SUBJECT: GENERATIVE AI

Introduction

With advancements in image generation, **Diffusion Probabilistic Models (DPMs)** have gained prominence for denoising and generating high-quality images. DPMs utilise a parameterized Markov chain, trained through variational inference, to reverse a diffusion process, where noise is incrementally added to data. This gradual noise addition mimics a sampling process, allowing the model to reconstruct the original signal accurately. By adding small amounts of Gaussian noise and then reversing the process, DPMs offer a robust approach to high-fidelity image generation and denoising.

Probabilistic models, such as DPMs, use mathematical frameworks to handle uncertainty, making them highly applicable in fields like machine learning and artificial intelligence. They also support applications in super-resolution, inpainting, and style transfer. Due to their stability, flexibility, and improved quality in generated outputs, DPMs offer significant advantages over traditional generative adversarial networks (GANs), especially for complex image structures.

Objective

The primary objective of this project is to develop a DPM-based denoising model that can effectively remove noise from images while retaining their quality and detail. Through this objective, the project aims to:

1. Explore the application of DPMs for stable, controllable, and high-quality denoising.
2. Evaluate the model's performance against GANs and other generative models in terms of image fidelity, stability, and control over generated samples.
3. Investigate DPM's adaptability across tasks such as denoising, super-resolution, inpainting, and style transfer.

Methodology

- **Diffusion Probabilistic Model (DPM)**:
    - A diffusion probabilistic model follows a Markov chain, gradually adding noise until the signal is entirely obscured.
    - Gaussian noise is used, and the sampling chain transitions are set to conditional Gaussians, allowing for a simplified neural network parameterization.

- **Probabilistic Framework**:
  - Probabilistic models provide a structure for analysing random events and variables.
  - Key components include random variables, probability distributions, joint probability distributions, and conditional probabilities.
- **Generative Models**:
  - Generative models such as DPMs aim to recreate data that resembles observed data.
  - Unlike GANs, DPMs do not suffer from issues like mode collapse, ensuring consistent and high-diversity sample generation.
- **Implementation**:
  - DPMs were trained to incrementally add noise and reverse the process, offering a high level of control over generated images.
  - This training allowed the model to handle various tasks beyond image denoising, such as super-resolution and style transfer

LangChain Components

1.Models

LLMs:

- Diffusion Probabilistic Models (DPMs) represent a parameterized probabilistic framework optimized for tasks such as denoising, super-resolution, and inpainting.
- Suitable for generative AI models using Markov chains and Gaussian noise for training.

Chat Models:

- Although not explicitly mentioned, extensions could include user-interactive models that explain DPMs or help users fine-tune models for specific tasks.

Text Embedding Models:

- Embedding models could be applied to represent multimodal data (e.g., image-text pairs) mentioned under enhancements for handling complex data distributions.

2. Prompts

1. Volumes

In LangChain, **volumes** refer to the diversity and variety of data used to create prompts, fine-tune models, and generate robust outputs. In the context of **DPMs**:

- **Training Data Volume**:
  - The **volume of training data** affects the model's ability to learn generalizable features.
  - Example: The report uses the *Oxford Flowers 102* dataset, which provides a sufficient volume of diverse floral images to ensure the model can generalize across various styles and details.
- **Noise Schedule Volume**:
  - Each step in the noise schedule adds a new "volume" of Gaussian noise. Managing these volumes effectively is crucial for ensuring a smooth retrieval process.
  - Example: Adjusting the number of steps in the noise schedule influences the fidelity of the retrieval process during denoising.
- **Task-Specific Volumes**:
  - Volumes vary depending on tasks like **denoising**, **super-resolution**, or **inpainting**. For instance:
    - **Super-resolution**: Requires high-volume paired datasets of low- and high-resolution images.
    - **Inpainting**: Requires datasets with incomplete images and ground truth for comparison.

2. Templates:

- Examples like the `resize_and_rescale` and `train_preprocessing` functions in the code snippets act as prompts/templates for training DPM models.

Example-Driven Templates:

- The report provides **examples** such as augmenting images using flipping, resizing, and normalization. These templates serve as benchmarks for developing customized training pipelines.

Example Section:

- Case studies of DPM applications in tasks such as super-resolution, denoising, and inpainting provide specific examples to guide prompt engineering.

Output Parameters:

- Parameters like noise schedules, image fidelity, and quality metrics form the core output parameters for DPMs.

3. Indexes

Document Loader:

- The report itself serves as a structured document loader for understanding DPM architecture, applications, and enhancements.

Splits:

- Training and testing datasets are split, as illustrated by loading the "Oxford Flowers 102" dataset and processing it for augmentation and scaling
- T**raining, Validation, and Testing**:
- As with any machine learning model, DPMs require splitting datasets into:
    - **Training**: For learning noise schedules and generating denoised outputs.
    - **Validation**: For tuning hyperparameters like the learning rate, diffusion steps, and noise schedule.
    - **Testing**: For evaluating the model's performance on unseen data

Vectors:

- Vectorized representations could be derived from embeddings generated during DPM training for images or image-text pairs.

Here, we have mainly used,

- **Gaussian Vectors**:
    - The DPM methodology relies heavily on Gaussian noise addition during the forward diffusion process. These Gaussian vectors represent the incremental noise applied to the data.
    - Gaussian vectors are used not only to obscure the signal but also in reverse to reconstruct high-fidelity outputs during denoising.
- **Feature Vectors**:

    - The embeddings derived from the diffusion process could be converted into feature vectors, representing specific image features like texture, color gradients, or object boundaries.
    - These feature vectors are crucial for tasks such as inpainting and super-resolution.
- **Multimodal Vectors**:
    - As proposed under enhancements, DPMs can handle **multimodal data** (e.g., image-text pairs). This would involve creating and integrating vector representations for both modalities, ensuring cross-modal dependency capture.
- **Diffusion Step Vectors**:
    - Each step in the Markov chain of the diffusion process can be encoded into vectors, tracking how noise is applied and reversed at each stage.
    - These vectors help analyze the progression of the denoising process and allow for fine-tuning the noise schedule.
- **Attention-Driven Vectors**:
    - Using attention mechanisms (proposed under enhancements) can produce attention-based vectors, emphasizing important regions in an image or aspects in multimodal tasks.
    - These vectors aid in improving the DPM's understanding of long-range dependencies and complex structures.

Retrieval:

- The use of retrieval mechanisms, such as doc loaders for datasets or theoretical references, is evident in model training and application setups

1. Noise Addition and Retrieval

- In DPMs, the forward diffusion process involves gradually adding **Gaussian noise** to the data through a **Markov chain**.
- **Retrieval** in this context is the reverse process, where the model sequentially **retrieves or reconstructs** the original data by denoising, one step at a time.

2. Sequential Retrieval of States

- The **Markov chain property** ensures that each state (i.e., the image at a given noise level) depends only on the previous state. This enables efficient step-by-step retrieval of the original image through:
    - **Conditional probabilities**: p(xt|xt+1)p(x_t \mid x_{t+1})p(xt|xt+1) These represent the likelihood of retrieving a less noisy image xtx_txt given a noisier image xt+1x_{t+1}xt+1.

    - By using these probabilities, the model "retrieves" the cleaner version of the data iteratively.

3. Vector Retrieval

- During the reverse process, **Gaussian vectors** encode the noise patterns at each step. These vectors serve as intermediaries for retrieving high-fidelity representations.
- For enhanced **retrieval efficiency**, vectors representing key features (e.g., edges, textures) can be prioritized during the denoising steps.

4. Noise Schedule and Efficient Retrieval

- The noise schedule governs how noise is added during the forward process and how it's retrieved in reverse. A well-designed noise schedule ensures:
    - **Accurate retrieval** of fine-grained details.
    - **Control over retrieval fidelity**, allowing users to balance speed and quality

4. Memory

Chains:

- The methodology of DPMs, including their Markov chain implementation, is a direct application of memory chains in diffusion processes.

LChain:

- Model optimization through parameter tuning (e.g., noise schedules) can be visualized as a linked chain of iterative improvements.

Prompts:

- Interaction between the diffusion process and generated outputs acts as an implicit memory prompt for maintaining high-fidelity results.

5. Agents

- **Tools**:

TensorFlow libraries and preprocessing functions (e.g., `resize_and_rescale`) serve as essential tools for implementing DPMs.

- **Toolkits**:

  The provided Python functions form a toolkit for image preprocessing and augmentation in DPM training workflows.

o .

Results

The diffusion probabilistic model demonstrated superior performance in several key areas:

1. **Stability**: Unlike GANs, DPMs showed increased stability with no significant instances of mode collapse.
2. **Controllability**: Adjusting diffusion process parameters (e.g., noise schedule) enabled control over image style, quality, and detail preservation.
3. **Flexibility and Quality**: DPMs handled denoising tasks effectively and produced high-quality images with finer details. The model also successfully completed other tasks like inpainting and super-resolution.
4. **Enhanced Theoretical Understanding**: The model's strong foundation in diffusion processes provided valuable insights into model behaviour and performance.

Code Snippets:

```
# general imports

import math

import numpy as np

import matplotlib.pyplot as plt


# TF imports

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

import tensorflow_datasets as tfds
```

In [3]:

```
batch_size = 32 # images per batch

num_epochs = 1 # number of train iterations
```

total_timesteps = 1000

norm_groups = 8 *# number of groups normalized at once*

learning_rate = 2e-4 *# speed the network updates weights*

img_size = 64 *# image dimensions*

img_channels = 3 *# RGB channels*

clip_min = -1.0 *# clip min*

clip_max = 1.0 *# clip max*

first_conv_channels = 64 *# number of channels in first layer*

channel_multiplier = [1, 2, 4, 8] *# how much to multiply the number of channels by*

widths = [first_conv_channels * mult for mult in channel_multiplier]

has_attention = [False, False, True, True]

num_res_blocks = 2 *# Number of residual blocks*

dataset_name = "oxford_flowers102"

splits = ["train"]

load data

(ds,) = tfds**.**load(dataset_name, split=splits, with_info=**False**, shuffle_files=**True**)

Output:

```
Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to C:\U
sers\carso\tensorflow_datasets\oxford_flowers102\2.1.1...
Dl Completed...: 0 url [00:00, ? url/s]
Dl Size...: 0 MiB [00:00, ? MiB/s]
Extraction completed...: 0 file [00:00, ? file/s]
Generating splits...:   0%|          | 0/3 [00:00<?, ? splits/s]
Generating train examples...: 0 examples [00:00, ? examples/s]
Shuffling C:\Users\carso\tensorflow_datasets\oxford_flowers102\2.1.1.incomplete5AHE2R\oxford_flowers102-train.…
Generating test examples...: 0 examples [00:00, ? examples/s]
Shuffling C:\Users\carso\tensorflow_datasets\oxford_flowers102\2.1.1.incomplete5AHE2R\oxford_flowers102-test.t…
Generating validation examples...: 0 examples [00:00, ? examples/s]
Shuffling C:\Users\carso\tensorflow_datasets\oxford_flowers102\2.1.1.incomplete5AHE2R\oxford_flowers102-valida…
Dataset oxford_flowers102 downloaded and prepared to C:\Users\carso\tensorflow_datasets\oxford_flowers102\2.1.1. Subsequent c
alls will reuse this data.
```

 # flip an image randomly

```
def augment(img):
    return tf.image.random_flip_left_right(img)
```

In [6]:

```
# resizes given image into given shape and rescales values to [-1.0, 1.0]
def resize_and_rescale(img, size):
  height = tf.shape(img)[0]
  width = tf.shape(img)[1]
  crop_size = tf.minimum(height, width)

  # crop
  img = tf.image.crop_to_bounding_box(
    img,
    (height - crop_size) // 2,
    (width - crop_size) // 2,
    crop_size,
    crop_size,
  )

  # resize
  img = tf.cast(img, dtype=tf.float32)
  img = tf.image.resize(img, size=size, antialias=True)

  # rescale
  img = img / 127.5 - 1.0
  img = tf.clip_by_value(img, clip_min, clip_max)
  return img
```

In [7]:

```
# resize, rescale, and augment given data
def train_preprocessing(x):
  img = x["image"]
  img = resize_and_rescale(img, size=(img_size, img_size))
  img = augment(img)
  return img
```

Enhancements

To further enhance the performance and applicability of DPMs, the following improvements are proposed:

1. **Improving Efficiency and Scalability**:
    - **Efficient Sampling**: Develop optimised sampling algorithms, such as Metropolis-Hastings or Hamiltonian Monte Carlo, to reduce computational costs.
    - **Parallel and Distributed Training**: Implement parallel computing techniques to improve scalability and reduce training times on large datasets.
2. **Handling Complex Data Distributions**:
    - **Multimodal Data**: Develop models that can handle complex, multimodal data, such as image-text pairs, and enhance DPM's capabilities to capture cross-modal dependencies.
    - **Long-Range Dependencies**: Use attention mechanisms and hierarchical structures to capture long-range dependencies, critical for tasks like image generation and natural language processing.
3. **Theoretical Analysis and Understanding**:
    - **Convergence Analysis**: Investigate convergence properties under varying noise schedules and diffusion steps to improve model behaviour.
    - **Generative Modeling Theory**: Explore connections between DPMs and frameworks like variational autoencoders (VAEs) and GANs to develop hybrid models with enhanced capabilities.
4. **Expanding Applications and Customization**:

    New Applications: Explore applications in fields like drug discovery, materials science, and climate modelling.

    - **Conditional Diffusion Models**: Implement conditional models to generate samples based on specific inputs, enabling style transfer and customization.

References

1. **Denoising Diffusion Probabilistic Models**
   Link:
   https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
2. **3D Medical Image Generation Using Denoising Diffusion Models**
   Link:
   https://www.researchgate.net/publication/370559381_Denoising_diffusion_probabilistic_models_for_3D_medical_image_generation