatedly, even though the training dataset contains diverse examples.
- **Solutions**: To address mode collapse, several techniques have been developed, including:
  - **Minibatch discrimination**: This technique introduces diversity by encouraging the generator to produce a wider variety of outputs. It does this by comparing groups of generated samples, discouraging the generator from producing identical or highly similar outputs.
  - **Unrolled GANs**: This method helps the generator anticipate the future steps of the discriminator during training, reducing the likelihood of collapsing onto a single mode.

Training Instability

GANs are also known for their **training instability**, which can result from the adversarial nature of their training process. Unlike traditional neural networks, GANs require the careful balancing of two models, each with opposing objectives. This creates several challenges that can lead to erratic training behavior.

- **Non-convergence**: Since the generator and discriminator are constantly improving to outsmart each other, the training process can sometimes fail to converge to a stable equilibrium. Instead, the generator and discriminator may enter into oscillating patterns where neither model consistently improves.
- **Vanishing Gradient Problem**: If the discriminator becomes too strong, it easily identifies the generator's outputs as fake, providing little useful feedback to the generator. As a result, the generator struggles to improve, and the training process stalls due to vanishing gradients—when the error signal that guides learning becomes too small for the generator to update meaningfully.

- **Exploding Gradients**: On the other hand, if the discriminator is too weak, it fails to provide meaningful feedback, causing the generator to produce poor-quality outputs. This imbalance can result in exploding gradients, where the model updates are too large, leading to unstable and erratic training.
- **Solutions**: Addressing training instability often involves carefully tuning the architecture and optimization process. Some solutions include:
  - **Improved loss functions**: Techniques like **Wasserstein GAN**[8] modify the loss function to improve the stability of training by measuring the distance between the real and generated data distributions more effectively.
  - **Gradient penalty**: Adding a gradient penalty can stabilize training by ensuring that the discriminator's gradients remain within a reasonable range, preventing both vanishing and exploding gradients.
  - **Feature matching**: Instead of optimizing to directly fool the discriminator, the generator can be trained to match the features of real data at an intermediate layer in the discriminator, which often leads to more stable learning and less aggressive competition between the two models.

3.3 Diffusion Models

Diffusion models[9] are a relatively new class of generative models that have gained significant attention due to their ability to produce high-quality, diverse data and their more stable training processes compared to Generative Adversarial Networks (GANs). These models are based on the idea of modeling the gradual transformation of data into noise, then learning to reverse this process to generate realistic data. This section covers the physical origins of diffusion models, the core concept of forward and reverse diffusion processes, and the strengths of diffusion models over GANs, particularly in terms of diversity and stability.

3.3.1 Physical Origins

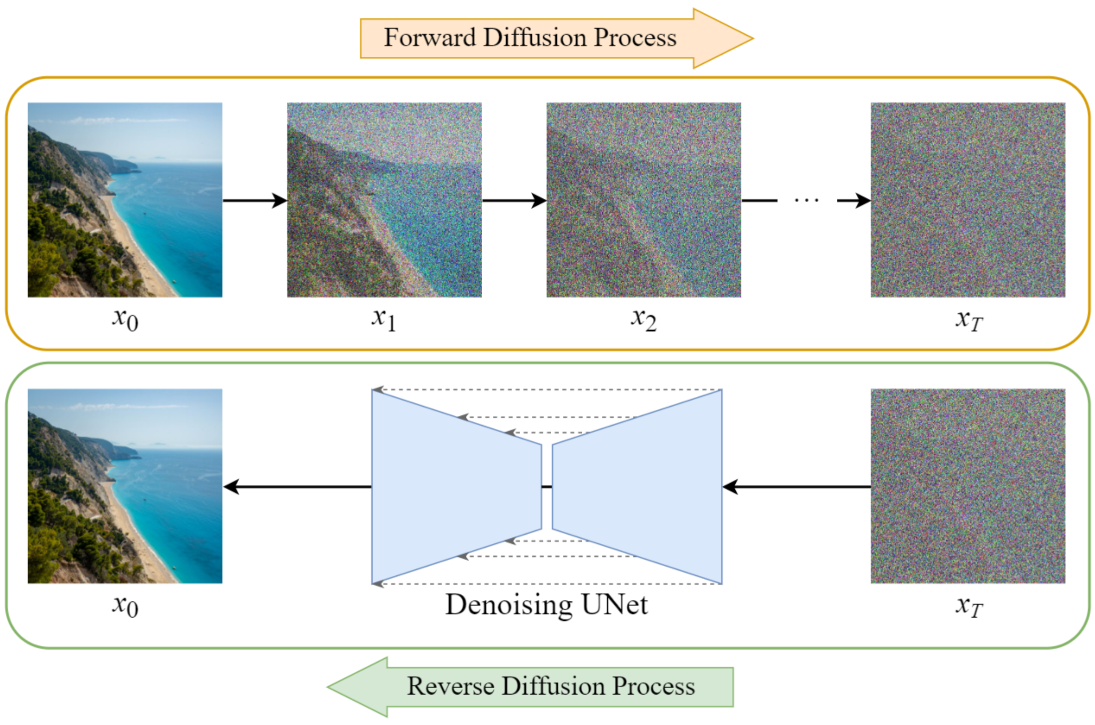
The underlying principles of diffusion models are rooted in thermodynamics and statistical physics, particularly in processes like Brownian motion and heat diffusion. In physics, diffusion refers to the process by which particles spread out from areas of high concentration to areas of low concentration, such as when molecules in a gas move from a dense region into empty space. In generative modeling, the concept of diffusion refers to gradually introducing noise into data and then learning how to reverse this noisy transformation.

- **Brownian Motion and Heat Diffusion**: These physical phenomena involve particles undergoing random motion as they interact with their environment.

Over time, this leads to the gradual "diffusion" of particles until they reach a state of equilibrium. Similarly, diffusion models in AI simulate this process by progressively corrupting data with noise and then reversing the process to recover the original, structured data.

- **Connection to Variational Inference**: From a probabilistic perspective, diffusion models can be seen as a form of variational inference, where the model learns a probabilistic path from noise to data. This approach draws from the physical intuition that complex systems can be described probabilistically and can evolve through small, reversible steps between different states.

3.3.2 Concept: Forward and reverse diffusion processes



The core mechanism of diffusion models revolves around two complementary processes: the forward diffusion process and the reverse diffusion process. Together, these processes enable the generation of new data by systematically introducing and then removing noise.

Forward Diffusion Process

In the **forward diffusion process**, noise is gradually added to real data through a series of steps, transforming it into pure noise by the end of the process. Each step introduces a small amount of Gaussian noise, progressively corrupting the structure of the original data.

- **Process Description**: Starting from a data point x_0 (which could be an image, text, or other types of data), the model adds noise at each time step t, resulting in a sequence of increasingly noisy data points $x_1, x : 2, \ldots, x_T$ until the data is indistinguishable from pure Gaussian noise at time step T. Mathematically, this forward process can be expressed as:

  $$x_t = \sqrt{1-\beta_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t$$

  where β_t controls the amount of noise added at each step, and ϵ_t is Gaussian noise.

- **Goal of Forward Diffusion**: The purpose of the forward diffusion process is to transform the data into a latent space of pure noise. Once this transformation is complete, the reverse process can be applied to generate new samples.

Reverse Diffusion Process

The **reverse diffusion process** aims to undo the forward process by removing noise step by step, eventually recovering realistic data from the pure noise state.

- **Process Description**: Starting from a sample of pure noise, the model learns to reverse each step of the forward diffusion process. This involves learning a parameterized function that predicts the distribution of the previous state x_{t-1} given the current noisy state x_t, allowing the model to "denoise" the data over time. the reverse process can be written as:

  $$x_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \left(x_t - \sqrt{\beta_t \cdot \epsilon_\theta(x_t, t)} \right) + \epsilon$$

  where ϵ_θ represents the model's learned noise estimation, ϵ is Gaussian noise.

- Learning the Reverse Process: The key challenge for diffusion models is learning the reverse process, which involves approximating the conditional distribution of each step. This is typically done through neural networks that predict how to remove the noise, based on the current state and the time step t.

Sampling and Generation

Once the reverse diffusion process has been trained, new samples can be generated by starting with pure noise and progressively denoising it using the learned model. The end result is a data point that resembles the original data distribution, such as a generated image or text sample.

3.3.3 Strengths over GANs: improved diversity and stability

Diffusion models have shown several advantages over GANs, particularly in terms of **diversity** of outputs and **training stability**. These strengths have made diffusion models a competitive alternative to GANs in fields such as image synthesis and natural language generation.

Improved Diversity

One of the major drawbacks of GANs is mode collapse, where the generator produces limited variations of data, reducing the diversity of the outputs. Diffusion models, by contrast, excel at generating diverse samples due to their probabilistic nature and the smoothness of the latent space they learn.

- **Diverse Output Generation**: Diffusion models are less prone to mode collapse because they model the entire data distribution rather than attempting to trick a discriminator, as in GANs. This results in the generation of a wide range of outputs that better reflect the diversity of the training data.

- **Smooth Latent Space**: The stepwise nature of diffusion models creates a smoother latent space, enabling the generation of intermediate data points that reflect subtle variations. This leads to more continuous and diverse sampling, even when small changes are made to the noise input.

Enhanced Training Stability

Training GANs can be notoriously unstable due to the adversarial nature of the generator-discriminator interaction. GANs often face issues such as exploding gradients, vanishing gradients, and non-convergence. Diffusion models, however, avoid these issues by relying on a simpler and more stable learning process.

- **No Adversarial Training**: Unlike GANs, diffusion models do not involve adversarial training between two networks, which eliminates the instability caused by the competition between the generator and the discriminator. As a result, diffusion models tend to converge more reliably during training.

- **Well-Defined Objective Function**: Diffusion models benefit from having a well-defined, log-likelihood-based objective function, making the optimization process more straightforward compared to the minimax game in GANs. This leads to fewer issues with oscillations or collapse during training.

High-Quality Output

Diffusion models are capable of producing **high-resolution** and **detailed images**, often surpassing GANs in terms of visual quality. This is because the reverse diffusion process enables the model to focus on fine details during the denoising steps, gradually refining the output.

- **Fine-Grained Generation**: The gradual nature of the reverse diffusion process allows diffusion models to capture intricate details in data generation. This can result in crisper, more realistic images compared to those generated by GANs, which sometimes produce artifacts due to unstable training.

Diffusion models represent a powerful class of generative models that offer significant advantages over GANs, particularly in terms of improved diversity and stability. By leveraging the forward and reverse diffusion processes, these models can generate high-quality, diverse samples while avoiding many of the training challenges associated with GANs. Their origins in physical processes like diffusion and their probabilistic foundation make them an exciting area of research and application in generative AI.

Chapter 4

Diffusion Models and Latent Diffusion

4.1 Introduction to Diffusion Models

In the domain of machine learning, diffusion models, also referred to as diffusion probabilistic models or score-based generative models, represent a class of latent variable generative models. These models aim to model a diffusion process for a given dataset, enabling the generation of new elements that follow the same distribution as the original data. A diffusion model typically comprises three major components: the forward process, the reverse process, and the sampling procedure. The overall objective is to learn a process that generates data by simulating a random walk with drift through the space of all possible data points.
Diffusion models have several equivalent formalisms, including Markov chains, denoising diffusion probabilistic models (DDPM), noise-conditioned score networks, and stochastic differential equations (SDEs) . These models are predominantly trained using variational inference techniques . The model tasked with denoising during the reverse process is often referred to as its "backbone", and typical choices for this architecture include U-nets or transformers.

4.1.1 Applications of Diffusion Models

As of 2024, diffusion models are primarily used in computer vision tasks such as image denoising, inpainting, super-resolution, image generation, and video generation. The standard procedure in these applications involves training a neural network to iteratively denoise images that have been blurred by Gaussian noise . Once trained, these models are capable of generating images by starting from a noise-filled image and applying the learned denoising process.

One prominent application is in image generation, where diffusion-based image generators such as Stable Diffusion and DALL-E have gained significant commercial interest. These models often integrate diffusion techniques with text-encoders and cross-attention modules to enable text-conditioned image generation, making them versatile tools for creative applications.
Beyond computer vision, diffusion models have also made inroads in natural language processing (NLP) for tasks like text generation , summarization , and more recently in areas such as sound generation and reinforcement learning.

4.2 Denoising Diffusion Models

4.2.1 Non-equilibrium Thermodynamics

Non-equilibrium thermodynamics is a field that focuses on systems that are not in thermodynamic equilibrium. Unlike equilibrium thermodynamics, which deals with systems in a steady state, non-equilibrium thermodynamics describes systems that evolve over time and can be characterized by macroscopic variables that extend those used in equilibrium conditions. This branch is particularly concerned with **transport processes** and the **rates of chemical reactions**.
Most natural systems are not in equilibrium; they are constantly changing or can be altered by external forces. These systems are subject to fluxes of matter and energy and undergo chemical reactions, both continuously and intermittently. Many of these processes can be locally approximated as being in equilibrium, allowing them to be analyzed with the tools of equilibrium thermodynamics. However, some systems remain beyond the reach of equilibrium methods due to **non-variational dynamics**, where the concept of free energy no longer applies.
The study of non-equilibrium systems requires broader principles than those found in equilibrium thermodynamics. One major distinction lies in how inhomogeneous systems behave. Unlike homogeneous systems studied in equilibrium thermodynamics, non-equilibrium systems require understanding reaction rates. Additionally, defining entropy at any given moment in non-equilibrium systems is challenging in macroscopic terms, though it can be defined locally. In such cases, the total entropy of the system can be described as the integral of these localized entropy densities. Remarkably, even systems far from global equilibrium often still follow local equilibrium rules.
In 2015, diffusion models were introduced as a way to learn from highly complex probability distributions, utilizing principles from non-equilibrium thermodynamics, particularly diffusion.
Consider the example of modeling the distribution of all natural images. Each photo can be represented as a point in the space of all possible images, where the distribution forms a "cloud." By adding noise repeatedly to the images, the cloud

diffuses throughout the entire image space, eventually resembling a **Gaussian distribution** $N(0, I)$.
A model capable of reversing this diffusion process can sample from the original distribution. This process is relevant to non-equilibrium thermodynamics because the initial distribution is not in equilibrium, while the final distribution is.
The equilibrium distribution in this context is the **Gaussian distribution** $N(0, I)$, with the **probability density function** $\mathbb{P}(x) \propto e^{-\frac{1}{2}\|x\|^2}$. This mirrors the **Maxwell-Boltzmann distribution** for particles in a potential well $V(x) = \frac{1}{2}\|x\|^2$ at temperature 1. The initial distribution, being far from equilibrium, diffuses toward this equilibrium state through biased random steps. These steps are a combination of pure randomness (like **Brownian motion**) and a gradient descent toward the potential well. Randomness is crucial here: without it, the particles would collapse into a single point at the origin, eliminating the distribution entirely.

4.2.2 Denoising Diffusion Probabilistic Model (DDPM)

The Denoising Diffusion Probabilistic Model (DDPM), introduced in a 2020 paper, is an improvement on prior generative models using variational inference techniques.

Forward Diffusion Process

The forward diffusion process gradually adds noise to an initial data point, denoted as $x_0 \sim q$, where q is the distribution we want to learn. The noise is added step by step until the data is completely transformed into pure noise, specifically a Gaussian distribution $N(0, I)$.
To explain this process, we introduce several variables:

- $\beta_1, \ldots, \beta_T \in (0,1)$ are predefined constants that control the noise scale.
- $\alpha_t := 1 - \beta_t$
- $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$
- $\sigma_t := \sqrt{1 - \bar{\alpha}_t}$
- $\tilde{\sigma}_t := \dfrac{\sigma_{t-1}}{\sigma_t}\sqrt{\beta_t}$
- $\tilde{\mu}_t(x_t, x_0) := \dfrac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)x_0}{\sigma_t^2}$
- $N(\mu, \Sigma)$ is the normal distribution with mean μ and variance Σ, and $N(x|\mu, \Sigma)$ is the probability density at x.

at each timestep t, we apply the transformation that adds noise repeatedly:

$$x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t} z_t$$

where $z_1, \dots, z_T \sim N(0, I)$ are independent and identically distributed (IID) Gaussian noise sample. As t increases, x_t converges towards a standard Gaussian distribution $N(0, I)$, ensuring that the original data x_0 becomes indistinguishable from noise after enough steps:

$$\lim_{t\to\infty} x_t | x_0 \to N(0, I)$$

The full process can be written as:

$$q(x_{0:T}) = q(x_0) \prod_{t=1}^{T} q(x_t|x_{t-1}) = q(x_0) \prod_{t=1}^{T} N(x_t|\sqrt{\alpha_t} x_{t-1}, \beta_t I)$$

Here, each transition $q(x_t|x_{t-1})$ is a Gaussian distribution with mean $\sqrt{\alpha_t} x_{t-1}$ and variance $\beta_t I$.
It can be rewritten as:

$$\ln q(x_{0:T}) = \ln q(x_0) - \sum_{t=1}^{T} \frac{1}{2\beta_t} \left\| x_t - \sqrt{1-\beta} x_{t-1} \right\|^2 + C$$

It can be osserved $x_{1:T}|x_0$ form a Gaussian process, allowing for flexibility in reparametrization.
Specifically, through standard Gaussian process manipulations, we have:

$$x_t|x_0 \sim N(\sqrt{\bar{\alpha}_t} x_0, \sigma_t^2 I)$$

and

$$x_{t-1}|x_t, x_0 \sim N(\tilde{\mu}_t(x_t, x_0), \tilde{\sigma}_t^2 I)$$

Notice that as t increases, $x_t|x_0 \sim N(\sqrt{\bar{\alpha}_t} x_0, \sigma_t^2 I)$ converges to $N(0, I)$. This implies that after a sufficiently long diffusion process, the variable x_T approaches $N(0, I)$, effectively erasing any information from the original $x_0 \sim q$.
Since $x_t|x_0 \sim N(\sqrt{\alpha}_t x_0, \sigma_t^2 I)$ we can directly sample $x_t|x_0$ in one step without needing to compute the intermediate steps $x_1, x_2, \dots, x_{t-1}$.

Reparameterization Derivation

We begin with the knowledge that $x_{t-1}|x_0$ and $x_t|x_{t-1}$ are both Gaussian distributions and independent of each other.
Therefore, we can express the following reparameterization:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}}z$$
$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}z'$$

where z and z' are independent, identically distributed **Gaussian random variables**.
Now, we have five variables: x_0, x_{t-1}, x_t, z, and z', governed by two linear equations. The randomness comes from z and z', which can be reparameterized through rotation, taking advantage of the **rotational symmetry** of the IID Gaussian distribution.

First Reparameterization

By substituting x_{t-1} into the equation for x_t, we can simplify:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \underbrace{\sqrt{\alpha_t - \bar{\alpha}_t}z + \sqrt{1-\alpha_t}z'}_{\sigma_t z''}$$

Here, z'' is a new Gaussian variable with zero mean and unit variance, representing a linear combination of z and z'.

Second Reparameterization via Rotation

To find the **second reparameterization**, we rotate the Gaussian variables z and z' using a **rotational matrix**. The relationship between the new variables z'' and z''' is given by:

$$\begin{bmatrix} z'' \\ z''' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\alpha_t-\bar{\alpha}_t}}{\sigma_t} & \frac{\sqrt{\beta_t}}{\sigma_t} \\ -\frac{\sqrt{\beta_t}}{\sigma_t} & \frac{\sqrt{\alpha_t-\bar{\alpha}_t}}{\sigma_t} \end{bmatrix} \begin{bmatrix} z \\ z' \end{bmatrix}$$

Since the inverse of a rotational matrix is its transpose, we can solve for z and z':

$$\begin{bmatrix} z \\ z' \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{\alpha_t-\bar{\alpha}_t}}{\sigma_t} & -\frac{\sqrt{\beta_t}}{\sigma_t} \\ \frac{\sqrt{\beta_t}}{\sigma_t} & \frac{\sqrt{\alpha_t-\bar{\alpha}_t}}{\sigma_t} \end{bmatrix} \begin{bmatrix} z'' \\ z''' \end{bmatrix}$$

Final Expressions

Substituting back into the original equations, we arrive at the final simplified forms:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sigma_t z''$$

$$x_{t-1} = \tilde{\mu}_t(x_t, x_0) - \tilde{\sigma}_t z'''$$

These equations describe the **reparameterized diffusion process**, with the randomness encoded in the rotated Gaussian variables z'' and z'''.

Backward Diffusion

The core idea behind DDPM is to train a neural network, parameterized by θ, to reverse the forward diffusion process. The network takes two inputs: x_t and t, and outputs a mean vector $\mu_\theta(x_t, t)$ and a covariance matrix $\Sigma_\theta(x_t, t)$. These outputs are used to approximate the reverse process by sampling x_{t-1} from a **Gaussian distribution**:

$$x_{t-1} \sim N(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

This defines the **backward diffusion process** p_θ as follows:

$$p_\theta(x_T) = N(x_T|0, I)$$

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}|\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

The goal is to learn the parameters θ such that the distribution $p_\theta(x_0)$ closely matches the data distribution $q(x_0)$. To achieve this, we use **maximum likelihood estimation via variational inference**.

Variational Inference

The **Evidence Lower Bound (ELBO)**[10] inequality provides the foundation for optimizing this process:

$$\ln p_\theta(x_0) \geq \mathbb{E}_{x_{1:T} \sim q(\cdot|x_0)} [\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$$

Taking the **expectation** over the data distribution $q(x_0)$, we get:

$$\mathbb{E}_{x_0 \sim q} [\ln p_\theta(x_0)] \geq \mathbb{E}_{x_{0:T} \sim q} [\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$$

Maximizing the right-hand side provides a lower bound on the **likelihood** of the observed data, allowing us to perform **variational inference**.

Loss Function

We define the **loss function** $L(\theta)$ as:

$$L(\theta) := -\mathbb{E}_{x_{0:T} \sim q} [\ln p_\theta(x_{0:T}) - \ln q(x_{1:T}|x_0)]$$

The goal is to minimize this loss using **stochastic gradient descent**. Simplifying the loss, we obtain:

$$L(\theta) = \sum_{t=1}^{T} \mathbb{E}_{x_{t-1}, x_t \sim q} [-\ln p_\theta(x_{t-1}|x_t)] + \mathbb{E}_{x_0 \sim q} [D_{KL}(q(x_T|x_0) \parallel p_\theta(x_T))] + C$$

Since $p_\theta(x_T) = N(x_T|0, I)$ is parameter-independent, the **Kullback-Leibler divergence** term can be ignored. This reduces the **loss function** to:

$$L(\theta) = \sum_{t=1}^{T} L_t$$

with each L_t defined as:

$$L_t = \mathbb{E}_{x_{t-1},x_t \sim q}\left[-\ln p_\theta(x_{t-1}|x_t)\right]$$

The overall goal is to minimize this loss $L(\theta)$ over time.

Noise Prediction Network

In the DDPM framework, we model the reverse process as:

$$x_{t-1}|x_t, x_0 \sim N(\tilde{\mu}_t(x_t, x_0), \tilde{\sigma}_t^2 I)$$

This suggests that the reverse mean $\mu_\theta(x_t, t)$ should ideally match $\tilde{\mu}_t(x_t, x_0)$. However, since the model doesn't have direct access to the original data point x_0, it must estimate it. We know that:

$$x_t|x_0 \sim N(\sqrt{\bar{\alpha}_t}x_0, \sigma_t^2 I)$$

which can be rewritten as:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sigma_t z$$

where z is some unknown Gaussian noise.
Therefore, estimating x_0 becomes equivalent to estimating the noise z.
To address this, the network is designed to output a noise vector $\epsilon_\theta(x_t, t)$, predicting the noise component directly. With this, we can express $\mu_\theta(x_t, t)$ as:

$$\mu_\theta(x_t, t) = \tilde{\mu}_t\left(x_t, \frac{x_t - \sigma_t \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) = \frac{x_t - \epsilon_\theta(x_t, t)\beta_t/\sigma_t}{\sqrt{\alpha_t}}$$

Next, we design $\Sigma_\theta(x_t, t)$, the covariance matrix. The original DDPM paper found that learning this covariance matrix led to unstable training and poorer sample quality. Instead, they fixed it to a predefined value:

$$\Sigma_\theta(x_t, t) = \zeta_t^2 I$$

where ζ_t^2 can either be β_t or $\tilde{\sigma}_t^2$, both yielding similar performance.

Loss Function simplification

The loss function for the model simplifies to:

$$L_t = \frac{\beta_t^2}{2\alpha_t \sigma_t^2 \zeta_t^2} \mathbb{E}_{x_0 \sim q; z \sim N(0,I)} \left[\|\epsilon_\theta(x_t, t) - z\|^2 \right] + C$$

This can be minimized using stochastic gradient descent. Empirically, the DDPM paper found that an even simpler loss function produced better results:

$$L_{simple,t} = \mathbb{E}_{x_0 \sim q; z \sim N(0,I)} \left[\|\epsilon_\theta(x_t, t) - z\|^2 \right]$$

This simpler formulation significantly improves model performance and stability during training.

Backward Diffusion Process

Once the noise prediction network is trained, it can be used to generate data points from the original distribution through an iterative process. The steps for each iteration are as follows:

1. **Estimate the noise**:
$$\epsilon \leftarrow \epsilon_\theta(x_t, t)$$
Here, $\epsilon_\theta(x_t, t)$ is the predicted noise for the current time step t and data point x_t.

2. **Estimate the original data point**:
$$\tilde{x}_0 \leftarrow \frac{x_t - \sigma_t \epsilon}{\sqrt{\bar{\alpha}_t}}$$
This step computes an estimate $\tilde{x}_0$ of the original data point based on the predicted noise.

3. **Sample the previous data point**:
$$x_{t-1} \sim N(\tilde{\mu}_t(x_t, \tilde{x}_0), \tilde{\sigma}_t^2 I)$$
The previous data point x_{t-1} is sampled from a Gaussian distribution with mean $\tilde{\mu}_t(x_t, \tilde{x}_0)$ and variance $\tilde{\sigma}_t^2$.

4. **Update the time step**:
$$t \leftarrow t - 1$$
Decrease the time step and repeat the process until $t = 0$.

This loop continues until the data point x_0 is recovered, completing the backward diffusion process and generating a new sample from the original data distribution.

4.2.3 Generation

Denoising Diffusion Implicit Model (DDIM)

The original Denoising Diffusion Probabilistic Model (DDPM)[11] generates high-quality images but is relatively slow because it typically requires around 1000 steps in the forward diffusion process for the distribution of x_T to closely resemble Gaussian noise. This results in the backward diffusion process also requiring 1000 steps, as each step in the backward process depends on the previous one. For example, sampling $x_{t-2}|x_t$ would require marginalizing over x_{t-1}, which is generally computationally intractable.
To address this issue, **DDIM** was introduced as a method to accelerate the generation process by skipping steps. It takes a model trained with **DDPM loss** and allows for skipping diffusion steps, trading off an adjustable amount of quality. Unlike DDPM, which is a **Markovian process**, DDIM operates in a **non-Markovian setting**, making the reverse diffusion process deterministic when the variance is set to zero. As a result, DDIM can generate images with fewer steps than DDPM, and often outperforms DDPM in scenarios with limited sampling steps.

Key Concepts

1. **Forward Diffusion Process**: The forward process remains the same as DDPM:
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sigma_t \epsilon$$
where x_0 is the original image and ϵ is Gaussian noise.

2. **Backward Diffusion Process**: During the backward process, given x_t and the predicted noise $\epsilon_\theta(x_t, t)$, the model first estimates the original image x_0' as:
$$x_0' = \frac{x_t - \sigma_t \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}$$
From this point, instead of performing the usual backward process step-by-step, DDIM allows jumping to any step s, where $0 \leq s < t$, by computing the next denoised sample:
$$x_s = \sqrt{\bar{\alpha}_s} x_0' + \sqrt{\sigma_s^2 - (\sigma_s')^2} \epsilon_\theta(x_t, t) + \sigma_s' \epsilon$$
Here, σ_s' is a flexible parameter within the range $[0, \sigma_s]$, and $\epsilon \sim N(0, I)$ is newly sampled Gaussian noise.

3. **Deterministic Process**: When all $\sigma_s' = 0$, the backward process becomes fully deterministic, allowing DDIM to generate images with fewer steps. In

some cases, it can generate comparable results to DDPM with as few as 20 steps instead of 1000.

4. **Eta Parameter (η)**: The DDIM paper introduces a single "eta value" $\eta \in [0, 1]$, where $\sigma'_s = \eta\tilde{\sigma}_s$. - $\eta = 1$ corresponds to the original DDPM. - $\eta = 0$ results in fully deterministic DDIM. - Intermediate values of η interpolate between the two methods.

5. **Eta Parameter (η)**: The DDIM paper introduces a single "eta value" $\eta \in [0, 1]$, where $\sigma'_s = \eta\tilde{\sigma}_s$. - $\eta = 1$ corresponds to the original DDPM. - $\eta = 0$ results in fully deterministic DDIM. - Intermediate values of η interpolate between the two methods.

Applicability

The DDIM algorithm can also be applied to score-based diffusion models, offering a flexible and efficient alternative for generating high-quality images.

Training and Inference of LDM

The Latent Diffusion Model (LDM) is trained using a process that gradually adds noise to training images via a Markov chain, and the model learns to reverse this process. Here's a clearer breakdown of the training and inference stages:

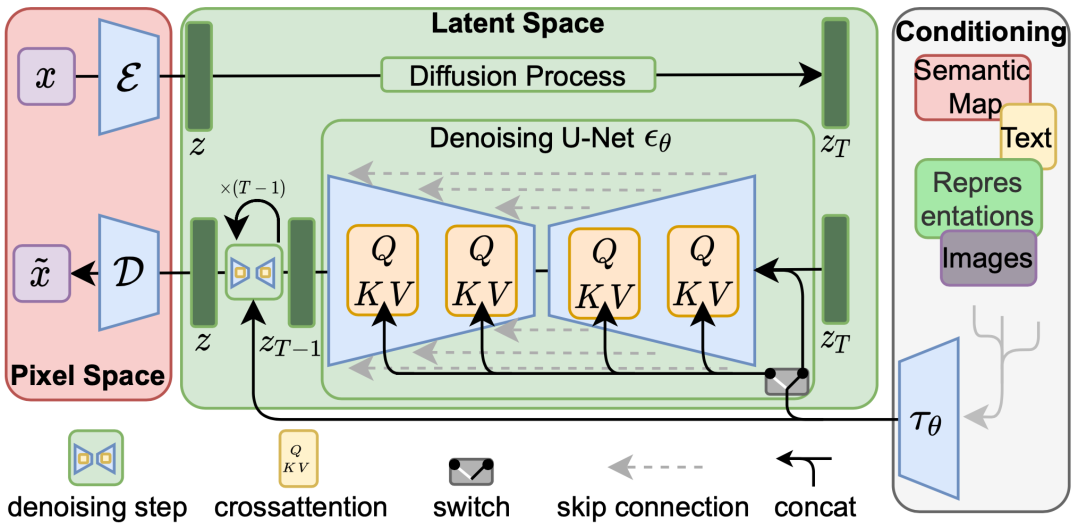
Training

1. Forward Diffusion Process: Starting with a real image x_0, a sequence of progressively noisier latent variables $x_1, x_2, \ldots, x_T$ is generated. Gaussian noise is added at each step following a predefined "noise schedule."

2. **Reverse Diffusion Process**: Beginning with a highly noisy image x_T, the model is trained to predict and remove the noise added at each previous step, eventually recovering the original image x_0.

3. **Loss Function**: The training objective is to minimize the difference between the model's predicted noise and the actual noise applied at each step. This is typically done using a mean squared error (MSE) loss function, ensuring the model effectively learns the noise distribution.

4.3 Latent Diffusion Models



The **Latent Diffusion Model (LDM)**, developed by the **CompVis** group at **LMU Munich**, is a diffusion model architecture designed to improve upon traditional diffusion models.
Diffusion models, first introduced in 2015, are trained to remove Gaussian noise that is gradually applied to images. LDM enhances this approach by conducting the diffusion process in a compressed **latent space**, which is more efficient. Additionally, it incorporates **self-attention** and **cross-attention conditioning** for more effective learning.
LDM has become the foundation for many diffusion models in use today, including the popular **Stable Diffusion** versions from 1.1 to 2.1.
Diffusion models were first introduced in 2015 as a way to sample from complex probability distributions using techniques from non-equilibrium thermodynamics, with an initial implementation in Theano. In 2019, the **Noise Conditional Score Network (NCSN)** improved this approach using score-matching with **Langevin dynamics**. This was followed by the **Denoising Diffusion Probabilistic Model (DDPM)** in 2020, which further enhanced the method with variational inference. In December 2021, the **Latent Diffusion Model (LDM)** was introduced, leading to the release of **Stable Diffusion** versions starting from 1.1 in August 2022. These versions, ranging from SD 1.1 to 1.5, refined the LDM architecture by training on increasingly aesthetic datasets, with version 1.5 being released by RunwayML in October 2022.

4.3.1 Architecture

The Latent Diffusion Model (LDM) can generate various types of data, but for clarity, let's focus on its application in conditional text-to-image generation.
LDM consists of three main components: a **variational autoencoder (VAE)**, a **modified U-Net**, and a **text encoder**.

1. **VAE Encoder**: It compresses the input image from pixel space into a lower-dimensional latent space, capturing the image's core semantic information. During the forward diffusion process, Gaussian noise is progressively added to this latent representation.

2. **Modified U-Net**: Built with a **ResNet backbone**, the **U-Net** is responsible for denoising the noisy latent representation during the reverse diffusion process. This step refines the latent representation, progressively removing noise to recover the core image features.

3. **VAE Decoder**: Once the denoised latent representation is obtained, the VAE decoder reconstructs the final image by converting it back to pixel space.

In text-to-image generation, the denoising process is conditioned on text prompts or other types of input data, such as images. The conditioning data is encoded and introduced to the U-Net through a cross-attention mechanism. For text-based conditioning, a pretrained **CLIP ViT-L/14 text encoder** is used to transform text prompts into a corresponding embedding space.

4.3.2 Variational Autoencoder

In Latent Diffusion Models, a **Variational Autoencoder** is used to compress image data into a more manageable form. The VAE is trained on a large dataset of images to learn how to map high-dimensional image data into a lower-dimensional latent space while preserving the essential features.

1. **Encoder**: The VAE's encoder takes an input image and transforms it into a **latent representation**, a compressed version of the image. This latent representation captures the core semantic content of the image in fewer dimensions, making it easier for subsequent processing.

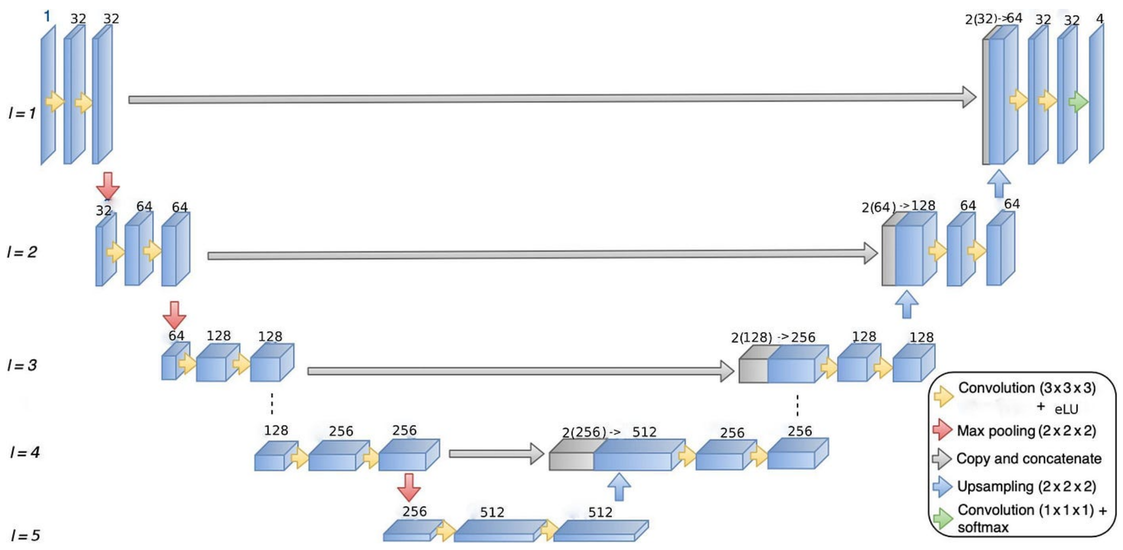
2. **Latent Representation**: This compressed latent space is then passed to the **U-Net** model, which performs the task of denoising and refining the latent data during the diffusion process.

3. **Decoder**: After the diffusion process is completed, the **VAE's decoder** is used to reconstruct the latent representation back into an image. The decoder

learns to reverse the encoding process, converting the abstracted latent data back into pixel space, producing a high-quality image.

The VAE thus acts as a bridge, encoding images into a latent space where they can be efficiently manipulated and decoded back into their original form or a modified version. By using this latent representation, LDMs can reduce computational complexity while maintaining the quality of the generated images.

4.3.3 U-Net Architecture in Latent Diffusion Models



The U-Net backbone in Latent Diffusion Models (LDM) processes three key inputs:

1. **Latent Image Array**: This is the output of the VAE encoder. It has dimensions like (channels, width, height), where the number of channels is typically much smaller than a standard RGB image. For example, a latent image with dimensions (3, 64, 64) could be viewed as a 64x64 image with 3 channels, though it is not meant for direct visualization.

2. **Timestep Embedding**: This vector encodes how much noise is present in the latent image. At the start (t=0), the image is almost noiseless, while at a high value like t=100, the image has significant noise. This helps the U-Net understand how much denoising is needed at each step.

3. **Modality Embedding**: This input provides additional conditioning information. For instance, in text-to-image generation, the text is tokenized and processed by a text encoder (like CLIP), converting it into a sequence of

embedding vectors that guide the denoising process. Other types of input, like images, can also be used to condition the U-Net for specific tasks (e.g., generating images in the same style as an input).

Denoising Process

The U-Net backbone performs iterative denoising. In each pass, it outputs a **predicted noise vector**, which is then scaled and subtracted from the latent image, resulting in a slightly cleaner latent representation. This process is repeated according to a predefined **denoising schedule**. Once the latent image has been denoised sufficiently, the **VAE decoder** converts it back into a final image.

U-Net Structure

Similar to a traditional U-Net, this backbone consists of **downscaling** (reducing the resolution) and **upscaling** (restoring the resolution) layers. However, it includes special modules for handling the additional embedding inputs. Here's an overview of a single downscaling layer:

1. **ResBlock with Time Embedding**: The latent array is processed through a ResNet-style block. The timestep embedding is passed through a simple feedforward network and added to the latent array after it goes through convolutional layers. This is repeated to refine the latent representation further.

2. **Spatial Transformer**: This module applies attention mechanisms, transforming the latent image based on the modality embedding (e.g., text or another condition). It operates like a standard Transformer decoder without causal masking. In **cross-attention blocks**, the latent array serves as the query, while the modality embedding serves as both the key and value. If no embedding is provided, the model defaults to **self-attention**.

Pseudocode

Listing 4.1: Python code for U-Net components

```
def ResBlock(x, time, residual_channels):
    x_in = x
    time_embedding = feedforward_network(time)
    x = concatenate(x, residual_channels)
    x = conv_layer_1(activate(normalize_1(x))) + time_embedding
    x = conv_layer_2(dropout(activate(normalize_2(x))))
    return x_in + x

def SpatialTransformer(x, cond):
    x_in = x
    x = normalize(x)
    x = proj_in(x)
    x = cross_attention(x, cond)
    x = proj_out(x)
    return x_in + x

def unet(x, time, cond):
    residual_channels = []
    for resblock, spatialtransformer in downscaling_layers:
        x = resblock(x, time)
        residual_channels.append(x)
        x = spatialtransformer(x, cond)

    x = middle_layer.resblock_1(x, time)
    x = middle_layer.spatialtransformer(x, time)
    x = middle_layer.resblock_2(x, time)

    for resblock, spatialtransformer in upscaling_layers:
        residual = residual_channels.pop()
        x = resblock(concatenate(x, residual), time)
        x = spatialtransformer(x, cond)

    return x
```

Inference

- After training, the model can generate new images by running the reverse diffusion process. Starting from a random noise sample, the model progressively removes noise in line with the learned distribution, resulting in a final image.

This method allows the LDM to generate high-quality images from noise by iteratively refining the noise sample using the learned noise-prediction model.

4.3.4 Motivation behind Latent Diffusion: computational efficiency and quality

Latent Diffusion Models (LDMs) were developed to address the challenges of computational efficiency and image quality in generative models. Traditional diffusion models, operating in pixel space, require significant computational resources due to the high dimensionality of the data. LDMs overcome this by performing diffusion in a lower-dimensional latent space, drastically reducing the computational load without sacrificing image quality. This approach allows for faster training and inference, while still maintaining the ability to generate high-resolution, detailed images. Additionally, by conditioning the diffusion process on modalities like text or images, LDMs enable more control over the generated content, making them highly versatile for tasks such as text-to-image synthesis.

Chapter 5

Experiments and Geo-localization Task

5.1 What is Geo-localization? Problem Definition and Real-World Importance

Geo-localization is the process of determining the precise geographical location where an image or video was captured, solely by analyzing the visual content of the image. Unlike traditional methods that rely on GPS data or other geospatial sensors, geo-localization through visual cues focuses on comparing the query image with a large database of geo-tagged reference images to infer its location. This task has become increasingly important due to the rising demand for location-aware applications in various industries, such as autonomous systems, smart cities, and augmented reality.

Problem Definition

In visual geo-localization, the key challenge is to identify the most similar geo-tagged images from a large database and deduce the query image's location based on these comparisons. The problem can be formalized as an image retrieval task, where the goal is to find reference images from a known geographic region that match the visual features of the query image. The retrieved images must provide accurate information about the location, even in the absence of any GPS metadata. The task is complicated by several factors such as changes in appearance due to time of day, weather conditions, seasonal variations, structural changes, and occlusions (e.g., parked vehicles or pedestrians).

Real-World Importance

Geo-localization has numerous practical applications that impact both industrial and societal domains. Some of the most critical applications include:

1. **Autonomous Driving**: In the field of autonomous vehicles, geo-localization is essential for precise navigation. Autonomous cars need to continuously determine their location relative to a map, especially in environments where GPS signals are unreliable or unavailable, such as urban canyons or tunnels. Visual geo-localization can serve as an alternative or complement to GPS, providing location estimates by analyzing surrounding visual landmarks.

2. **Augmented Reality (AR)**: Many AR applications require real-time localization to overlay digital content in the correct geographic context. For example, in AR navigation systems, geo-localization enables accurate placement of directions or points of interest (POI) in the user's real-world view. AR-based tourism apps also rely on visual geo-localization to provide information about historical landmarks or nearby attractions based on the user's location.

3. **Robotics and Drone Navigation**: Mobile robots and drones, especially those operating in indoor or urban environments, often face challenges with GPS signal loss. Geo-localization using visual information can provide reliable positioning for these systems, enabling them to navigate efficiently in environments like warehouses, disaster zones, or densely populated cities.

4. **Social Media and Digital Forensics**: In the era of social media, users frequently share images without providing geographic context. Geo-localization helps identify where an image was captured, aiding in content verification and digital forensics. This is particularly useful for investigative journalism, environmental monitoring, and law enforcement.

5. **Visual Assistive Technologies**: For visually impaired individuals, geo-localization-based assistive technologies can provide spatial awareness and help them navigate unfamiliar environments by describing their surroundings and offering turn-by-turn navigation based on their current location.

6. **Smart Cities**: In the context of smart city initiatives, visual geo-localization enables real-time tracking of infrastructure changes, monitoring of urban development, and disaster management. City planners can use this technology to gather data on urban expansion and traffic management by analyzing images from various sources, including cameras placed around the city.

Overall, geo-localization has emerged as a critical technology for solving real-world problems, particularly in cases where traditional GPS methods are either unavailable or unreliable. Its potential to enhance localization accuracy in both indoor and outdoor environments makes it indispensable for modern navigation, safety, and information systems.

5.1.1 Introduction to the Cosplace Datasets and Task Specifications

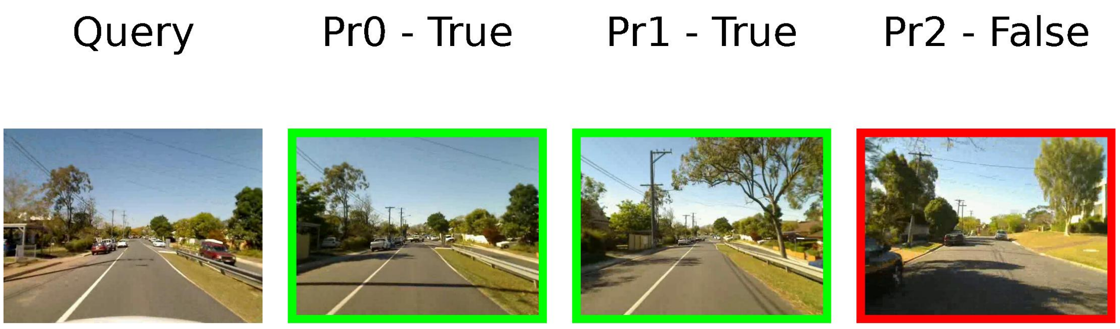
The **Cosplace datasets**[12] have been specifically curated to address the unique challenges of large-scale visual geo-localization. They were designed by experts in the field to provide a robust and realistic evaluation platform for geo-localization algorithms, particularly in urban settings. The datasets include millions of images collected from various locations across entire cities, capturing diverse environmental conditions, landmarks, and scenes that are typically encountered in city-wide geo-localization tasks.

Datasets Overview

The Cosplace datasets are designed to simulate real-world conditions, presenting a variety of challenges that make geo-localization a complex task. The images in the datasets are collected from multiple cities, and each image is geo-tagged with precise latitude and longitude coordinates. This allows for both training and evaluation of geo-localization models across a wide geographic area. The datasets include various environmental and temporal variations, such as:

- **Time of Day**: Images taken during different times of the day, from bright daylight to nighttime scenes.
- **Weather Conditions**: The datasets capture images under varying weather conditions such as sunny, rainy, and foggy environments.
- **Seasonal Changes**: Seasonal shifts like summer versus winter are reflected in the dataset, affecting the appearance of vegetation, lighting, and other visual cues.
- **Structural Changes**: The datasets also account for permanent or temporary changes in the environment, such as new buildings, roadworks, or parked vehicles, which can obscure critical visual features.
- **Occlusions**: Common obstacles such as pedestrians, cyclists, and cars that partially block landmarks or important visual information are included, forcing models to generalize well in challenging conditions.

5.1.2 Task Specifications



The primary task of the Cosplace datasets is formulated as an **image retrieval problem**. Given a query image, the goal is to retrieve the most visually similar geo-tagged images from a large reference database and use them to estimate the location of the query image. This task can be broken down into several key components:

1. **Training**: Models are trained on a subset of geo-tagged images from the Cosplace datasets, with the goal of learning to extract robust visual features that are invariant to environmental changes. The training process involves using image descriptors that capture key visual information such as shapes, textures, and spatial relationships between objects. These descriptors are then compared with those from other images in the dataset.

2. **Evaluation**: During evaluation, a query image (without any GPS information) is provided to the model, which must then retrieve the most similar images from the reference dataset. The performance of the model is typically measured by how accurately it can estimate the geographic location of the query image, often using metrics such as mean reciprocal rank (MRR), precision at k, or recall at specific distance thresholds.

3. **Environmental Variation Handling**: One of the key challenges for models using the Cosplace datasets is the ability to handle the wide range of environmental changes captured in the images. To succeed in this task, models must be able to generalize across different times of day, seasons, and weather conditions. The presence of occlusions and structural changes in the dataset also necessitates robust feature extraction techniques that can still identify critical landmarks despite visual noise.

4. **Data Scale**: The Cosplace datasets are large-scale, consisting of millions of images spread across a broad geographic region. This presents additional challenges in terms of computational efficiency, as the model must process and retrieve images from a vast database in real time. This requires both optimized image retrieval techniques and efficient memory management to ensure scalability.

5. **Data Augmentation and Preprocessing**: To improve model performance, data augmentation techniques such as random cropping, rotations, and color jittering are often applied during training. These augmentations help the model learn to handle the variability in real-world images. Additionally, preprocessing steps like feature normalization and dimensionality reduction are used to streamline the retrieval process, making it computationally feasible to handle large-scale data.

Importance of the Cosplace Dataset

The Cosplace datasets are significant for several reasons.

- **First**, its large scale and diverse set of images make it a highly realistic benchmark for evaluating the robustness and scalability of geo-localization models. It forces models to deal with real-world challenges such as environmental variability and large-scale image retrieval, which are crucial for practical applications.

- **Second**, the Cosplace datasets are designed for urban environments, which are some of the most challenging settings for geo-localization due to the high density of visually similar features (e.g., buildings, roads) and the presence of dynamic objects (e.g., vehicles, pedestrians). The dataset encourages the development of models that can reliably geo-locate images in these complex settings, making it highly relevant for applications in autonomous driving, AR navigation, and urban robotics.

In conclusion, the Cosplace datasets and their associated task specifications provide a comprehensive platform for developing and testing state-of-the-art visual geo-localization models. Its focus on real-world variability and scalability ensures that models trained on this dataset will be well-equipped to handle the challenges of geo-localization in practical scenarios.

5.2 Latent Diffusion for Geo-localization

5.2.1 Why Latent Diffusion? Applicability to geo-localization tasks

Latent diffusion models (LDM)[13] were chosen for this study due to their computational efficiency and their ability to offer a more lightweight alternative to traditional diffusion models. Standard diffusion processes are powerful generative models but tend to be computationally expensive, especially when applied to large-scale tasks such as image generation or geo-localization. The latent diffusion model alleviates this by operating within a compressed latent space, thus significantly reducing the computational load while retaining high-quality generation capabilities.

5.2.2 Applicability to Geo-localization Tasks

The decision to use latent diffusion in the context of geo-localization, particularly with images of urban landscapes and panoramas sourced from Google Maps, stems from the need to explore whether the diffusion process can effectively handle such datasets. Geo-localization tasks involve retrieving and matching specific geographical features from large databases, and understanding how well diffusion models can generate or recreate such features is critical to advancing the field.
By testing the unconditional generation of images with latent diffusion, we aimed to investigate whether the diffusion process can preserve key features present in urban and panoramic images. The ability of the model to generate realistic, feature-rich images plays an important role in assessing its potential applicability to geo-localization tasks. The experiment helps to determine if critical geographical elements, such as building structures, road layouts, and natural landmarks, are maintained in the generated images—features that are essential for accurate geo-localization.
In particular, the **following questions** are explored:

- **Can latent diffusion generate accurate images based on urban environments?** If the diffusion model retains distinct features like buildings, streets, and other landmarks in its generated outputs, it indicates that the model could be useful for tasks involving geo-localization.

- **Does the model maintain meaningful details in panoramic images?** Panoramic images often cover wide, varied spaces, and the ability of the latent diffusion model to recreate these expansive views with important detail is crucial for its success in geo-localization tasks.

5.2.3 Explainability and Model Understanding in Geo-localization

Another key motivation for adopting latent diffusion is its potential to enhance explainability in the Cosplace model. Explainability in machine learning is an essential tool for understanding what features a model prioritizes during training and inference, and why it occasionally makes errors. In the case of visual geo-localization, understanding why a model may misclassify a location or fail to retrieve the correct image from a large database can offer valuable insights into model limitations.
Latent diffusion offers an opportunity to probe the Cosplace model and uncover the specific features it learns during training.
By observing the latent space, which serves as a compressed and meaningful representation of image features, we can identify:

- **What features are most important in the geo-localization task?** This includes visual landmarks like unique building shapes, street signs, and road structures that the model frequently relies on for matching.

- **Why does the model make mistakes?** By analyzing the images generated by latent diffusion and examining cases where the model fails, we can understand what misleading or absent features may be causing the model to produce incorrect results.

Errors in geo-localization are often attributable to several factors such as:

- **Visual similarity**: Areas in different parts of a city, or even different cities, might appear visually similar, making it difficult for the model to distinguish between them.

- **Occlusions or changes in environment**: Temporary or permanent changes, such as construction, parked cars, or weather conditions, can obscure critical features, leading to mistakes.

- **Lack of distinctive landmarks**: Some locations may not have prominent features that are easy for the model to distinguish, causing ambiguity.

By integrating latent diffusion with Cosplace, we not only enhance the computational efficiency of the diffusion process but also gain a valuable tool for understanding and refining the geo-localization model. This deeper insight into the learned features helps inform future improvements, including how to better handle cases of visual similarity and feature ambiguity, ultimately driving better performance and fewer localization errors.

5.2.4 Explanation of the repository used (CompVis/Latent-Diffusion)

The **CompVis/Latent-Diffusion** repository is the source from which we obtained the models used for our geo-localization task. This repository is based on the **Latent Diffusion Models (LDMs)** introduced in the paper **High-Resolution Image Synthesis with Latent Diffusion Models** by *Robin Rombach* et al. The purpose of utilizing these models for our geo-localization task is to take advantage of LDM's computational efficiency and its ability to maintain high-quality image generation. By operating in latent space, LDMs reduce the computational cost of traditional diffusion models while still capturing important features of images, making them a good fit for our work on city panoramas and street views.

Purpose of the Paper

The core motivation of the paper is to address the issue of high computational demands in traditional diffusion models, particularly when operating in pixel space. LDMs introduce a more efficient approach by conducting the diffusion process in a lower-dimensional latent space, which allows for faster image generation while preserving quality. The paper highlights the benefits of this approach, particularly in cases where high-resolution images are required. Key objectives of the paper are:

- **Improving Computational Efficiency**: LDMs aim to reduce the number of operations required during the diffusion process by transforming images into a compact latent space. This reduces the complexity of the forward and backward diffusion steps, making the model faster to train and use.
- **Maintaining Image Quality**: Despite operating in a reduced latent space, LDMs are designed to preserve the semantic and perceptual quality of the generated images. The approach ensures that even with fewer diffusion steps, the output images retain high fidelity.
- **Conditional Generation Flexibility**: The paper also emphasizes the versatility of LDMs in conditional generation tasks. The model can incorporate different types of conditioning signals, such as text or image inputs, through cross-attention mechanisms. This makes it adaptable to a wide range of generative tasks.

Model Architecture

The architecture of LDMs comprises three major components:

1. **Variational Autoencoder (VAE)**: The VAE compresses high-dimensional image data into a lower-dimensional latent space, where the diffusion process

takes place. The encoder part of the VAE transforms the input images into latent representations, while the decoder reconstructs images from these latent representations.

2. **U-Net Backbone**: During the reverse diffusion process, a U-Net model is used to denoise the noisy latent representations step-by-step. Equipped with a ResNet backbone, the U-Net refines the latent representation progressively, eventually producing a clean latent image.
3. **Cross-Attention Mechanism**: To handle conditional generation tasks such as text-to-image generation, the model uses cross-attention layers. For example, text inputs are processed by a CLIP encoder to generate embedding vectors, which are then used to condition the diffusion process during denoising. This allows the model to incorporate contextual information effectively.

Performance and Efficiency

The **CompVis/Latent-Diffusion** repository demonstrates several performance improvements:

- **Faster Training and Inference**: Since LDMs operate in a lower-dimensional latent space, they require fewer computational resources. This makes both training and inference faster and more feasible, even on large datasets or high-resolution images.
- **High-Quality Image Generation**: Despite the reduced computational cost, LDMs are able to maintain high image quality. They are particularly well-suited for generating high-resolution images and handling complex tasks like super-resolution and inpainting.
- **Adaptability to Various Tasks**: The cross-attention mechanism allows LDMs to condition the generation process on a variety of inputs, such as text or images. This makes the model versatile and applicable to different generative tasks beyond simple image synthesis, including those relevant to geo-localization.

The **CompVis/Latent-Diffusion**[14] repository provided us with the foundational models for our geo-localization experiments. Its architecture, designed for computational efficiency and high-quality image generation, was highly suitable for our task of understanding whether latent diffusion models could be applied effectively to panoramic and city images for geo-localization purposes. The repository includes pre-trained models and scripts that facilitated our work, enabling efficient image generation and serving as a tool for exploring how diffusion models can be adapted for explainability and feature learning in geo-localization tasks.

Chapter 6

Results and Analysis

6.1 Unconditional Generation

6.1.1 Explanation of unconditional generation

In the first set of experiments, we applied the method of unconditional generation of images using diffusion models. Unconditional generation refers to the process of generating images without providing any additional conditional inputs such as text prompts, image references, or other external data. The objective is to see how well the model, trained purely on a dataset of images, can independently generate samples that reflect the characteristics of the training data. For this purpose, we used two well-known geo-localization datasets: **San Francisco Extra Large** and **Pittsburgh 30k**.
Both datasets contain a wide variety of images capturing urban landscapes, streets, buildings, vehicles, and other city-related features. Our goal in starting with unconditional generation was threefold:

1. **Assess the Model's Ability to Train on City and Street Images**:
   The first motivation was to examine whether the latent diffusion model could be trained effectively using images from cities and streetscapes. Urban landscapes often consist of intricate details like buildings, vehicles, trees, and pedestrians, which makes it challenging for generative models to learn and replicate. Starting with unconditional generation allowed us to evaluate the model's ability to learn these complex scenes and determine whether it could handle the diversity of urban elements.

2. **Evaluate the Preservation of Characteristics and Details in Generated Images**:
   A key reason for this experiment was to understand whether the model could generate images that maintain the distinct visual characteristics and details

of the datasets. We wanted to see if the model could successfully recreate not only the large-scale structural elements like buildings and streets but also finer details such as cars, trees, and pedestrians. Maintaining these details is crucial for generating realistic cityscapes and determining the model's ability to represent the complexity of real-world urban environments.

3. **Assess the Model's Potential for Conditional Generation and Geo-localization Task Explainability**:
   Lastly, this experiment also served as a foundation to explore the model's potential for **conditional generation**, particularly in the context of the geo-localization task using the **Cosplace** dataset. If the model can generate realistic, high-quality images in an unconditional setting, it stands to reason that it could also perform well in a conditional generation setting. Conditional generation would allow the model to be guided by specific inputs such as location-based data or specific features (e.g., streets, landmarks), which could enhance the accuracy and explainability of the geo-localization task. By starting with unconditional generation, we aimed to gather insights into how the model interprets and synthesizes various elements of cityscapes. This would, in turn, help us understand what features the model learns and prioritizes, which could inform us about the strengths and weaknesses of the model when applied to the geo-localization task.

6.1.2 Unconditional Generation Process

For both the San Francisco Extra Large and Pittsburgh 30k datasets, we conducted unconditional generation by training the latent diffusion model on the entire image corpus. The images were processed into latent space using a **Variational Autoencoder**, and Gaussian noise was applied during the forward diffusion process. The model was then tasked with denoising the noisy latent representation through the reverse diffusion process to generate new images.

6.1.3 Observations from Unconditional Generation

- **Model's Ability to Capture Urban Features**:
  In both datasets, the latent diffusion model successfully captured and generated large-scale urban features such as buildings, roads, and the overall structure of city environments. This shows that the model is capable of learning complex street and city layouts from the training data.

- **Details such as Cars, Trees, and People**:
  A significant outcome was the model's ability to generate finer details in some cases, such as vehicles, trees, and occasionally pedestrians. However, the

clarity and fidelity of these smaller elements varied depending on the amount of noise removed during the diffusion process. In some cases, the generated cars and trees lacked sharpness or appeared blended into the surroundings, indicating room for improvement in capturing high-frequency details.

- **Image Diversity**: The generated images showed reasonable diversity in terms of different types of buildings, street views, and urban layouts. This suggests that the model does not merely memorize the training data but learns a broader representation of urban scenes, allowing it to create varied outputs.

Potential for Conditional Generation

The success of the model in unconditional generation opens up promising avenues for conditional generation in geo-localization tasks. By conditioning the generation process on specific input features, such as image embeddings or location data, the model could be guided to focus on relevant aspects of the scene for more precise geo-localization. This could enhance the **Cosplace** model by helping it understand which visual elements—such as specific building styles, road structures, or natural features—are most important for accurately identifying locations.

Role in Explainability

Another potential application is using latent diffusion to improve m**odel explainability** in the geo-localization task. By analyzing how the model generates images and what features it emphasizes, we can gain insights into what visual cues are most significant for the model when making predictions. Furthermore, understanding why the model sometimes makes mistakes (e.g., confusing two similar-looking locations) can also be informed by studying the generated images. This could highlight which elements the model struggles with and help refine both the training data and model architecture for better geo-localization performance.

6.2 Experimental setup: hyperparameters, training, and evaluation metrics

In this section, we outline the experimental setup for our unconditional generation model, which closely follows the settings used in the **LSUN Churches** model provided in the **CompVis/latent-diffusion** repository. The LSUN Churches dataset was chosen as a baseline due to its similarity in terms of image complexity and features (such as buildings and landscapes) to the city and street images in our geo-localization task. The performances achieved with the LSUN Churches dataset were among the best reported in the repository, making it a solid foundation for our experiments.

Image Sizes and Model Configuration

For this experiment, we conducted two sets of tests with different image resolutions:

- **128x128** and **256x256** pixels.
  While the 256x256 setting is the default configuration in the original repository, the 128x128 option was introduced to reduce computational demands. Lowering the image resolution to 128x128 allows for faster training and lighter GPU memory usage, albeit with some potential loss in image detail and quality.

Model Channels

We also experimented with two different configurations for the number of model channels:

- **192 model channels**: This is the default configuration in the original repository, known for producing high-quality outputs.
- **160 model channels**: This reduced channel setting was used to lighten the overall model, making it more efficient in terms of GPU memory usage. This reduction in channels allows for training on lower-end hardware without consuming all available GPU memory, at the potential cost of a slight performance tradeoff.

Diffusion Timesteps

In all training runs, we used **1000 timesteps** for the Denoising Diffusion Probabilistic Model (DDPM). Previous experiments have shown that increasing the number of timesteps beyond 500 does not significantly impact model performance. Therefore, 1000 timesteps were chosen as a balanced approach to ensure model stability without unnecessarily increasing training time.

Learning Rate and Loss Function

The base learning rate was set to **5.0e-5**, consistent with the original settings in the repository. This learning rate was empirically determined to provide a stable convergence for models of this size.
For the loss function, we utilized **L1 loss**, which measures the mean absolute error between the generated images and the original images. L1 loss was chosen for its ability to produce sharper results in generative models compared to L2 loss.

U-Net Input Sizes

The input image sizes for the U-Net, or the output size from the VAE encoder, were configured as follows:

- **16x16 pixels** for input images of size 128x128.
- **32x32 pixels** for input images of size 256x256.

These sizes are based on the configuration in the repository, ensuring that the U-Net receives appropriately downscaled representations from the encoder for efficient diffusion processing.

Autoencoder Configuration

For the autoencoder component in the **first_stage_config**, we used the following setup:

```
target: ldm.models.autoencoder.AutoencoderKL
params:
  kl_f8
```

This configuration leverages **KL divergence** as part of the autoencoder architecture to compress and reconstruct images, helping the model learn compact latent representations efficiently.

Image Logging and Callbacks

Finally, during training, image logging was set up to monitor the generation process and model progress. We used the **Lightning** module to configure the logging parameters:

```
lightning:
  callbacks:
    image_logger:
      target: main.ImageLogger
      params:
        batch_frequency: 5000
        max_images: 8
        increase_log_steps: False
```

These settings ensure that the model generates sample images every 5000 batches, logging up to 8 images at a time for evaluation purposes. This helps monitor the model's progress and the quality of generated images throughout training without overwhelming the logging system.

6.2.1 Details Lsun-Churches Experiment

Dataset details:
Number of images: 126227
Image size: $256 \times H$ (with $H > 256$)

Model Details:
base_learning_rate: $5.0e - 5$
target: ldm.models.diffusion.ddpm.LatentDiffusion

Params:
linear_start: 0.0015
linear_end: 0.0155
num_timesteps_cond: 1
log_every_t: 200
timesteps: 1000
loss_type: l1
first_stage_key: "image"
cond_stage_key: "image"
image_size: 32
channels: 4
cond_stage_trainable: False
concat_mode: False
scale_by_std: True
monitor: 'val/loss_simple_ema'

Scheduler:
scheduler_config: target: ldm.lr_scheduler.LambdaLinearScheduler
params:
warm_up_steps: [10000]
cycle_lengths: [10000000000000]
f_start: [1.e-6]
f_max: [1.]
f_min: [1.]

Unet Model:
target: ldm.modules.diffusionmodules.openaimodel.UNetModel

```
params:
image_size: 32
in_channels: 4
out_channels: 4
model_channels: 192
attention_resolutions: [ 1, 2, 4, 8 ] num_res_blocks: 2
channel_mult: [ 1,2,2,4,4 ] num_heads: 8
use_scale_shift_norm: True
resblock_updown: True
```

First Stage Model:

```
target: ldm.models.autoencoder.AutoencoderKL
params:
embed_dim: 4
monitor: "val/rec_loss"
ckpt_path: "models/first_stage_models/kl-f8/model.ckpt"
ddconfig:
double_z: True
z_channels: 4
resolution: 256
in_channels: 3
out_ch: 3
ch: 128
ch_mult: [ 1,2,4,4 ]
num_res_blocks: 2
attn_resolutions: [ ]
dropout: 0.0
lossconfig:
target: torch.nn.Identity
cond_stage_config: "___is_unconditional___"
```

Data:

```
target: main.DataModuleFromConfig
params:
batch_size: 96
num_workers: 5
wrap: False
train:
target: ldm.data.lsun.LSUNChurchesTrain
params:
size: 256
```

validation:
target: ldm.data.lsun.LSUNChurchesValidation
params:
size: 256

Lightning:
callbacks:
image_logger:
target: main.ImageLogger
params:
batch_frequency: 5000
max_images: 8
increase_log_steps: False

Trainer:
benchmark: True

Samples

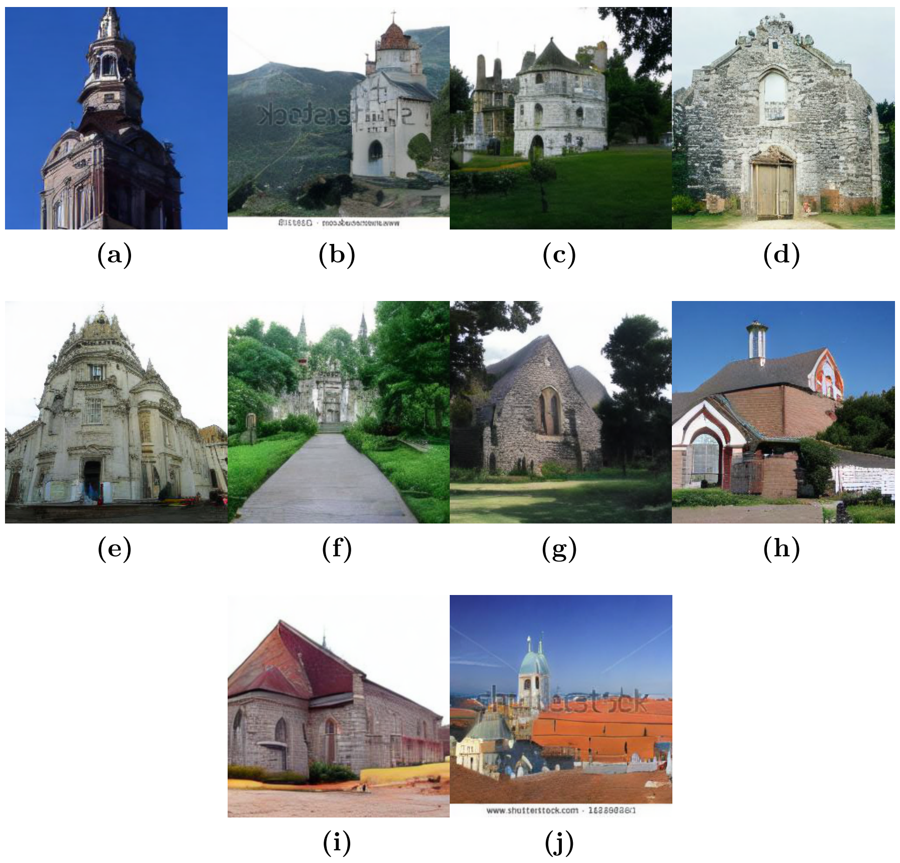


Figure 6.1: Samples lsun-churches original setup

6.2.2 San Francisco ExtraLarge

Dataset details
Number of Images: 41173104
Images size: 512×512

6.2.3 Original Setup

Model Details:
Size: 256×256
Batch Size: 48 Number of Images per Epoch: 41137
Trainer:

accumulate_grad_batches: 2

In this section, the experiment conducted used the original model configuration for the LSUN Churches dataset. The primary objective was to evaluate the impact of varying the batch size on the performance and efficiency of the model.
Specifically, we maintained the identical model architecture and hyperparameters, altering only the batch size from 96 to 48 to fit the memory of the GPU Titan RTX 24GB, and setting the accumulate gradient batches equal to 2.
The rationale behind this adjustment lies in the exploration of batch size effects on training stability, convergence speed, and overall model accuracy. Batch size is a critical parameter in deep learning that influences the gradient estimates, memory usage, and training dynamics.

6.2.4 Loss

Loss Calculation
In this latent diffusion model, the loss is calculated as the L1 loss between the Gaussian noise used as the target and the noise predicted by the model. The relevant function for this calculation is defined as follows:

```
def get_loss(self, pred, target, mean=True):
    if self.loss_type == 'l1':
        loss = (target - pred).abs()
    if mean:
        loss = loss.mean()
    return loss
```

In this function:
- *'pred'* represents the noise predicted by the model.
- *'target'* represents the Gaussian noise used as the target.

The L1 loss is computed by taking the absolute difference between the target noise and the predicted noise:

$$\text{loss} = |\text{target} - \text{pred}|$$

If the 'mean' parameter is set to 'True', the function calculates the mean of these absolute differences, resulting in a single scalar value representing the average loss:

$$\text{mean\_loss} = \frac{1}{N}\sum_{i=1}^{N} |\text{target}_i - \text{pred}_i|$$

This mean loss provides a concise measure of how well the model's predictions match the target Gaussian noise, facilitating model optimization during training.

Loss Adjustment with Variance

In this model, the loss is adjusted to account for varying levels of uncertainty by scaling the loss based on the variance. This adjustment ensures that the model remains adaptive to changes in the variance of the data. Here's how this adjustment works in detail:

```
loss_simple = self.get_loss(model_output, target, mean=False).mean
    ([1, 2, 3])
logvar_t = self.logvar[t].to(self.device)
loss = loss_simple / torch.exp(logvar_t) + logvar_t
```

Step-by-Step Explanation

1. **Calculate the Simple Loss**:

```
    loss_simple = self.get_loss(model_output, target, mean=False).mean
    ([1, 2, 3])

```

- `self.get_loss(model_output, target, mean=False)` computes the L1 loss between the predicted noise (`model_output`) and the target noise (`target`).
- `mean([1, 2, 3])` averages this loss over the dimensions `[1, 2, 3]`, which typically represent the height, width, and depth of the data tensor, resulting in a loss value for each batch item.

2. **Convert Log-Variance to Variance**:

```
    logvar_t = self.logvar[t].to(self.device)

```

- `self.logvar[t]` retrieves the log-variance for the current time step `t`.
- `.to(self.device)` ensures that the log-variance tensor is moved to the correct device (CPU or GPU) for computation.

In latent diffusion models, **log-variance** ($\log(\sigma^2)$) is used in the loss function to improve numerical stability and optimization.

- **Log-Variance Retrieval:** At each time step t, the log-variance $\log(\sigma_t^2)$ is retrieved using:

  $$\text{logvar}_t = \text{self.logvar}[t].\text{to(self.device)}$$

  where self.logvar[t] obtains the log-variance for the current time step t, and .to(self.device) ensures that the log-variance tensor is moved to the appropriate computation device (CPU or GPU).

- **Conversion to Variance:** To use log-variance in the loss calculation, it needs to be converted back to the variance (σ^2) using the exponential function:

  $$\sigma_t^2 = \exp(\text{logvar}_t)$$

  This conversion is necessary because the variance is used for scaling and computing the loss accurately.

3. **Adjust the Loss Using Variance**:

```
    loss = loss_simple / torch.exp(logvar_t) + logvar_t

```

- `torch.exp(logvar_t)` converts the log-variance to variance. Exponentiating the log-variance yields the actual variance since exp(log(variance)) = variance.

- `loss_simple / torch.exp(logvar_t)` scales the simple loss by the inverse of the variance. This scaling normalizes the loss according to the level of uncertainty represented by the variance. Higher variance (more uncertainty) results in a smaller contribution to the loss, while lower variance (less uncertainty) results in a larger contribution.

- `+ logvar_t` adds the log-variance to the scaled loss. This term ensures that the optimization process takes into account the uncertainty, encouraging the model to reduce the uncertainty when possible.

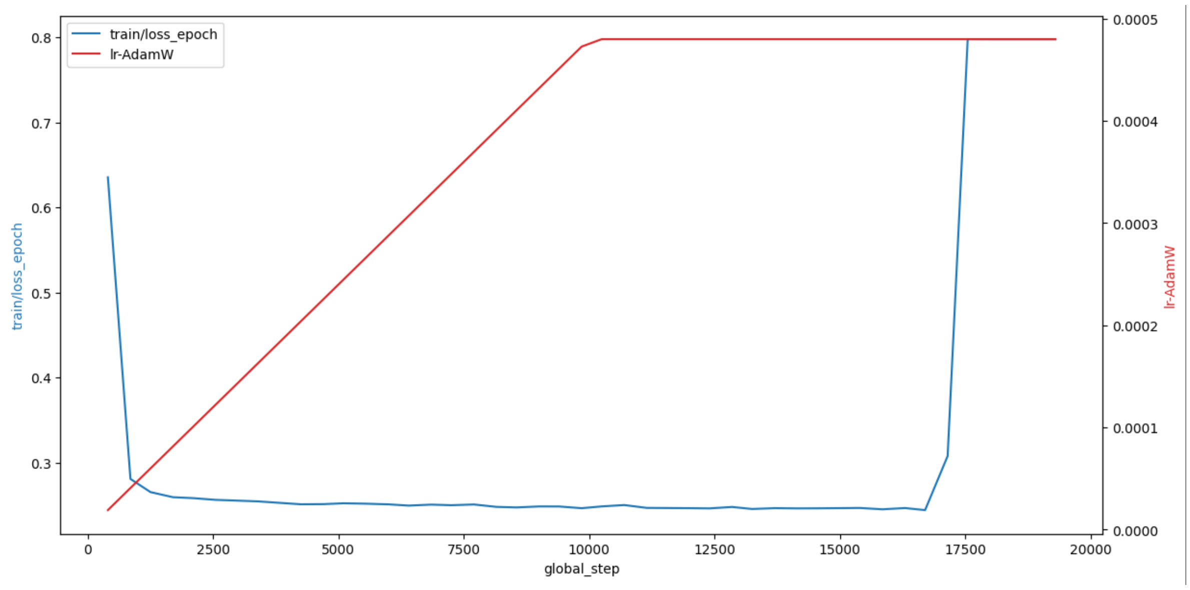
Purpose and Benefits

- **Adaptive Loss Scaling**: By scaling the loss according to the variance, the model becomes adaptive to changes in the variance of the data. This means that the model can appropriately weigh the loss contributions from predictions with different levels of certainty.

- **Handling Uncertainty**: Adding the log-variance term helps the model to handle varying levels of uncertainty in the data. When the uncertainty (variance) is high, the corresponding loss is reduced, preventing the model from being overly penalized for predictions in uncertain regions. Conversely, when the uncertainty is low, the loss is increased, pushing the model to improve accuracy in more certain regions.

- **Encouraging Certainty Reduction**: The term `+ logvar_t` in the loss function also encourages the model to minimize the uncertainty where possible. This helps the model not only to improve accuracy but also to become more confident in its predictions.

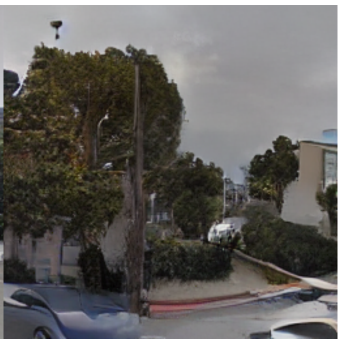
Overall, this adjustment makes the model more robust and effective in handling data with varying levels of uncertainty, leading to more reliable and accurate predictions.



Samples

(a)

(b)


(c)

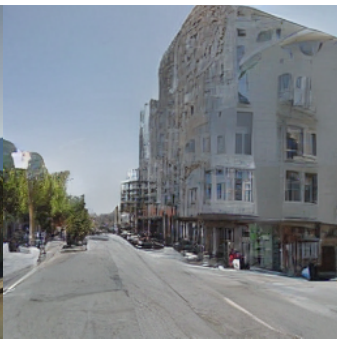
(d)

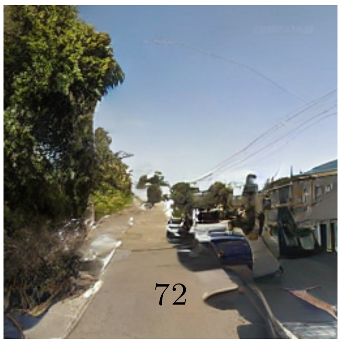
(e)

(f)

(g)

(h)


(i)

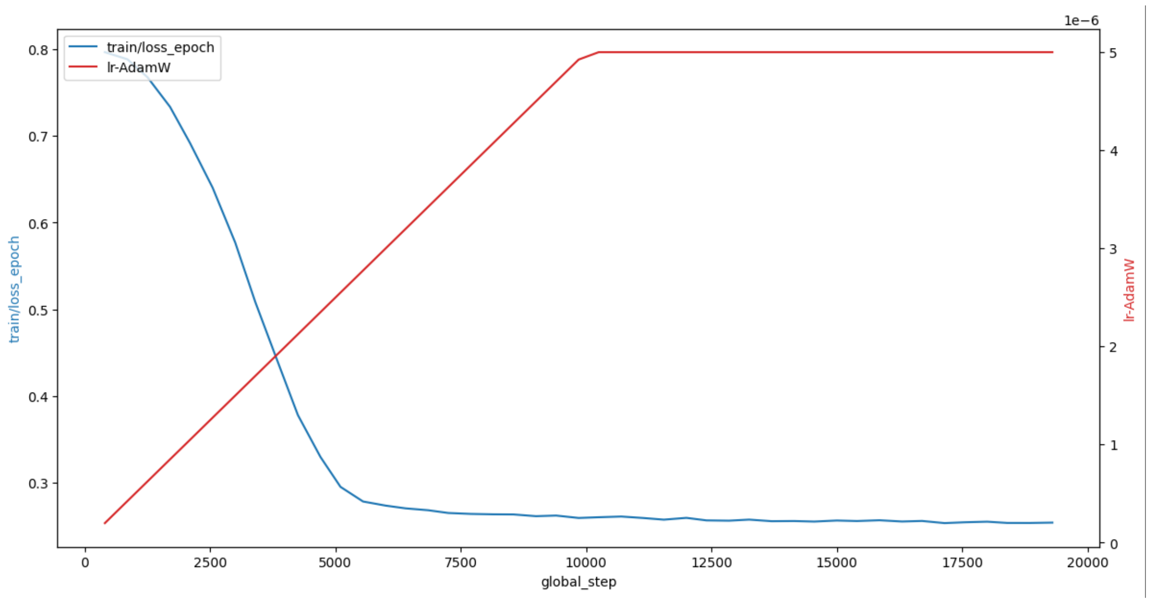


(j)

Figure 6.2: Samples San Francisco original setup

6.2.5 Original Setup with –scale_lr False

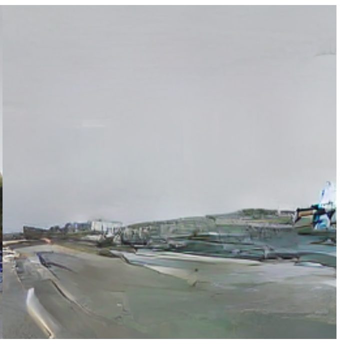
Loss



Samples

(a)

(b)

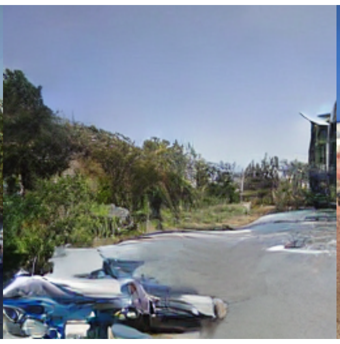

(c)

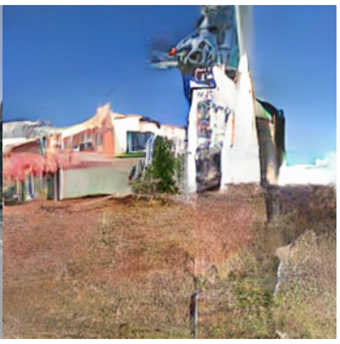
(d)

(e)

(f)

(g)


(h)


(i)

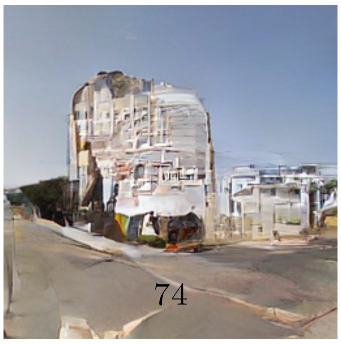
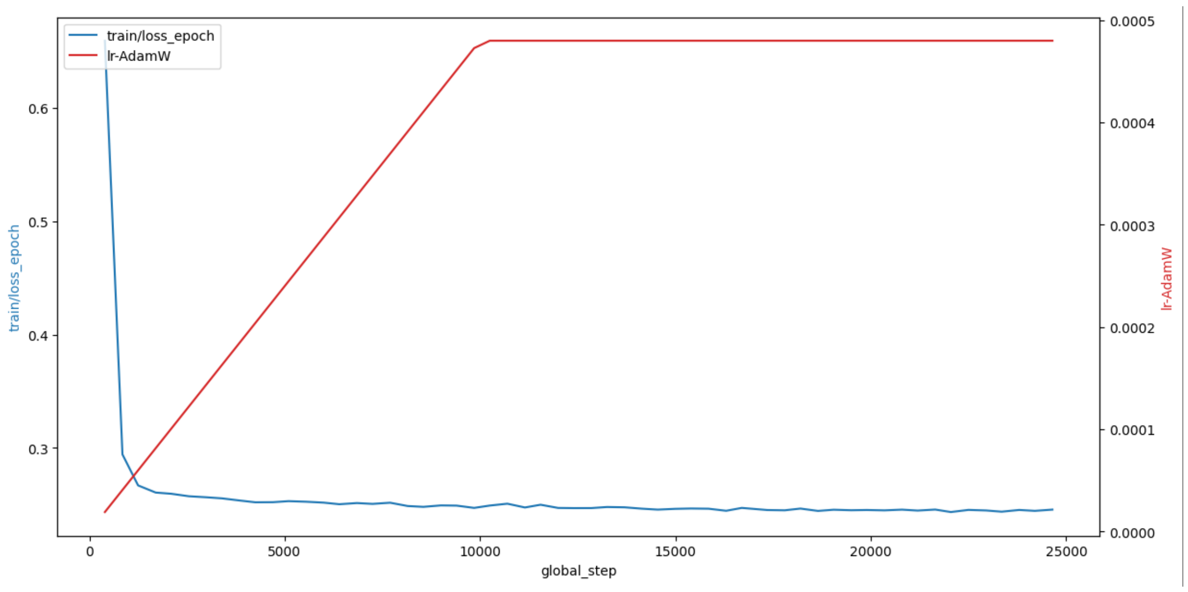

(j)

Figure 6.3: Samples San Francisco original setup with -scale_lr False

6.3 San Francisco ExtraLarge 160 Channels

6.3.1 Original Setup with 160 channels 256×256

Loss



Samples

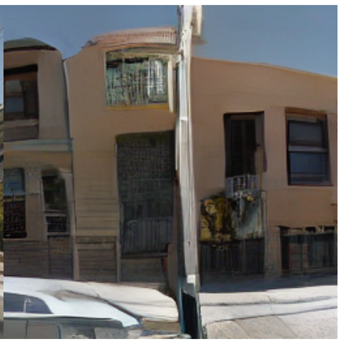
(a)

(b)

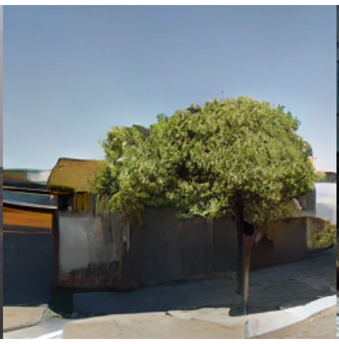
(c)

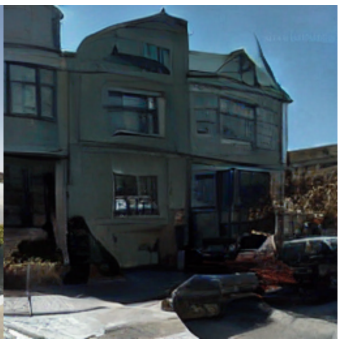
(d)

(e)


(f)

(g)


(h)


(i)

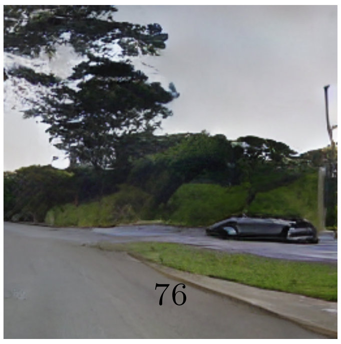
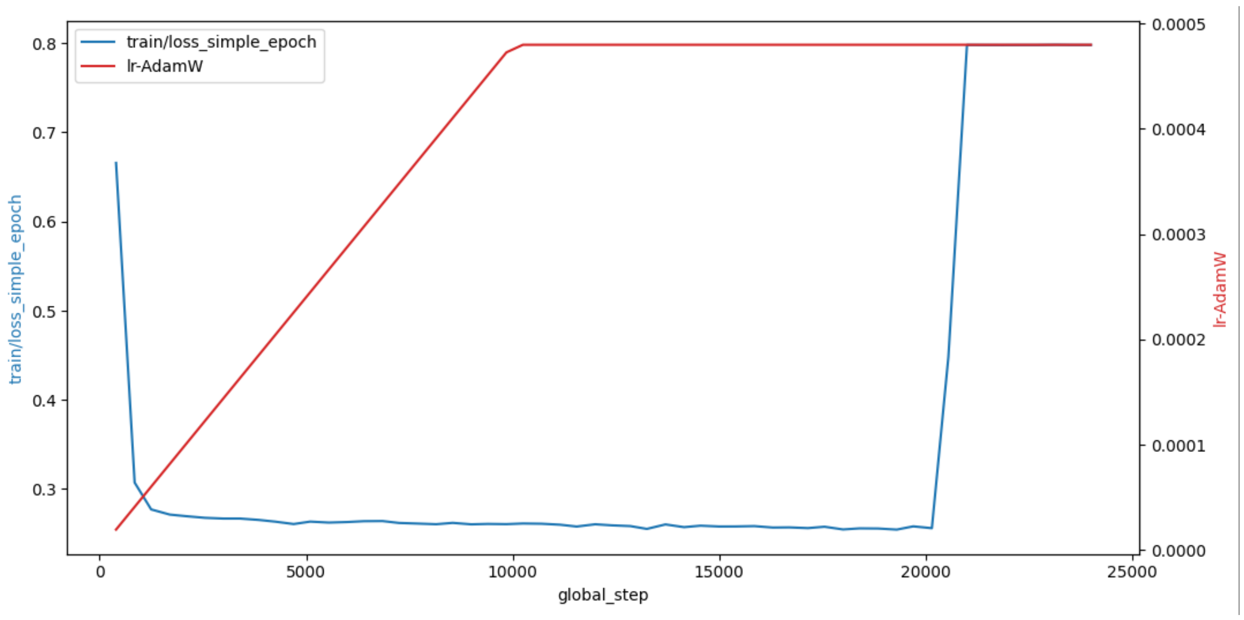

(j)

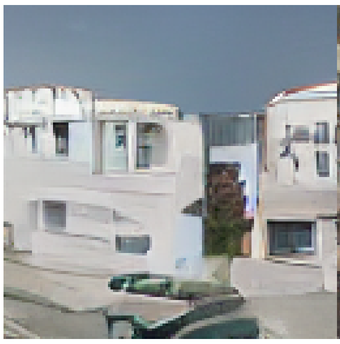
Figure 6.4: Samples San Francisco with 160 channels

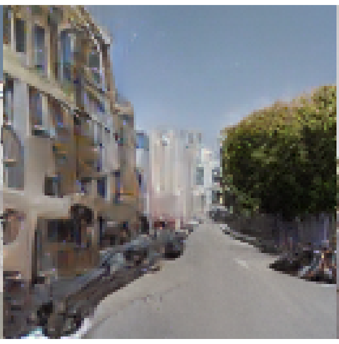
6.3.2 Original Setup with 160 channels 128×128

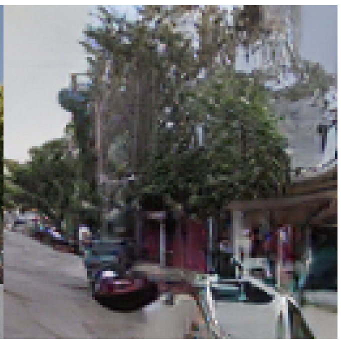
Loss



Samples


(a)


(b)

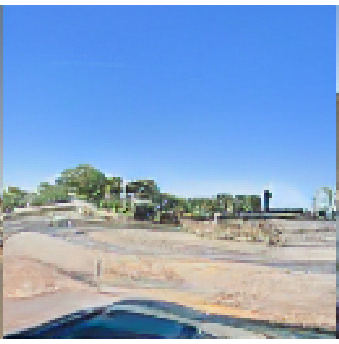

(c)

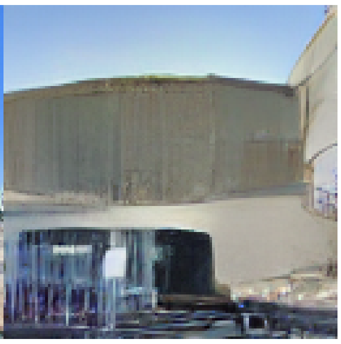
(d)

(e)

(f)

(g)


(h)


(i)

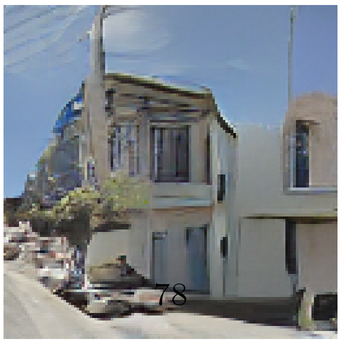
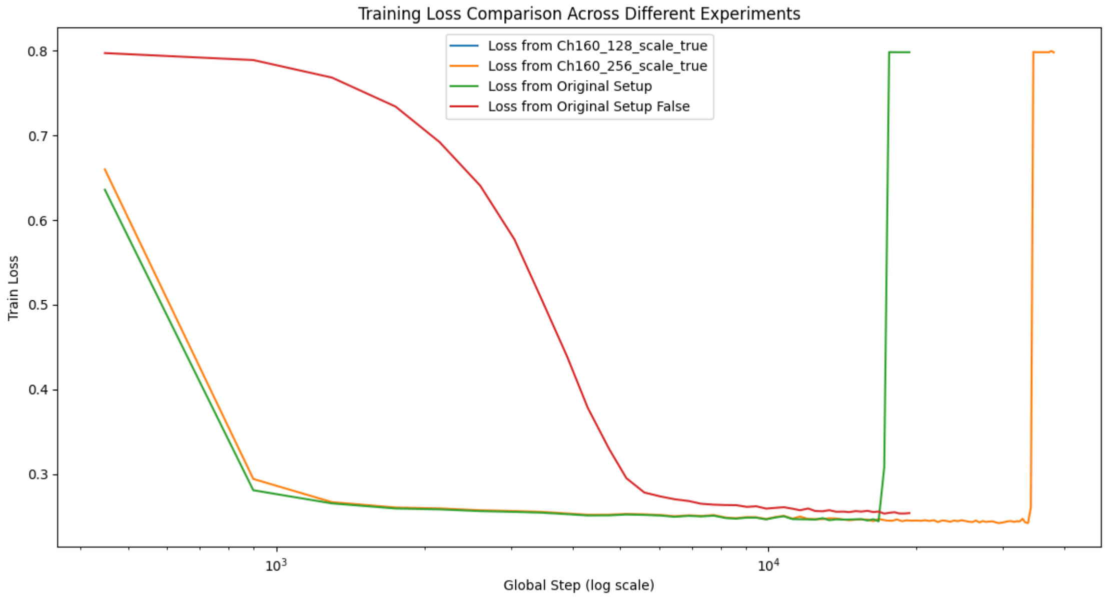

(j)

Figure 6.5: Samples San Francisco with 160 channels 128×128



6.3.3 Discussion on the effectiveness of unconditional generation

The experiments conducted across various datasets and configurations have yielded promising results for the effectiveness of unconditional generation using diffusion models. The findings indicate that the models consistently achieve high precision and exhibit a strong resemblance to the distribution of city images. This performance underscores the models' ability to accurately capture and reproduce the essential features of urban landscapes.

One of the significant implications of this success is the potential for transitioning from **Unconditional** to **Conditional** generation. Since the diffusion models effectively represent the underlying characteristics of the dataset, they can be adapted for conditional generation tasks. This adaptability opens the door for employing these models in applications such as the Cosplace model for geo-localization.

By utilizing the robust feature representation obtained through unconditional generation, we can enhance the interpretability and explainability of the Cosplace model. The insights gleaned from the conditional generation processes will provide a deeper understanding of which features are most influential in the geo-localization task. It allows for identifying the attributes that contribute to successful localization as well as recognizing the conditions under which the model may encounter difficulties or make errors.

6.4 Conditional Generation using Cosplace

6.4.1 Introduction to conditional generation and how the Cosplace network is used

Conditional Generation refers to the process of generating new data (such as images) based on certain input conditions, allowing for greater control and specificity in the generative output. In the context of image generation, these conditions can take various forms, such as **text prompts**, **class labels**, or other features that influence the generated output. One powerful approach for achieving conditional generation is through the use of **cross-attention** mechanisms within models such as diffusion models. **Cross-attention** is a technique that allows the model to focus on specific parts of the conditioning vector during the generation process, effectively guiding the output based on the provided information.
The **conditioning vector** can be any meaningful set of data, and it is incorporated into the model through a specific number of **attention heads**, the individual mechanisms that process different parts of the input, enabling the model to attend to multiple aspects of the conditioning simultaneously. Traditionally, text prompts have been used as the conditioning input, often in natural language processing tasks or image generation models like **DALL · E**, where a description influences the generated image.
In our case, however, instead of using a text prompt, the conditioning input will be derived from the **Cosplace network**. Cosplace is a neural network architecture designed to generate high-quality embeddings of places or scenes, enabling place recognition and comparison. To achieve conditional generation using Cosplace, we first pass the original image through the Cosplace network, which produces a **unique embedding vector**. This embedding represents the key features of the input image and serves as our conditioning input for the generative process.
Simultaneously, the original image is passed to the diffusion model, which is responsible for the actual generation of the new image. The diffusion model works by progressively refining a noisy image, transforming it step by step into a coherent and high-quality output. Throughout this process, the **embedding from Cosplace** is used to condition the generation via cross-attention, ensuring that the resulting image aligns with the features encoded in the original scene.
By using the Cosplace embedding as the conditioning vector, we are able to guide the image generation process to produce outputs that are strongly correlated with the original image's features, but with the flexibility inherent in generative models. This method combines the powerful **scene-recognition** capabilities of Cosplace with the versatile **generation** capabilities of diffusion models, leading to controlled, high-quality results that are informed by the visual characteristics of the input image.

6.5 Integration of Cosplace with Latent Diffusion for Geolocalization

In this section, we describe how the Cosplace model is integrated into the pipeline for latent diffusion-based image generation, specifically for the task of geolocalization. To achieve this, we utilize the `CosplaceImageEmbedder`, a custom class that leverages the Cosplace model to generate embeddings of input images. These embeddings are then used as conditioning vectors in the latent diffusion process, guiding the model to produce outputs that are contextually informed by the geographical features present in the input images.
The `CosplaceImageEmbedder` class, shown below, is built using PyTorch. It initializes with the Cosplace model (with `ResNet50` as the backbone by default), and applies a series of preprocessing steps to the input images before generating the embeddings. The output embeddings are of dimension 512, which are used to condition the diffusion model via cross-attention mechanisms.

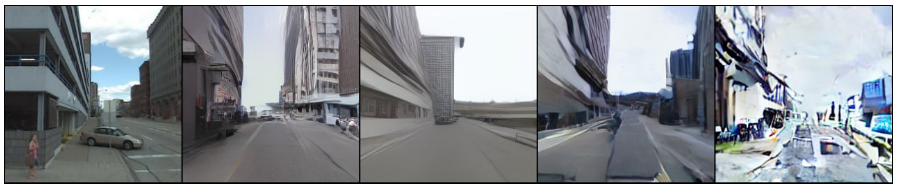
Listing 6.1: CosplaceImageEmbedder class for generating embeddings from the Cosplace model

```python
import torch
import torch.nn as nn
from torchvision import transforms

class CosplaceImageEmbedder(nn.Module):
    """
    Uses the Cosplace model for image embedding.
    """
    def __init__(self, backbone="ResNet50", fc_output_dim=512):
        super().__init__()
        self.model = torch.hub.load("gmberton/cosplace", "get_trained_model",
                                    backbone=backbone, fc_output_dim=fc_output_dim)

        # Define the preprocessing transformations
        self.preprocess = transforms.Compose([
            transforms.Resize(512),
            transforms.CenterCrop(512),
            transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225]),
        ])

    def forward(self, x):
        # If input is a Tensor, assume it's already been preprocessed
        if isinstance(x, torch.Tensor):
            x = (x + 1) / 2  # Rescale values to [0, 1] range
            x = self.preprocess(x)

        """
        # If input is a PIL Image or numpy array, preprocess it
        elif isinstance(x, (Image.Image, np.ndarray)):
            x = self.preprocess(x)
        else:
            raise TypeError("Input should be a PIL Image, ndarray, or
    Tensor")

        if x.dim() == 3:  # If single image, add batch dimension
            x = x.unsqueeze(0)
        """

        # Get the output from the model
        output = self.model(x)

        # Add extra dimension to get shape [1, 1, 512]
        output = output.unsqueeze(1)

        return output # Modify to return self.model(x) if final two
    outputs are removed
```

The `CosplaceImageEmbedder` class is designed to integrate seamlessly into the diffusion model pipeline. During the forward pass, it processes the input image by applying resizing, cropping, and normalization, ensuring that the image is in the proper format for the Cosplace model. The processed image is then passed through the Cosplace model, which generates a 512-dimensional embedding that captures the key features of the scene, specifically tailored for geolocalization tasks. The embedding produced by the Cosplace model serves as a conditioning vector in the latent diffusion process. Through the use of cross-attention mechanisms, this embedding ensures that the diffusion model's output remains consistent with the geographical and contextual features of the input image. This integration allows the model to generate highly relevant images for geolocalization, leveraging both the generative power of latent diffusion and the specialized place-recognition capabilities of Cosplace.

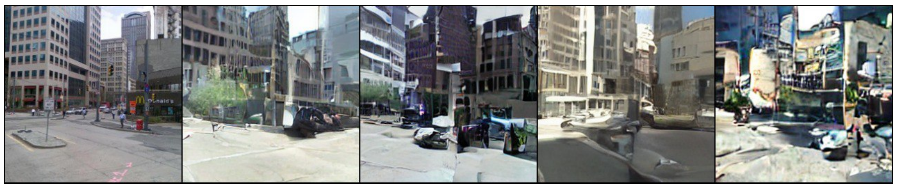
6.5.1 Results: Performance comparison with unconditional generation

In this section, we will analyze the images generated during the training of the conditional model, which was trained on the Pittsburgh Dataset. The analysis will focus on the outputs produced at different stages of the training process, specifically at epochs 7, 14, and 25. This will provide insight into how the model's performance evolves over time and how effectively it learns to generate images conditioned on the input embeddings.

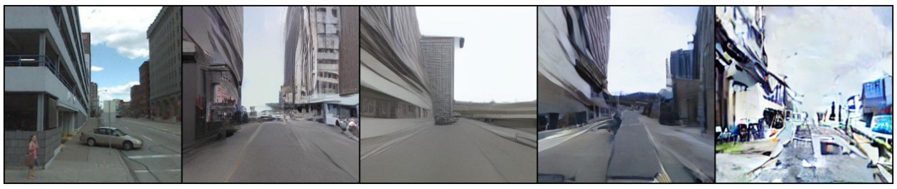


(a)

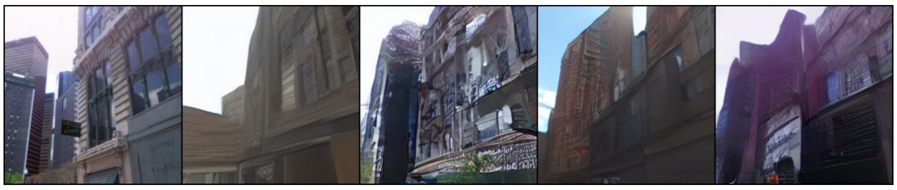
(b)



(c)

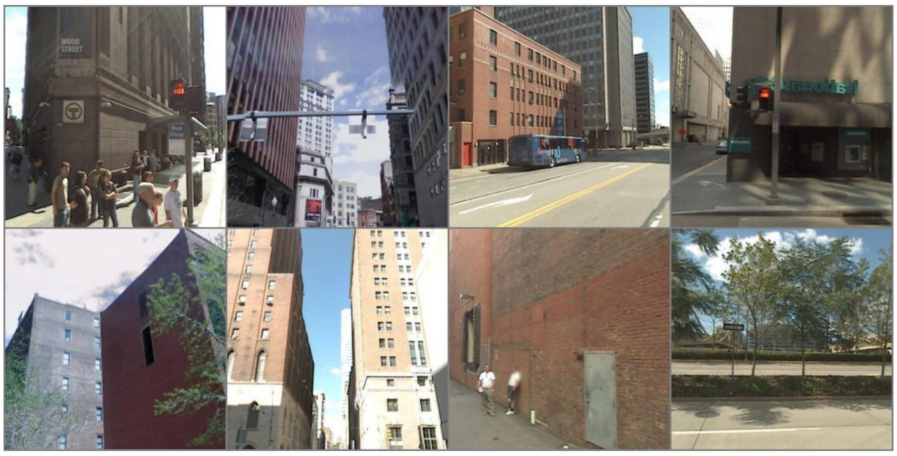


(d)

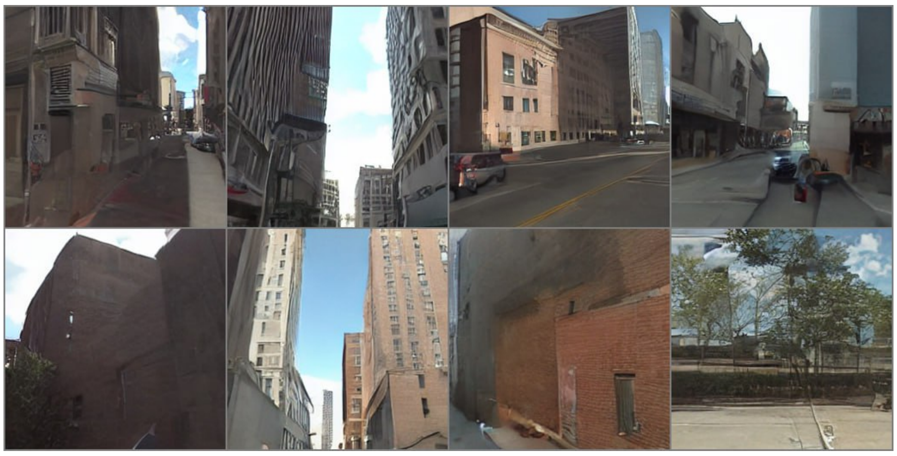


(e)

Figure 6.6: Original and Generated images after 7 epochs

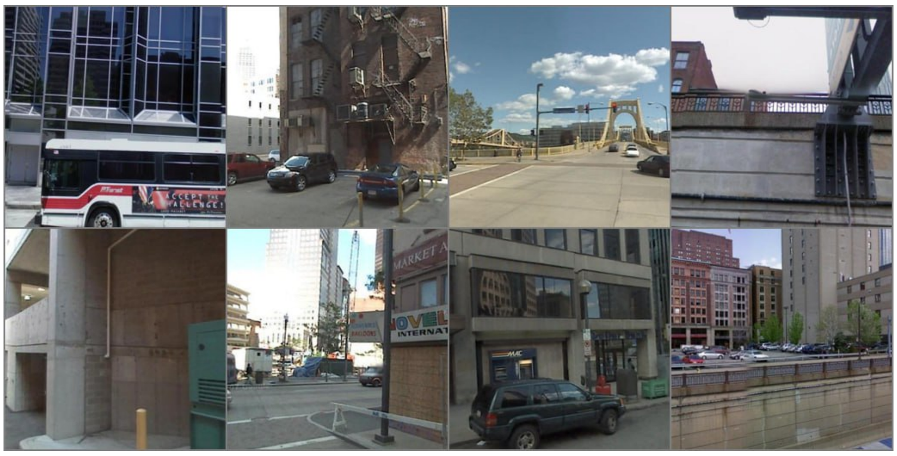


(a)

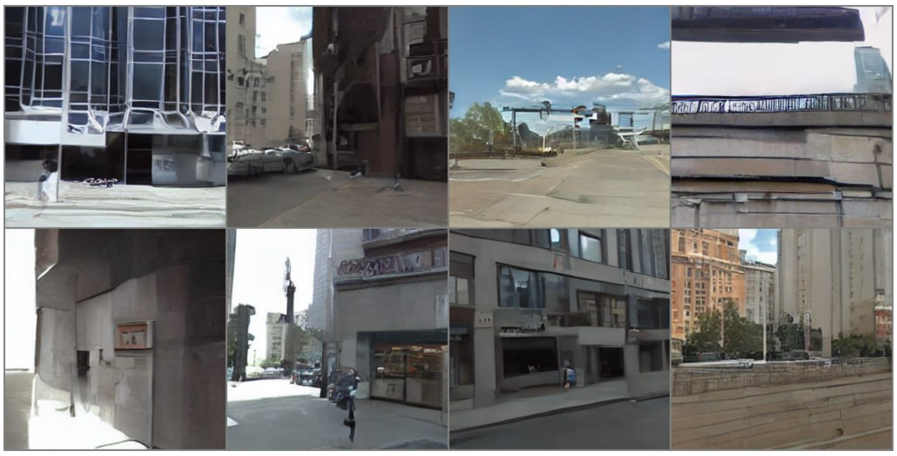


(b)

Figure 6.7: Original and Generated images after 14 epochs



(a)



(b)

Figure 6.8: Original and Generated images after 25 epochs

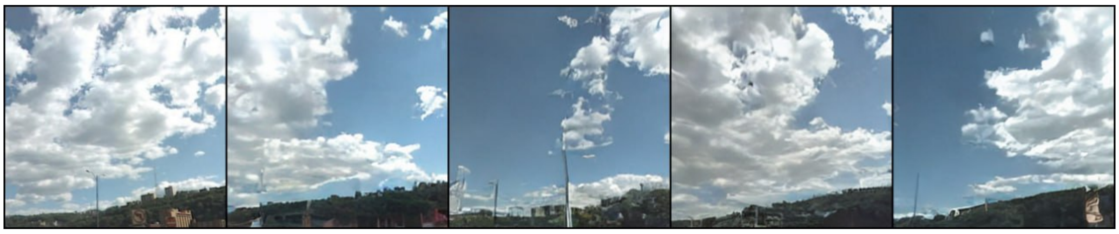
As can be observed from the images, the model's performance is impressive even in the early epochs. By epoch 7, the generated images already display fine details and bear a striking resemblance to the original images, highlighting the effectiveness of the conditional model at capturing key features early in the training process.
However, as the model continues to train through epochs 14 and 25, certain limitations become more apparent. Specifically, it is clear that the Cosplace model, which generates the feature embeddings, has not effectively captured transient visual elements such as moving vehicles or pedestrians. These mobile obstacles, present in the original scenes, are either absent or inaccurately represented in the generated outputs. Additionally, the model struggles with rendering fixed structures like street sign poles, which often do not appear in the generated images.

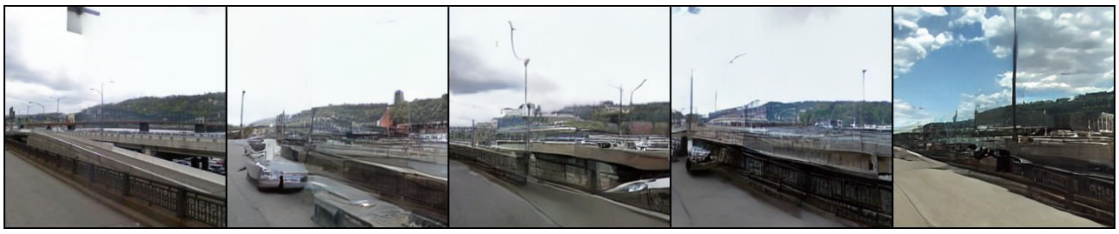
Moreover, the reconstruction of windows also presents challenges for the model. While windows are generated, they often differ in both number and shape compared to the original images, suggesting that the model has difficulty capturing precise architectural details. These issues highlight specific areas where the model's embedding features fall short, particularly in handling dynamic or small-scale elements within the scene.

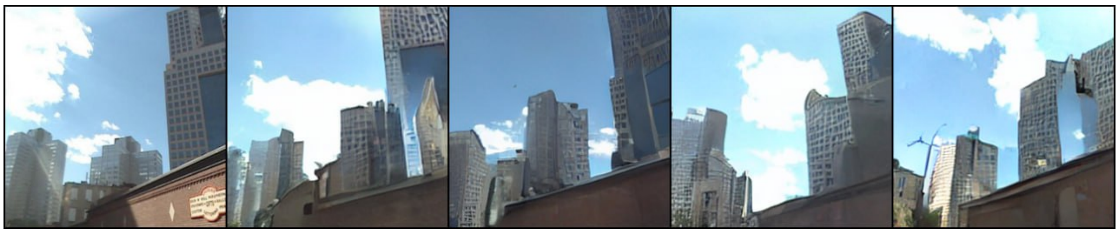
6.5.2 Visual Results and Feature Analysis of Cosplace-Conditioned Image Generation

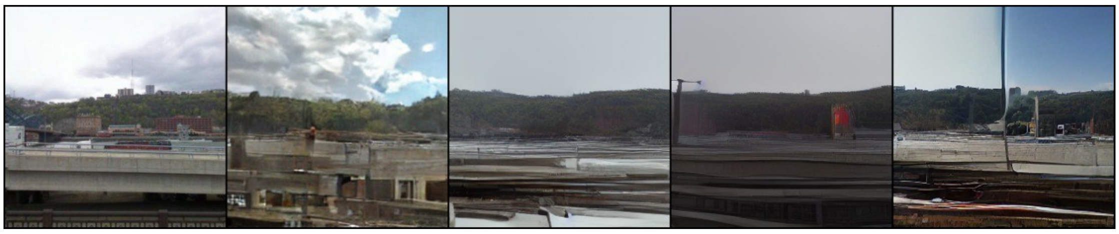
Sky Feature Interpretation

In the following images, the leftmost image is the original input, from which the conditioning embedding is extracted. The subsequent images are generated by the model using the Cosplace-conditioned diffusion process.





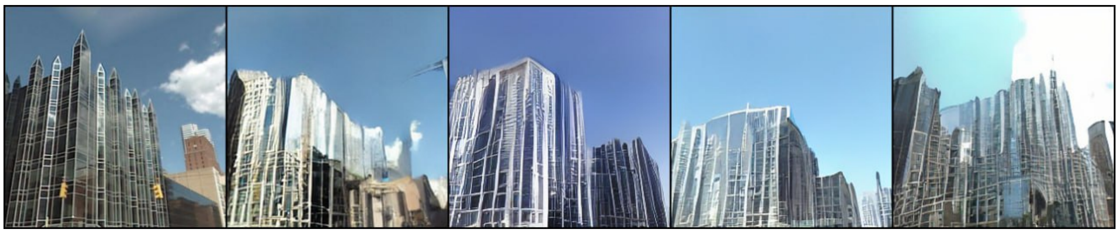


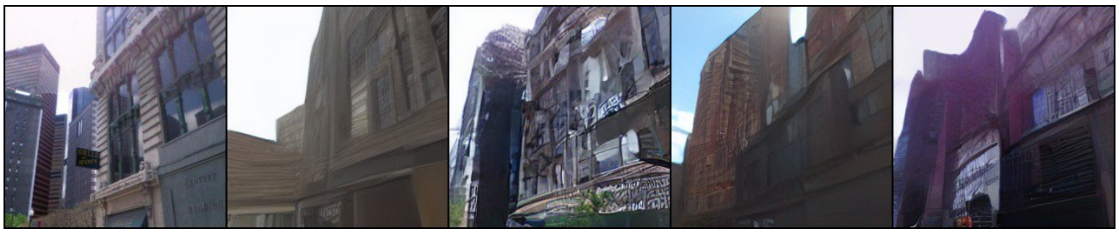


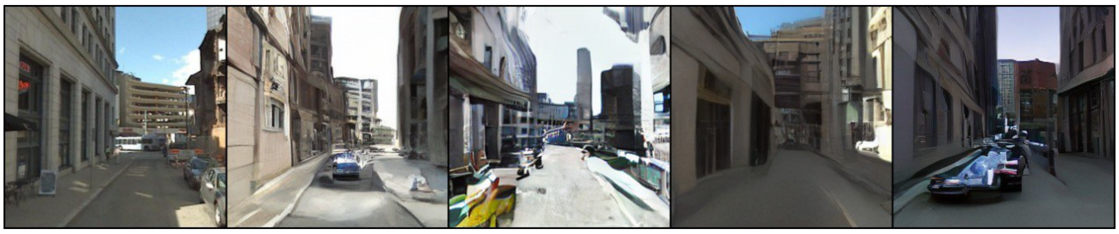
As can be observed from the generated images, the sky tends to differ significantly from the original input. For instance, in cases where clouds are present in the input, the generated images often show a clear sky, or vice versa. This suggests that the sky is not considered a significant feature for Cosplace's embedding in the geolocation task, meaning that it is not crucial for the classification process.
Since the sky is not embedded as an important feature, it is sampled from the distribution governed by the diffusion process, rather than being explicitly retained from the conditioning embedding. This demonstrates that elements like the sky are less relevant for Cosplace's focus, which prioritizes other environmental or structural features that are more critical for geolocation.

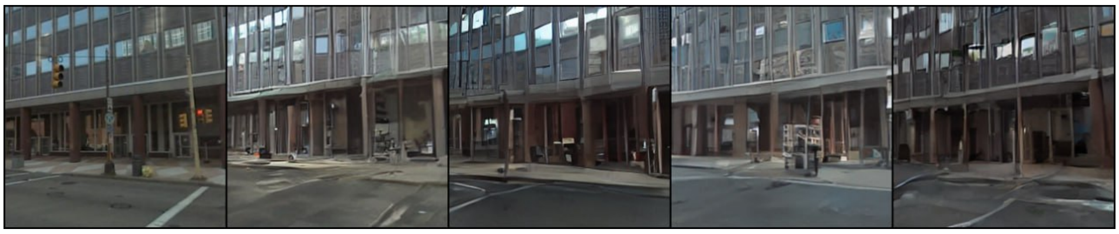
Structural Feature Analysis: Buildings and Roads

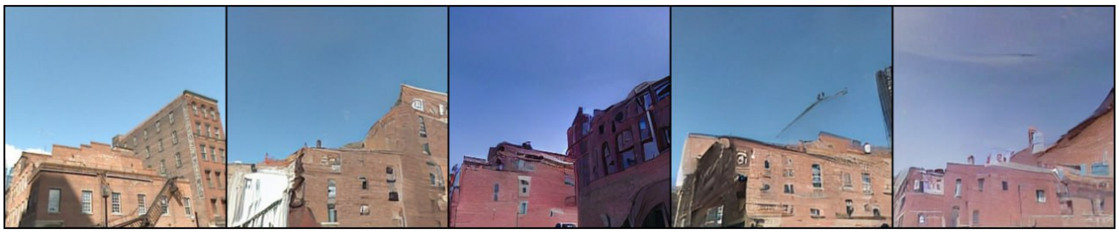
In the following images, the leftmost image is the original input, while the others are generated by the model.

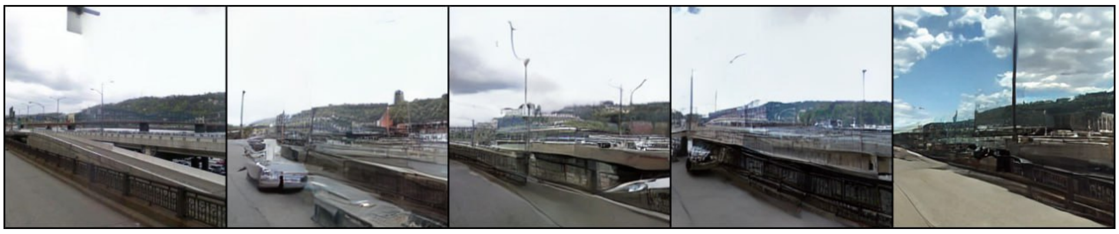










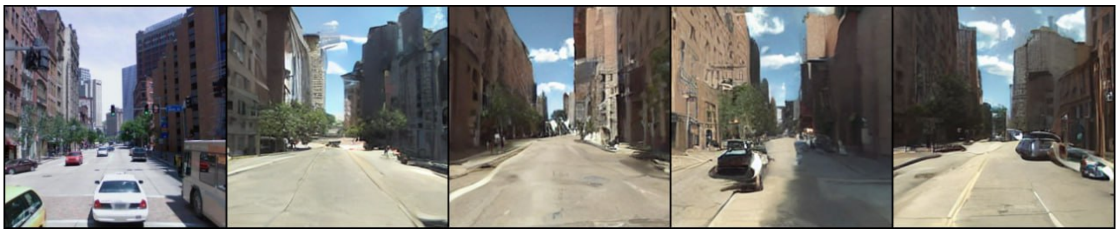


When analyzing the features of roads and buildings, it is clear that the relative positions between the buildings and the streets, as well as their orientation, are well preserved in the generated images. However, it is important to note that the lighting conditions often change. Some images originally in the shade are generated with sunlight, and vice versa. This is a positive outcome, as it shows that the Cosplace model is not sensitive to lighting conditions, making it robust to variations in lighting (i.e., augmentation unaffected).

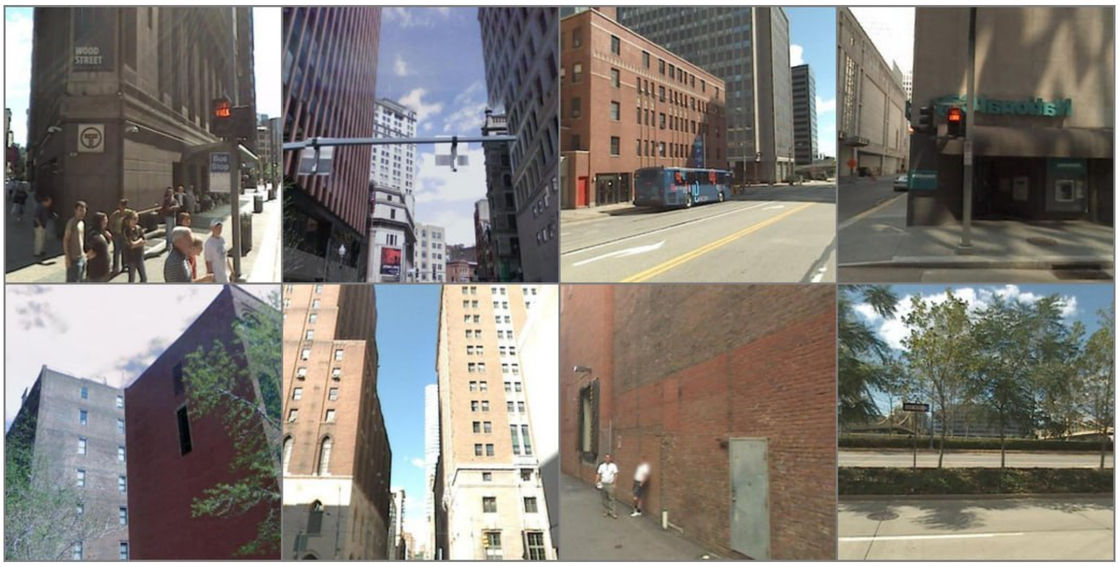
Additionally, the number and placement of windows are not consistently retained in the generated images. This suggests that windows are rarely considered informative features for geolocation. Moreover, given that the dataset spans a period of 12 years, architectural modifications, such as changes in window designs or building colors, are likely. For example, in the second-to-last image, the building's color shifts from orange in the original image to red in the generated one, reflecting potential real-world changes over time.

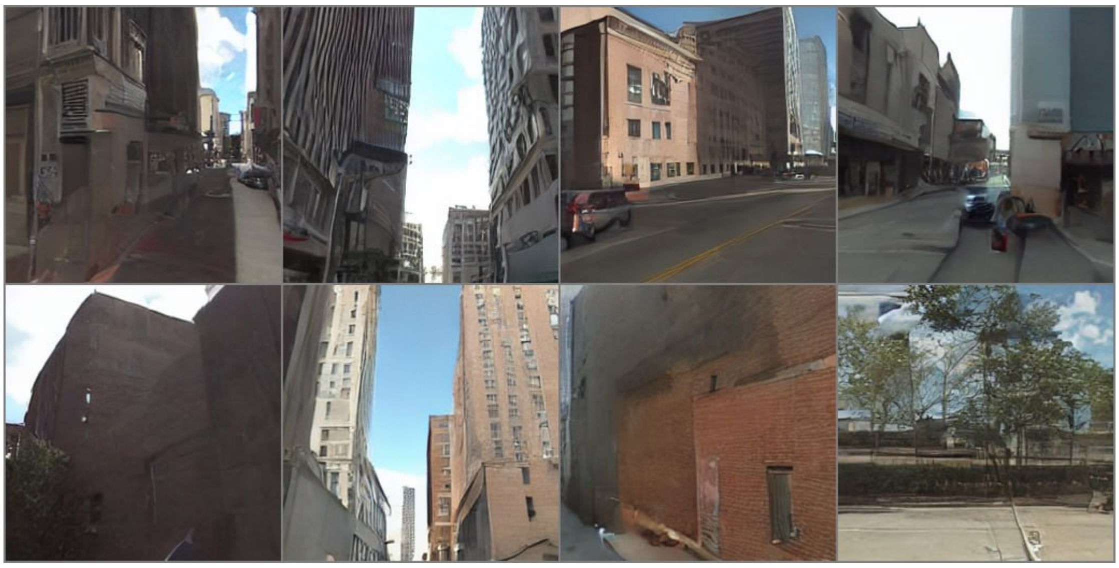
Handling of Dynamic Elements: Vehicles and Pedestrians

In the following image, the original image is on the left, and the generated ones are on the right.



In this set, the original images are shown at the top, with the generated images below.

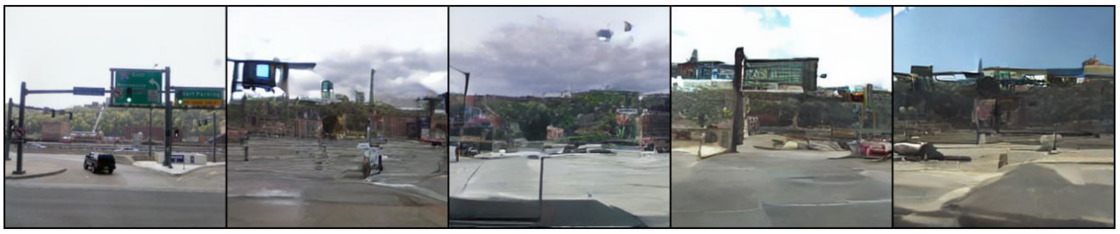




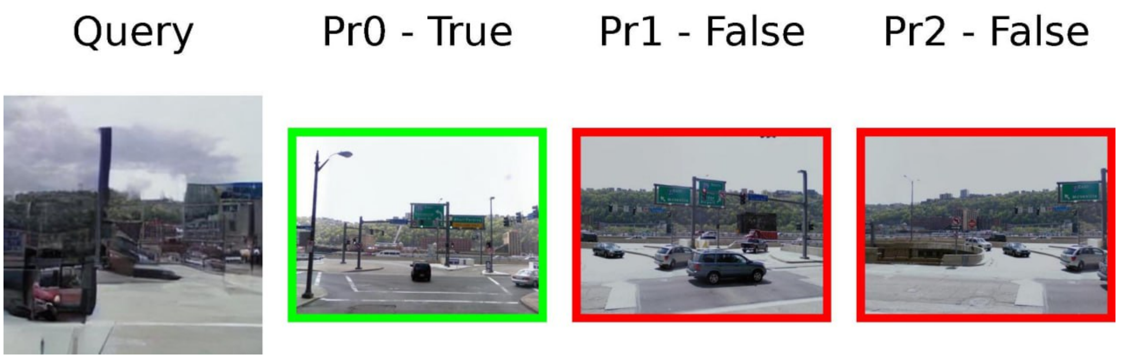
As seen in these examples, mobile obstacles such as cars and pedestrians are often not recognized by the Cosplace model. Since these dynamic objects are not consistently present in the environment, they are not considered informative features for geolocation tasks. As a result, during the generation process, such obstacles are typically removed. This is a positive indication that Cosplace does not focus on non-informative, transient elements, suggesting that it would perform well even in scenarios with varying traffic conditions.
However, in rare cases where a mobile object becomes informative—such as when there are very few images of a location, all taken within a short time frame, and the same vehicle appears in each image—that object may become a significant feature. In such cases, diffusion-based generation can be a valuable tool for identifying when it might be necessary to enrich the dataset to prevent the model from over-relying on such features.

Misrepresentations of Structural and Traffic Elements: Road Signs and Bridges

In the following examples, we can observe how frequently changing road features, such as road signs and lampposts, are not accurately reconstructed. This is because the Cosplace model identifies these elements but struggles to attribute consistent features to them. As a result, the diffusion model generates these objects in completely different forms from the original images, while maintaining their relative positions. This inconsistency highlights Cosplace's difficulty in processing dynamic or less stable features of the environment.



For instance, road signs, which may change in size, shape, or even be absent in certain frames, are not accurately retained during the image generation process. The signs appear in the correct position but vary significantly in appearance compared to the original input images.
This error in Cosplace is further evidenced when an incorrectly generated image is reclassified by the model. Despite the erroneous generation, Cosplace still classifies it correctly because it focuses on identical, stable features in both the original and generated images. This behavior can be seen in the case below:



This shows that while Cosplace may not consistently capture traffic-related or structural elements like road signs and overpasses, it continues to recognize more stable and permanent features for classification purposes, ignoring minor variations in less significant elements.

6.5.3 Analysis of the strengths and limitations of the approach

The approach demonstrated here has proven to be both highly effective and innovative in enhancing the explainability of the base model. By integrating a conditional generation process, we gained deeper insights into how the Cosplace model operates and how it captures scene-specific features through its embeddings. This allowed for a more transparent understanding of the model's decision-making process, offering clarity into what the model perceives as important for geolocalization tasks.
Additionally, the use of a diffusion model as a tool for explainability is particularly novel and scarcely explored in existing literature. The diffusion model enabled us to visualize how the model responds to specific features during the generation process, offering a new dimension of explainability beyond traditional methods. This represents a significant advancement in the field, as generative models like diffusion have not been widely utilized for such purposes.
However, while this approach excels in improving the model's transparency and providing a more interpretable framework, certain limitations were noted, such as difficulties in reconstructing dynamic or small-scale visual elements, as discussed earlier. Despite these limitations, the innovative combination of Cosplace and diffusion models marks a valuable contribution to the explainability of complex neural networks.

Chapter 7

Conclusion and Future Work

7.1 Summary of Findings

7.1.1 Recap of key insights from experiments and generative models

The experiments conducted in this research have highlighted several important insights regarding the integration of the Cosplace model with latent diffusion for geolocation tasks. The conditional generative approach proved effective at generating images that closely resemble the original scenes, particularly in early epochs, demonstrating that this method can capture a wide range of visual details. The model also provided valuable insights into how Cosplace embeddings function, specifically in geolocalization contexts. Moreover, the use of diffusion models for explainability emerged as a novel and promising direction, offering visual outputs that clarify the model's internal decision-making process in a way that traditional methods could not.

7.1.2 Implications for AI-driven geolocation tasks

The findings from this work have direct implications for AI-driven geolocation tasks. The integration of conditional generation allows for a better understanding of what features are emphasized by neural networks like Cosplace, thus enhancing both the accuracy and transparency of location-based models. This is especially valuable in critical applications such as autonomous navigation, smart city management, and urban planning, where explainability and precision are crucial. The use of generative AI in this context opens new avenues for improving model robustness and interpretability, paving the way for more reliable AI-driven geolocation systems.

7.2 Limitations of the Current Work

7.2.1 Challenges faced during experiments (e.g., computational costs, model limitations)

Despite the promising results, several challenges were encountered throughout the research process. One of the main issues was the need to reduce the number of channels in the model architecture. The original repository of the Latent Diffusion Model by CompVis was trained using an NVIDIA A100 GPU, a high-end device with greater computational capabilities. However, due to hardware constraints, this work was conducted on an NVIDIA RTX Titan with 24 GB of memory and GTX 1070 cards with only 8 GB. This led to significant compromises in model complexity and batch size, limiting the speed and scale of experimentation.
Additionally, the images processed in this work had a resolution of 256x256, which, while suitable for initial experimentation, resulted in a lower level of detail in the generated outputs. This may have constrained the precision of the model when handling intricate structures like windows or small dynamic elements such as pedestrians and moving vehicles.

7.3 Future Research Directions

7.3.1 Possible improvements to latent diffusion models

There are several potential avenues for improving latent diffusion models in future work. One direction could involve optimizing model architectures to work more efficiently on hardware with limited computational resources, such as by incorporating more efficient layers or compression techniques. Another possibility is to explore alternative training strategies that allow for better scaling with smaller batch sizes, which could help overcome the current hardware limitations.
Furthermore, increasing the resolution of the generated images and improving the model's ability to capture fine details will be critical. This could be achieved by adopting multi-scale training approaches or by experimenting with higher-resolution diffusion models. The development of more advanced conditioning mechanisms might also help improve the model's ability to generate more realistic reconstructions of dynamic and small-scale elements in urban environments.

7.3.2 Further applications of generative AI in geospatial tasks and in Explainability

The success of this diffusion-based explainability approach suggests that it could be extended to a wide variety of research domains beyond outdoor geolocalization.

For example, indoor visual geolocation systems could benefit greatly from this method, helping AI models learn and explain how they recognize specific indoor environments based on spatial cues.
In addition to geolocation, diffusion models for explainability could be applied in several other fields:

- **Genetic code analysis**: Diffusion models could help visualize and explain how AI systems interpret and generate synthetic genetic sequences, improving our understanding of AI in genomics and bioinformatics.
- **Weather prediction**: The ability of diffusion models to generate high-quality visualizations could aid in interpreting complex atmospheric models, offering clearer explanations for weather forecasts and helping researchers better understand AI-based climate models.
- **Economic forecasting**: In economic modeling, diffusion approaches could be used to generate interpretable scenarios that clarify how AI predicts market trends or economic outcomes, providing greater transparency for decision-makers.

These examples demonstrate the broad applicability of diffusion models in enhancing the explainability of AI systems across a range of disciplines. Future research could explore these avenues, applying generative models to offer clearer, more interpretable outputs in various scientific and practical fields.

Bibliography

[1] Vikramkumar, Vijaykumar B, and Trilochan. *Bayes and Naive Bayes Classifier.* 2014. arXiv: `1404.0933 [cs.LG]`. URL: `https://arxiv.org/abs/1404.0933` (cit. on p. 5).

[2] Jürgen Schmidhuber. «Deep learning in neural networks: An overview». In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: `10.1016/j.neunet.2014.09.003`. URL: `http://dx.doi.org/10.1016/j.neunet.2014.09.003` (cit. on p. 6).

[3] Christian Ledig et al. *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.* 2017. arXiv: `1609.04802 [cs.CV]`. URL: `https://arxiv.org/abs/1609.04802` (cit. on p. 15).

[4] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. *AlphaFold Meets Flow Matching for Generating Protein Ensembles.* 2024. arXiv: `2402.04845 [q-bio.BM]`. URL: `https://arxiv.org/abs/2402.04845` (cit. on p. 17).

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks.* 2014. arXiv: `1406.2661 [stat.ML]`. URL: `https://arxiv.org/abs/1406.2661` (cit. on pp. 22, 24).

[6] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes.* 2022. arXiv: `1312.6114 [stat.ML]`. URL: `https://arxiv.org/abs/1312.6114` (cit. on p. 23).

[7] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis.* 2023. arXiv: `2307.01952 [cs.CV]`. URL: `https://arxiv.org/abs/2307.01952` (cit. on p. 24).

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN.* 2017. arXiv: `1701.07875 [stat.ML]`. URL: `https://arxiv.org/abs/1701.07875` (cit. on p. 27).

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. «Denoising Diffusion Probabilistic Models». In: *CoRR* abs/2006.11239 (2020). arXiv: `2006.11239`. URL: `https://arxiv.org/abs/2006.11239` (cit. on p. 27).

[10] Yuki Yasuda and Ryo Onishi. *Theory of Super-Resolution Data Assimilation with Conditional Variational Autoencoders: Using Super-Resolution Operators as Background Error Covariance Matrices*. 2024. arXiv: `2308.03351 [physics.ao-ph]`. URL: `https://arxiv.org/abs/2308.03351` (cit. on p. 38).

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. «Denoising Diffusion Implicit Models». In: *CoRR* abs/2010.02502 (2020). arXiv: `2010.02502`. URL: `https://arxiv.org/abs/2010.02502` (cit. on p. 41).

[12] Gabriele Berton, Carlo Masone, and Barbara Caputo. «Rethinking Visual Geo-Localization for Large-Scale Applications». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 4878–4888 (cit. on p. 51).

[13] Andreas Blattmann, Robin Rombach, Kaan Oktay, and Björn Ommer. *Retrieval-Augmented Diffusion Models*. 2022. DOI: `10.48550/ARXIV.2204.11824`. URL: `https://arxiv.org/abs/2204.11824` (cit. on p. 54).

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: `2112.10752 [cs.CV]` (cit. on p. 57).