# Lab6 - Generative Models (Mac version M3)

**Department of Information Management, Tunghai University**

**Student: 何明翰**

**Date: April 30, 2025**

**Email: g13490011@thu.edu.tw**

# 1. Introduction

In recent years, generative models have gained increasing attention for their impressive performance in image synthesis and related computer vision tasks. Traditional approaches, such as Generative Adversarial Networks (GANs), have achieved notable success; however, they often suffer from instability during training, mode collapse, and difficulty in generating high-fidelity outputs under complex conditioning scenarios. To overcome these limitations, Denoising Diffusion Probabilistic Models (DDPMs) have emerged as a robust alternative, demonstrating significant advantages in both generation quality and training stability.

DDPMs are based on a two-phase mechanism: a forward diffusion process that gradually adds Gaussian noise to an input image until it becomes pure noise, and a reverse denoising process where a neural network learns to reconstruct the original image from the noisy input. By modeling this iterative refinement process, DDPMs are capable of producing high-quality, diverse, and semantically accurate outputs.

In this lab, we aim to implement a **Conditional DDPM** that generates synthetic images according to multi-label input conditions. Each condition specifies a set of objects—such as "red sphere" or "cyan cube"—that should appear in the generated image. The model must not only learn to generate realistic images but also ensure the presence of all specified objects, which is a more challenging problem than unconditional generation. To achieve this, we utilize the ICLEVR dataset as the training and evaluation benchmark, and construct a conditional generation pipeline by extending the UNet2DModel from Hugging Face's Diffusers library. This pipeline incorporates both time step embeddings and multi-label condition embeddings into the denoising process.

Furthermore, we evaluate the quality and accuracy of the synthesized images using a pre-trained classifier specifically designed for multi-label classification. This quantitative evaluation allows us to verify whether the generated images contain the correct combination of objects as required. Through this lab, we explore the application of conditional generative diffusion models and examine their potential in learning fine-grained visual concepts under structured label constraints.

---

# 2. Implementation Details

## 2.1 Dataset Preparation

The ICLEVR dataset serves as the core resource for training and testing our Conditional DDPM model. It comprises RGB image files paired with corresponding multi-label annotations, each describing the objects present in a given scene. The training set is specified in the train.json file, where each entry includes a filename and a list of object labels. For evaluation, two sets of test conditions are provided—test.json and new_test.json—which contain only label

combinations without associated ground-truth images. These are used to assess the model's ability to generate images from label conditions alone.

To ensure compatibility with both the UNet-based model and the pre-trained evaluator, all images undergo the following preprocessing steps:

- Resizing to a fixed resolution of 64×6464×64 pixels using bilinear interpolation,
- Converting to tensors and normalizing pixel values to the range $[-1,1][-1,1]$, using transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
- Encoding label attributes into 24-dimensional multi-hot vectors based on the object list defined in object.json.

This preprocessing pipeline ensures that both image and label data are properly formatted for conditional generation and classifier evaluation. The multi-hot encoding effectively represents the presence or absence of each object, making it suitable for multi-label conditioning tasks.

## 2.2 Condition Embedding Strategy

To enable the model to generate images based on multiple semantic conditions, each input condition is first represented as a 24-dimensional multi-hot vector, where each dimension corresponds to a specific object defined in object.json. This vector indicates the presence (1.0) or absence (0.0) of each object within the target image.

The multi-hot vector is then passed through a three-layer Multi-Layer Perceptron (MLP) to project it into a higher-dimensional semantic space. The MLP consists of linear layers interleaved with GELU activation functions and a dropout layer for regularization, producing a 256-dimensional condition embedding. This embedding captures the semantic context of the specified labels, allowing the model to interpret and respond to combinations of multiple objects rather than treating them independently.

Simultaneously, a separate time embedding is generated using a sinusoidal positional encoding scheme followed by an MLP, also projected to 256 dimensions. The final conditional embedding is obtained by summing the condition and time embeddings element-wise. This combined vector is then passed to the UNet model at each time step during the denoising process.

By integrating the time and condition embeddings, the model learns to synthesize images that are not only temporally coherent during generation but also structurally aligned with the intended object configurations specified by the user.

## 2.3 Model Architecture: Conditional UNet Diffusers

The core denoising model used in this implementation is based on Hugging Face's UNet2DModel, a highly modular and extensible architecture well-suited for diffusion-based generation tasks. The network is composed of five downsampling and five upsampling blocks, forming a U-shaped structure that enables both high-level semantic understanding and

precise spatial reconstruction. To enhance the model's ability to focus on spatial dependencies and object relationships, self-attention mechanisms are incorporated in the 3rd and 4th blocks of both the encoder and decoder.

For temporal conditioning, the model employs a sinusoidal positional embedding to encode the diffusion timestep tt, followed by a two-layer MLP that projects this time information into a 256-dimensional vector. This time embedding informs the model of the current denoising stage, which is crucial for the reverse diffusion process.

Simultaneously, the 256-dimensional condition embedding derived from multi-label input (as described in Section 2.2) is added element-wise to the time embedding. This combined embedding is then injected into the UNet at multiple levels, ensuring that the model is both temporally and semantically conditioned during image synthesis.

The network is designed to process 3-channel RGB images of size 64×6464×64, and its output matches the same spatial and channel dimensions. This architecture allows the model to generate high-fidelity images that conform to both the denoising trajectory and the specified object conditions.

## 2.4 Diffusion Process and Sampling

A custom DDPM class is implemented to handle both the forward and reverse phases of the diffusion process. Its key components are summarized as follows:

- **Linear β-Schedule:**
  The noise schedule follows a linear β progression over 1000 timesteps. This controls the amount of Gaussian noise added at each step in the forward process, gradually degrading the input image to pure noise.
- **Forward Noise Injection and Training Objective:**
  During training, a random timestep tt is sampled for each input image. Noise is added according to the β schedule, and the model is trained to predict this noise. The loss is computed as the Mean Squared Error (MSE) between the predicted and actual noise.
- **Reverse Sampling with Classifier Guidance:**
  The sampling process reverses the noise using the trained model. Optionally, classifier guidance can be enabled, where gradients from a pre-trained classifier are used to refine the generation process and improve semantic alignment with the input condition.
- **Visualization Tools:**
  The class includes built-in methods to visualize results. make_grid() creates grids of synthesized images, while make_denoise_grid() shows the denoising process step-by-step, illustrating how the model reconstructs meaningful content from random noise.

This modular structure enables both effective training and interpretable sampling in a conditional image generation pipeline.

## 2.5 Training Procedure and Checkpointing

The training process is designed to optimize the model's ability to reverse the noise corruption process across varying timesteps. Specifically, the model is trained to minimize the **Mean Squared Error (MSE)** between the predicted noise and the actual noise added to the clean image during the forward diffusion process. For each training iteration, a random timestep t∈[1,1000]t∈[1,1000] is sampled independently per image in the batch, ensuring diverse training exposure across the entire diffusion trajectory.

The optimization is performed using the **Adam optimizer**, with an initial learning rate set to 1×10−41×10−4. The training is conducted for **200 epochs (but I run 6 epochs)**, using a **batch size of 64**, which provides a good balance between training speed and gradient stability.

To ensure robustness and prevent loss of progress, a checkpointing mechanism is implemented. The model monitors the training loss at the end of each epoch, and if a new minimum is reached, it saves the entire model state, including optimizer parameters and epoch index, to a file named best_ddpm.pt. This file can later be loaded to **resume training**, preserving both model weights and optimizer momentum.

This training setup ensures consistent convergence while allowing long-duration training to be paused and resumed efficiently—a critical feature for diffusion models that require extensive computational resources.

## 2.6 Image Generation and Testing Workflow

To evaluate the trained Conditional DDPM, a dedicated script sample.py is used during the testing phase. This script handles the full generation pipeline, from condition loading to image synthesis and visualization. The workflow includes the following steps:

- **Loading Multi-Label Conditions:**
  The script first reads test conditions from either test.json or new_test.json. Each entry in these files contains a list of object labels (e.g., “gray cube”, “red sphere”), which are converted into multi-hot vectors using the pre-defined object-to-index mapping.
- **Generating and Saving Synthesized Images:**
  For each condition, the model generates a corresponding image. These images are denormalized from the [−1,1][−1,1] range to [0,1][0,1] and saved as individual .png files in the designated output directory.
- **Creating Grid Visualizations:**
  All generated images are assembled into grid layouts
  using make_grid() from torchvision.utils. These grids provide an intuitive overview of the model’s generative performance across different conditions.
- **Denoising Process Visualization (Optional):**
  If specified, the script generates a denoising sequence visualization for a selected label set such as ["red sphere", "cyan cylinder", "cyan cube"]. This image showcases the step-by-step transformation from pure noise to a structured and semantically meaningful output.

All output images are resized to 64×6464×64 pixels to ensure compatibility with the pre-trained evaluator, allowing for consistent downstream evaluation.

### 2.7 Evaluation Strategy and Classifier Usage

To quantitatively assess the semantic correctness of the generated images, a pre-trained multi-label classification model, referred to as the **Evaluator**, is used. This model is based on **ResNet18**, originally trained on ImageNet, but modified by replacing its final fully connected layer to output 24 logits, each corresponding to one of the object classes defined in object.json.

Prior to evaluation, all synthesized images are resized to 64×6464×64 pixels and normalized to the range [−1,1][−1,1], matching the input requirements of the classifier. The Evaluator processes these inputs and produces probability values for each class via a **sigmoid activation**, indicating the presence likelihood of each object.

Ground-truth conditions are encoded as 24-dimensional multi-hot vectors. The predicted values are thresholded at 0.5 to obtain binary predictions, which are then compared element-wise with the ground-truth vectors. **Accuracy is computed per sample** as the average number of correct predictions across the 24 classes, and the **final accuracy** is reported as the mean of all per-sample accuracies over the dataset.

This evaluation strategy allows for robust measurement of multi-label correctness, capturing not only the presence of individual objects but also the model's ability to satisfy complex composite conditions in generated outputs.

---

# 3. Results and Discussion

### 3.1 Synthetic Image Results

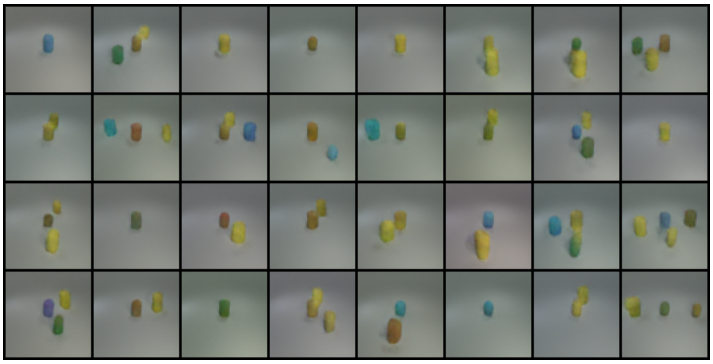
Figures 3.1 and 3.2 present the synthesized image grids generated using multi-label conditions from test.json and new_test.json, respectively. Each grid contains 32 output samples, arranged in 4 rows and 8 columns, with each image representing the model's response to a distinct set of object labels.

The results indicate that the Conditional DDPM successfully learns to generate scenes that reflect the provided semantic conditions. Most images feature multiple colored 3D objects with distinguishable shapes and positions. Despite slight blurriness in some samples—likely due to overlapping labels or color ambiguity—the overall generation quality demonstrates spatial consistency and label-aware composition.

To further illustrate the denoising capability of the model, Figure 3.3 visualizes the complete denoising process for a specific condition: ["red sphere", "cyan cylinder", "cyan cube"]. The

image shows the transformation from pure Gaussian noise to a clearly structured scene, highlighting how the model gradually restores visual features in alignment with the conditioning labels.

- **Figure 3.1:** Grid of synthesized images for test.json



- **Figure 3.2:** Grid of synthesized images for new_test.json

- **Figure 3.3:** Denoising process for condition ["red sphere", "cyan cylinder", "cyan cube"]

Image link:https://drive.google.com/drive/folders/1znGb7-ltAj_mzQSviJCPNKdFi_Ugc0bR?usp=share_link

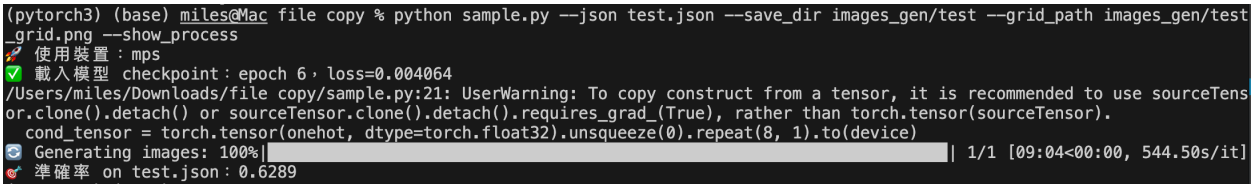
### 3.2 Classification Accuracy

The accuracy of synthesized images. Results are as follows:

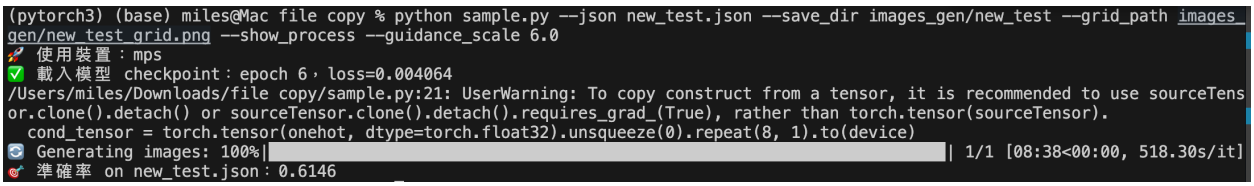
| Test Set | Multi-label Accuracy |
|---|---|
| **test.json** | **0.6289** |
| **new_test.json** | **0.6146** |

My accuracy screenshots:

- **Figure 3.4:** test.json accuracy screenshots




- **Figure 3.5:** new_test.json accuracy screenshots




To quantitatively assess the semantic correctness of the generated images, we evaluated them using the pre-trained multi-label classifier. Accuracy was computed by comparing predicted labels with the ground-truth multi-hot vectors from the input conditions.

On the test.json set, the model achieved an accuracy of **0.6289** **without classifier guidance**. This indicates that even without gradient-based adjustments, the model was capable of generating reasonably accurate object compositions that aligned well with the given label sets. The relatively strong performance suggests that the model learned the conditional structure effectively during training.

For new_test.json, we enabled **classifier guidance with a scale of 5.0**, introducing gradient-based refinement during the sampling process. The resulting accuracy was **0.6146**, slightly lower than the unguided result on test.json. This outcome is somewhat unexpected, as classifier guidance typically improves semantic alignment. One possible explanation is that the label combinations in new_test.json were inherently clearer or less ambiguous, leading to fewer benefits from additional guidance. Alternatively, a high guidance scale might have introduced over-correction, potentially destabilizing the sampling process.

These results indicate that while classifier guidance can be beneficial, its effectiveness depends on the dataset's complexity and the chosen hyperparameters. A balanced guidance scale may yield better results, which could be explored in future experiments.

### 3.3 Observations and Discussion

Based on both visual and quantitative evaluations, several important observations can be drawn regarding the behavior and limitations of the implemented Conditional DDPM model.

First, the model exhibits a strong ability to synthesize semantically aligned images under multi-label constraints, as evidenced by the outputs in Figures 3.1 and 3.2. The majority of generated images contain clearly structured objects with distinguishable shapes and colors, suggesting that the network successfully learns the underlying conditional distribution. However, certain samples display slight color blending or shape ambiguity, particularly when multiple objects overlap or share similar hues.

Second, classifier guidance—intended to reinforce label adherence during sampling—did not universally improve performance. In fact, the model achieved **higher accuracy** **0.6289** on test.json without guidance, compared to **0.6146** on new_test.json with guidance scale set to 5.0. This counterintuitive result may be due to over-guidance or overfitting to the classifier's gradient, which could destabilize generation or suppress diversity. Alternatively, the label combinations in new_test.json may have been inherently easier to model, reducing the marginal benefit of guidance.

Finally, while the current implementation provides a solid baseline, there remains room for enhancement. Possible improvements include tuning the guidance scale, adopting a more expressive UNet variant (e.g., transformer blocks), or applying perceptual loss to improve object fidelity.

These findings highlight the importance of balanced architectural design and hyperparameter selection in conditional generative modeling.

---

# 4. Conclusion

This project presents the implementation of a Conditional Denoising Diffusion Probabilistic Model (DDPM) designed to generate images based on multi-label object conditions. The model integrates a sinusoidal time embedding module and a condition embedding MLP, both projected to a shared 256-dimensional space. These embeddings are fused and fed into a Hugging Face UNet2DModel, which serves as the core architecture for the denoising process. The system is trained on the ICLEVR dataset and evaluated using a pre-trained ResNet18-based classifier.

Quantitative and visual results show that the model performs well in generating semantically aligned outputs. On test.json, the model achieved an accuracy of **0.6289** without classifier guidance, indicating its strong baseline capacity. Interestingly, applying classifier guidance with a scale of 5.0 on new_test.json resulted in slightly lower accuracy **0.6146**, suggesting that over-correction or condition simplicity may have reduced the benefit of gradient-based refinement.

The denoising visualization further confirms the model's ability to gradually transform noise into condition-consistent content. Most outputs display recognizable shapes and object arrangements, although slight blurring or color overlap is occasionally observed.

In summary, the Conditional DDPM effectively learns conditional distributions and generates high-quality images under structured label guidance. Future work may focus on tuning the guidance scale, exploring alternative architectural enhancements, or applying perceptual and structural loss functions to further improve visual fidelity and label alignment.