Report For Project3

Yuqi Pan

November 13, 2025

1 Introduction

By completing this assignment, I gradually learned and explored the generative capabilities of diffusion models, progressing from basic architecture to advanced multimodal applications. In the basic task, I systematically experimented and analyzed variational autoencoders (VAE), denoising diffusion probabilistic models (DDPM), denoising diffusion implicit models (DDIM), and latent space diffusion pipelines on the MNIST dataset. Experiments revealed a key trade-off between reconstruction fidelity and latent space regularity in VAEs, demonstrated the computational efficiency advantage of DDIM over DDPM, and validated the effectiveness of latent space compression for diffusion modeling. In the advanced task, I completed a simplified text-to-image model (tiny-mnist-sd), achieving conditional generation with label alignment accuracy around 60%. Furthermore, I conducted ComfyUI workflow experiments, constructing different workflows and comparing them with the tiny-mnist-sd pipeline, discovering fundamental similarities in workflow architecture and differences in flexibility and component modularity.

2 VAE: Reconstruction and Latent Encoding

2.1 Basic Theory

Reconstruction Loss:The reconstruction loss measures the fidelity of the decoded output compared to the original input. It compares the differences between the original and reconstructed images at each pixel. For binary images like MNIST, we typically use the binary cross-entropy loss, as shown in Equation 1. This term ensures that the encoded information can be faithfully reconstructed, serving as an autoencoder constraint.

$$L_{recon} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] \tag{1}$$

KL Divergence Loss:The KL divergence loss normalizes the latent space by forcing a similarity between the learned posterior distribution $q(z|x)$ and the prior distribution $p(z)$ (typically a standard normal distribution). This encourages a well-structured and continuous latent space, enabling meaningful sampling and interpolation. The complete VAE loss is a weighted combination, as shown in Equation 2. Here, β controls the tradeoff between reconstruction quality and latent space regularity.

$$L_{VAE} = L_{\text{recon}} + \beta \cdot L_{KL} \tag{2}$$

2.2 VAE Experimental Results and Analysis

I conducted experiments using β values of 0.001, 0.01, 0.1, 1, 5 and 10 to systematically analyze the trade-off between reconstruction fidelity and latent spatial regularity of VAE under different β values. In this comparative experiment, all parameters except the β parameter were controlled to the same value. The specific parameter settings are shown in Table 1.

Table 2 shows a comparison of VAE performance metrics under different β values, with the results from the last 40 epochs selected for each scenario. From the table, we can see that for low β values (between 0.001 and 0.1), the KL divergence is extremely high, the reconstruction loss is relatively low, the reconstruction quality is excellent but the sample generation quality is extremely poor, the latent space lacks structure, and it cannot support meaningful sampling. For medium β values, i.e.,

Table 1: VAE training parameter configuration

Parameter	epochs	batch-size	lr	latent-dim	kl-weight	loss-type
Value	40	128	2×10^{-4}	100	1	MSE

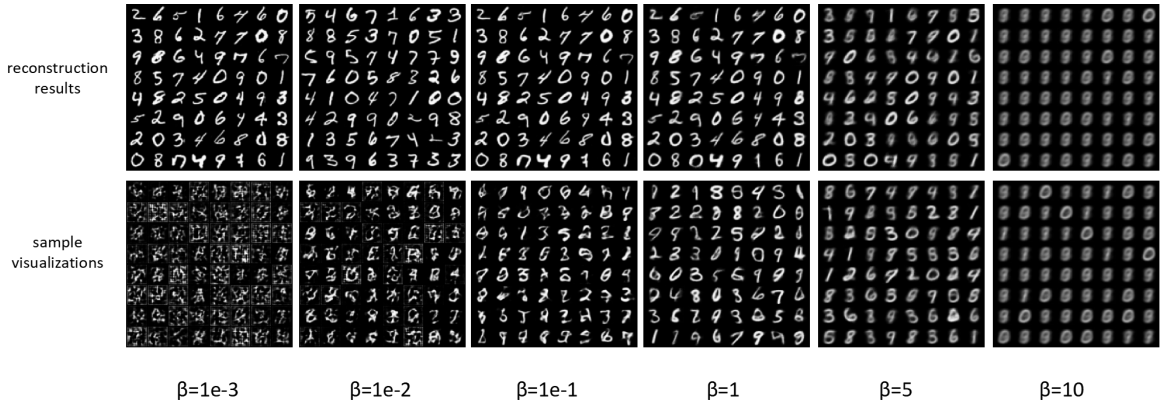


Figure 1: Visualized sample plots based on different β

when the β value is 1, the KL divergence and reconstruction loss reach a relative balance, the sample quality is good, and the reconstruction quality is relatively good based on the results. For high β values (between 5 and 10), the KL divergence drops sharply to near zero, and the reconstruction loss increases significantly. This reveals a clear trade-off: β is strongly negatively correlated with KL divergence, increasing β leads to a rapid decrease in KL divergence; while β is positively correlated with reconstruction loss, increasing β leads to an increase in reconstruction loss. Figure 1 shows the reconstruction results and sample visualizations obtained under different β values. We can clearly see that as the β value increases, the reconstruction image gradually deteriorates and becomes blurry, eventually becoming completely indistinguishable. The sample image, on the other hand, initially becomes clearer, then deteriorates again after β=1. This result demonstrates that focusing too much on reconstruction loss or sacrificing reconstruction accuracy for regularization will not yield good results; only a balanced parameter selection can achieve satisfactory outcomes.

Table 2: Impact of different β values on VAE performance

β	Total Loss	Recon Loss	KL Loss	Sample Quality	Recon Quality
0.001	0.83	0.53	298.14	Poor, unrecognizable	Good
0.01	2.33	1.03	130.01	Poor, unrecognizable	Good
0.1	7.95	3.48	44.76	Fair, few recognizable	Good
1	26.41	13.29	13.12	Good, mostly recognizable	Fair, slight ghosting
5	51.63	38.29	2.67	Fair, ghosting	Poor, ghosting
10	57.65	55.17	0.25	Poor, unrecognizable	Poor, unrecognizable

Based on the above analysis of the results, we draw a clear conclusion: there is a fundamental trade-off between reconstruction accuracy and latent spatial regularity, and the β parameter in VAE is essentially the regulator of that trade-off. In fact, there always exists an optimal range of β values that achieves the best balance between reconstruction fidelity and latent spatial interpretability. For VAE applications on the MNIST dataset, $\beta = 1$ has proven to be the most suitable choice.

Furthermore, by observing the changes in loss during training, it was found that the loss consistently showed a downward trend. It was speculated that the model might not have converged at 40 epochs, and could potentially be trained better. Therefore, the number of epochs was increased to 200, and training was re-run. The results showed that the loss did indeed decrease, but only slightly, from 26.41 to 24.46, and the difference was barely noticeable in the generated sample images (see Figure 2). From this result, it can be inferred that VAE training converges relatively quickly, and increasing the

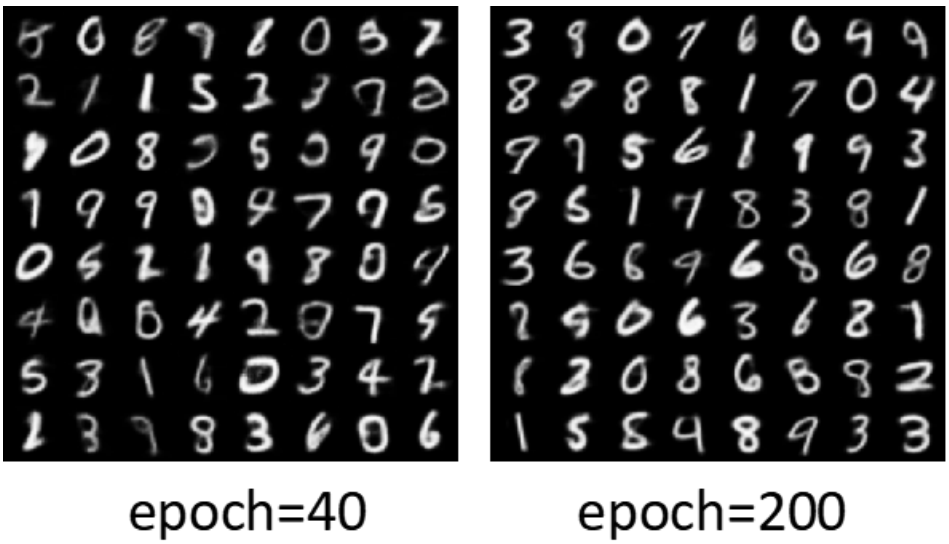


Figure 2: Comparison of sample images from different epochs

number of epochs has limited effect on improving model performance.

I also explored the impact of latent space dimensionality on VAE performance by modifying the value of latent-dim. In this experiment, I controlled the variables, setting kl to 1, epoch to 40, and selecting latent-dim values of 10, 50, 100, and 500 for the experiment. The final results are shown in Table 3. We can see that when the latent space is too small, the reconstruction error increases, and the kl divergence decreases. This is mainly because the latent space dimension is too low, forcing the encoder to compress the input image into a space with only two variables, which leads to information loss and a decrease in the quality of the reconstructed image.

Table 3: Impact of different latent dimensions on VAE performance

Latent Dim	Total Loss	Recon Loss	KL Loss
2	36.64	31.43	5.21
50	26.33	13.15	13.18
100	26.41	13.29	13.12
500	26.90	13.96	12.94

2.3 DDPM and DDIM: Accelerating Generative Synthesis

Based on the theoretical framework in the paper, I implemented the DDIM sampling algorithm, which significantly improves the sampling efficiency of the diffusion model through a non-Markov process. In this task, I set the training parameters as shown in Table 4. The final sampling results of DDPM and DDIM are shown in Figure 3. From the image, the sampling results of DDIM and DDPM appear similar. However, upon closer inspection of each digit, the number of identifiable digits in DDIM is greater than that in DDPM. Except for a slightly lower recognition rate in DDIM, the digit ghosting and distortion are similar for both methods. The main breakthrough of DDIM lies in breaking the Markov chain constraint of traditional DDPM, allowing skipping intermediate time steps, using subsequences for sampling, and controlling the degree of randomness through the eta parameter, thus significantly reducing the number of forward passes. Even though the final result of DDIM is slightly worse, its sampling speed is many times faster than that of DDPM.

From the sampling formula of DDIM (Equation (5)):

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\epsilon_\theta(x_t, t) + \sigma_t\epsilon \tag{3}$$

Figure 3: Visualizing diffusion results

we can see that DDIM uses a non-Markovian process to skip unnecessary intermediate steps. Furthermore, DDIM achieves a balance between randomness and efficiency by adjusting the parameter η to control σ_t (Equation (4) in the paper):

$$\sigma_t = \eta \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\left(1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}\right)} \tag{4}$$

achieving an order-of-magnitude efficiency improvement while maintaining generation quality.

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\epsilon_\theta(x_t,t) + \sigma_t\epsilon \tag{5}$$

Table 4: Diffusion model training parameter configuration

Parameter	*batch_size*	epochs	lr	timesteps	*ddim_steps*	*ddim_eta*
Value	128	60	1×10^{-4}	1000	200	0.0

2.4 Latent Space Diffusion: Enhancing Efficiency

When a VAE is used as a compressor for latent diffusion, its main function shifts from "generative model" to "efficient encoder." In this case, priority should be given to optimizing reconstruction capability rather than latent space regularization. In the experiments in section 2.1, β=1 achieved a balance between latent space regularity and image regularity. For this task, I pre-selected the model with β=1 and latent-dim=50 from task 2.1 as the model for the latent diffusion stage for diffusion training. The visualization results of the sampling can be seen in Figure 4. Overall, the samples generated by the DDPM and DDIM methods obtained by using this model for diffusion are of good quality; 85% of the numbers in the figure can be clearly identified.

To verify the suitability of my chosen model, I used the models trained with latent-dim=100 and different β values from task 2.1 as models for the latent diffusion stage, and repeated the diffusion training. The sample results are shown in Figure 5. We can see that the models with β values of 0.5 and 1 generated the best sample quality. As the β value decreases from 0.5 to even smaller values,

Figure 4: Based on diffusion results of β=1-latend-dim=50

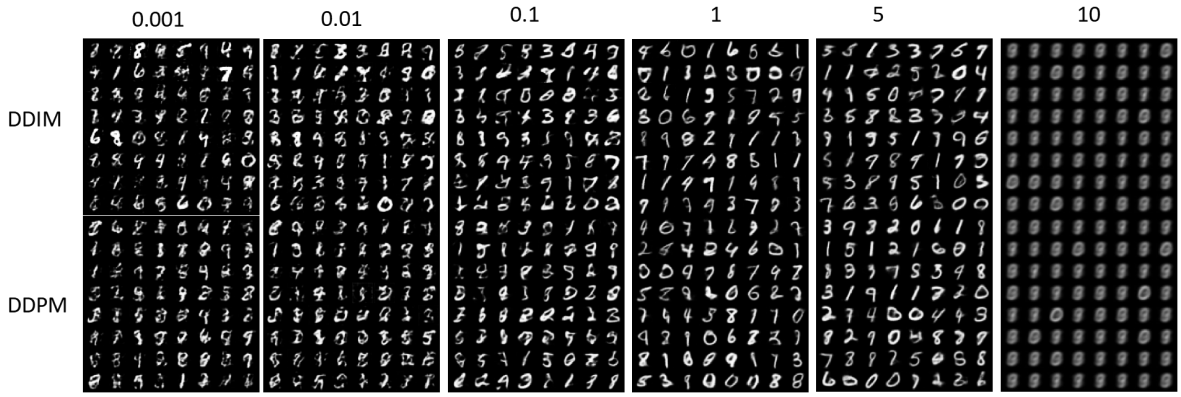


Figure 5: Visualization results of diffusion based on different β

the samples gradually generate a lot of noise, and the numbers cannot be distinguished due to low regularity. When the model's β value is too high (β=10), the sample numbers become blurry and distorted. In addition, I controlled the β value to 1 and used VAE models with different latent-dim values for validation experiments. The results are shown in Figure 6. It can be seen that the sampling effect is best when latent-dim is 50. The dimension of the latent space determines how much information the latent space can encode. This value is too low (latent-dim=2) and will lead to blurry information reconstruction, while it is too high (latent-dim=500) and will lead to overfitting.

The above results verify that selecting a VAE parameter with a β value of 1 and an latent-dim value of 50 performs well in both theoretical analysis and experimental verification, and successfully constructs a high-quality latent space suitable for diffusion models.

3 Advanced Applications: Controllable Generation

In the advanced task, I first ran '*train_clip.py*' to obtain a CLIP model trained on MNIST images and their corresponding labels. Then, I injected CLIP conditions into the logic of EpsMLPCond to implement and complete the training of a Tiny Multimodal SD model. Furthermore, I downloaded and installed the ComfyUI software, used it to implement a complex generation workflow, completed multiple text-to-image tasks, and analyzed the experimental results.

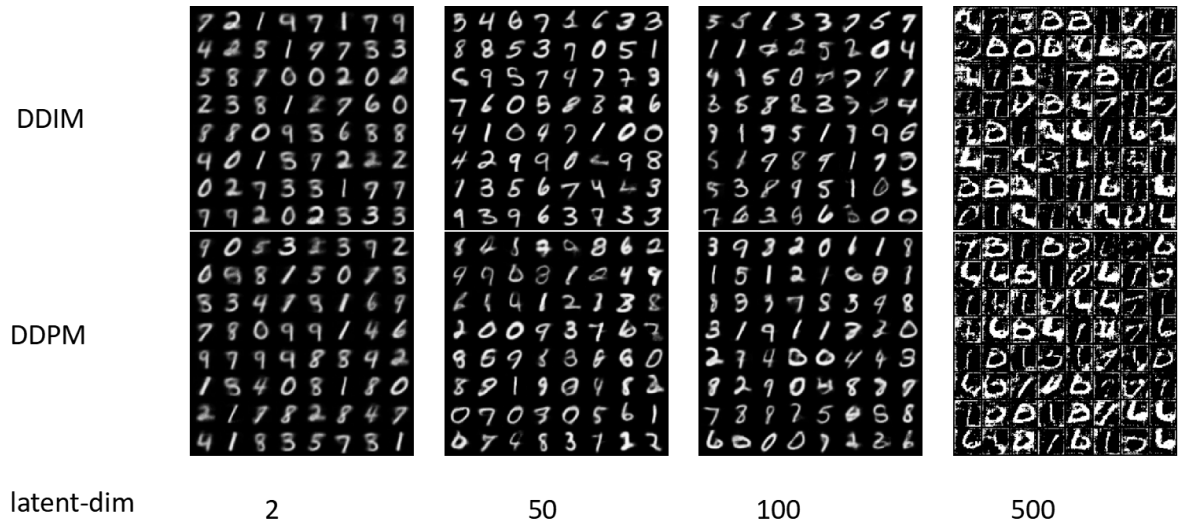


Figure 6: Visualization results of diffusion based on different latend-dim

Figure 7: Visualization results of tiny multimodal SD

3.1 Conditional Generation: Training a Tiny Multimodal SD

In the Tiny Multimodal SD training task, I completed the key code and performed training, with the training parameters set as shown in Table 5. The final controlled generated digit grid is shown in Figure 7. It can be observed that nearly 60% of the digit samples can be correctly generated based on the adjusted labels. The figure also shows that the success rate for digit samples 5, 8, and 9 is relatively low.

Table 5: Diffusion model training parameter configuration

Parameter	embed_dim	epochs	lr	timesteps	*ddim_steps*	*ddim_eta*
Value	128	70	5×10^{-5}	1000	50	0.0

3.2 ComfyUI: A Glimpse into Diffusion Workflows

In this task, I used v1-5-pruned and the Stable Diffusion 1.0 base model as the test model and deployed two different workflows to test the model's text-to-image performance by providing prompts with varying levels of detail. The first workflow uses the basic process, as shown in Figure 8, which includes

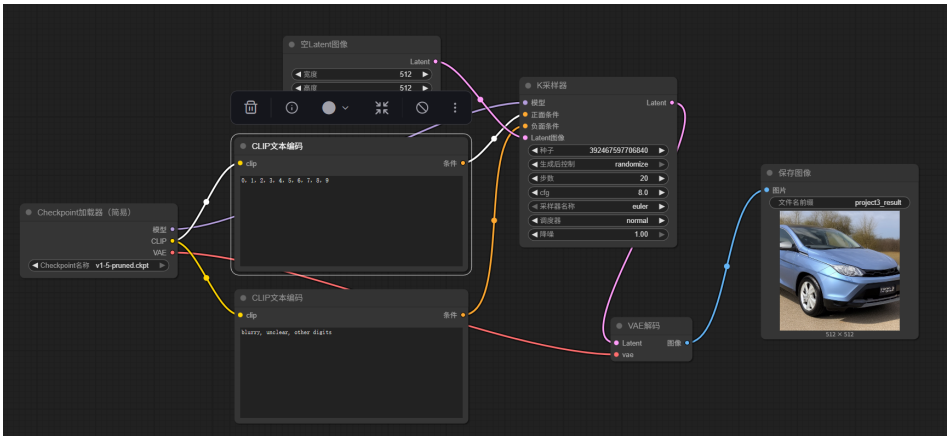


Figure 8: first workflow of ComfyUI

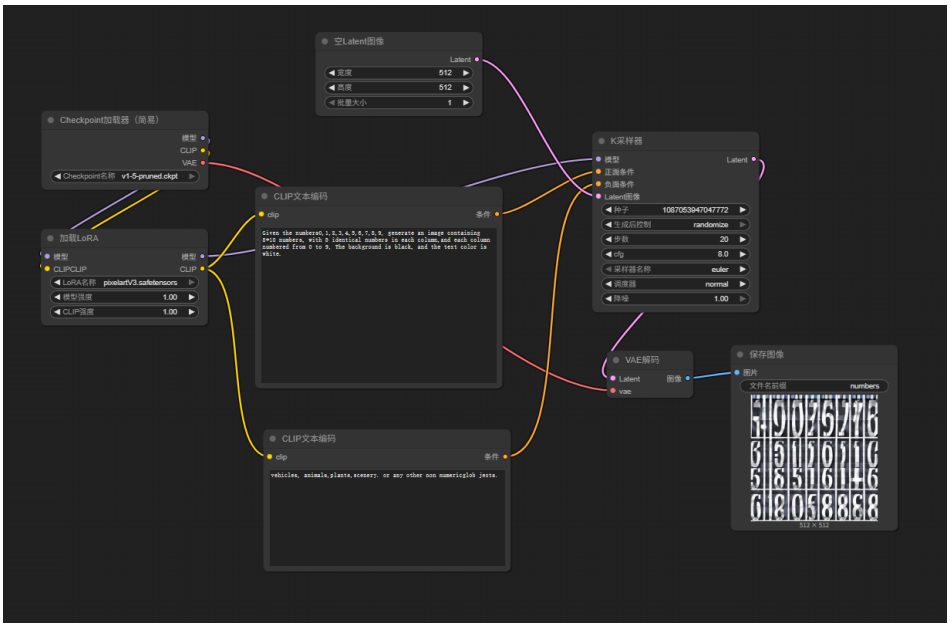


Figure 9: second workflow of ComfyUI

components such as text prompts, a CLIP text encoder, a K-sampler, a VAE decoder, and the output image. The second workflow adds a LoRa model to the first workflow, as shown in Figure 9.

Based on the established workflow, I conducted a cue word experiment, as shown in Figure 10. When I provided a simple positive cue word "0,1,2,3,4,5,6,7,8,9" in the basic workflow, and set the negative cue words to "blurry, unclear, other digits", the result was an image of a car. When I adjusted the cue words and added constraint details, setting the positive cue word to "Given the numbers 0,1,2,3,4,5,6,7,8,9, generate an image containing 8*10 numbers, with 8 identical numbers in each column, and each column numbered from 0 to 10. The background is black, and the text color is white.", and the negative cue words to "vehicles,objects,animals,plants,scenery,or any other non-numerical", the result changed significantly. The image generated numbers, but it did not present them as I intended. When I ran the same detailed prompts using a workflow with a LoRa model, I got different results. This time, the number arrangement in the generated image matched my prompt requirements, but the content still didn't fit; it didn't present rows of identical numbers. Overall, all three attempts failed, but the third attempt was closer to what I wanted. The poor performance of the generated digit images is mainly because ComfyUI is based on natural datasets. It has strong recognition capabilities for common objects like animals and vehicles, but the number of digit samples is relatively small and homogeneous, resulting in a weaker ability for the model to recognize numbers.

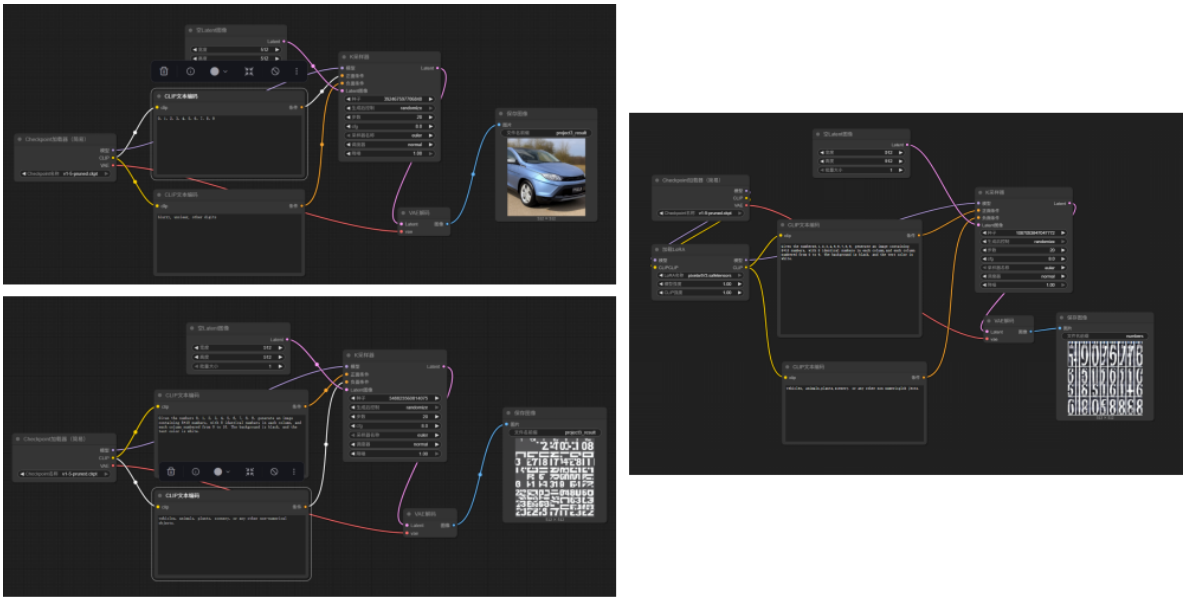


Figure 10: Comparison of comfyUI text-to-image effects under different prompts(Unsuccessful cases)

When no negative prompts related to cars are set, the model might be affected by the prevalence of car samples in the natural dataset. Using more specific prompts can avoid this. However, the latter two attempts using detailed and specific prompts still didn't achieve the desired results, perhaps because the number of digit samples in the natural dataset was too small, leading to low model recognition. To address this, I changed the object to a puppy and obtained better results, as shown in Figure 11.

Next, based on the second workflow, I used three different LoRa styles—Pixel Art style LoRa, Detail LoRa, and 3D Style LoRa—to conduct LoRa style verification experiments. Figure 12 below shows my visualization results. It can be seen that by combining different LoRa styles with clear prompts, images that meet the requirements and are rich in detail can be generated from text.

After using ComfyUI, looking back at our previous $tiny-mnist-sd$ code, we see many similarities, but the user experience is completely different. From a core architecture perspective, both follow similar logical chains: first, text or label information is CLIP-encoded into conditional vectors; then, a diffusion model is used step-by-step to denoise the latent space; finally, a VAE decoder converts the result into an image. However, in actual use, the difference is stark. Our $tiny-mnist-sd$ is like a sealed black box—conditional labels are inserted, images are output, and what happens in between is entirely up to imagination. The parameters set during training are the final result; fine-tuning requires retraining the entire model. ComfyUI, on the other hand, breaks down the entire generation process into visual nodes, clearly displaying each step. Sampling steps, CFG weights, LoRA model—all parameters can be adjusted in real time, and this can be done simply by dragging and dropping graphical components, making it significantly more convenient. The biggest advantage of $tiny-mnist-sd$ is its transparent code and comprehensive debugging capabilities, which allows researchers more freedom to design or modify algorithms for research.

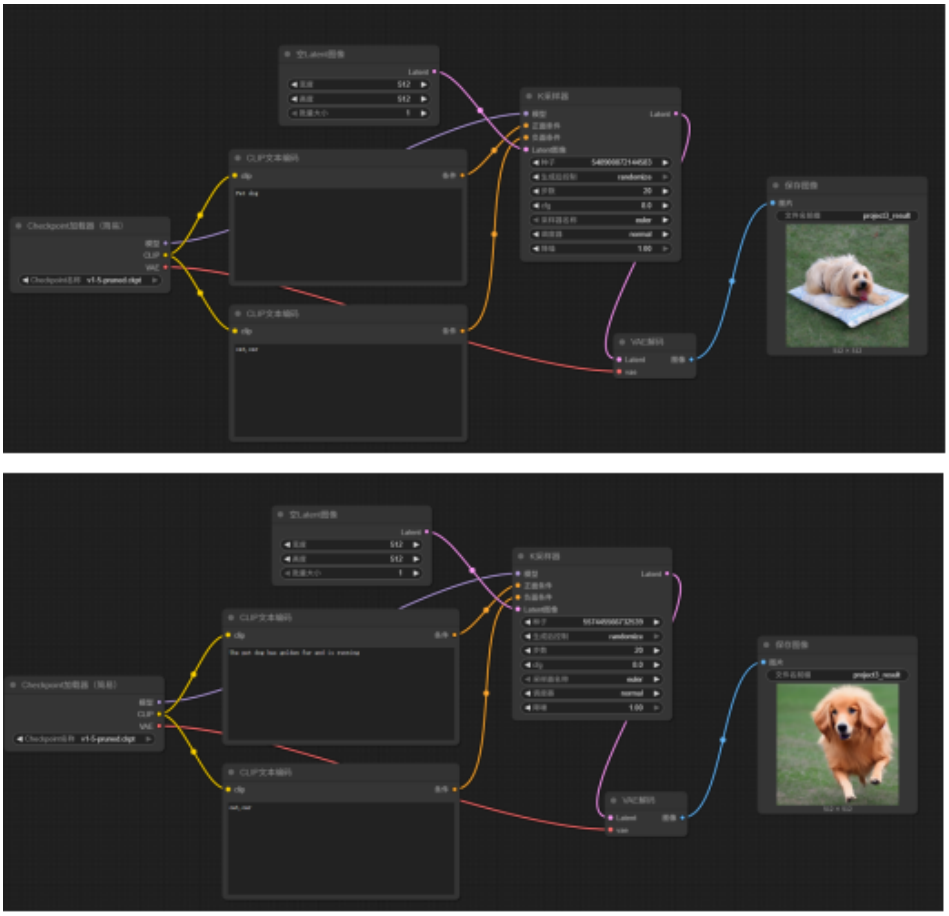


Figure 11: Comparison of comfyUI text-to-image effects under different prompts(Successful cases

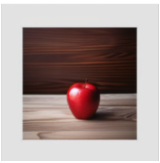
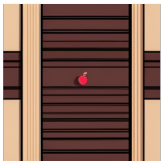
positive promt	nagative promt	no lora	Pixel Art style LoRA	Detail LoRA	3D Style LoRA
a futuristie robot tendins to a vibrant flover garden, intricatemechanical details,sunny day, cinematic lighting, highly detailed, 8kresolution	blurry, low quality. deformed, mutated, usly, bad anatony, extra limbs, missing limbs, poorly drawn				
a peaceful lakeside cabin at sunset, cozy, invitins atnosphere	photorealistic, blurry,low quality, deformed, usly				
a single red apple on a wooden table	multiple apples, rotten, blurry, unrealistic				

Figure 12: Image renderings based on different LoRa models