ControlNet: Adding Edge Conditioning for Enhanced Control with Unconditional Denoising Diffusion Probabilistic Models

Anonymous submission

Abstract

ControlNet, introduced by Zhang, Rao, and Agrawala in 2023, (Zhang, Rao, and Agrawala 2023) is essentially a neural network structure that is used to control diffusion models by adding extra conditions and offers a significant innovation in the world of text-to-image diffusion models. While models like 'Stable Diffusion' developed by Rombach et al. in 2022 (Rombach et al. 2022) have transformed visual content creation by generating images from text, they've often struggled with giving users fine-grained spatial control. ControlNet solves this issue by enabling users to condition the image generation process with additional inputs, such as edge maps, depth maps, segmentation maps, or even human pose skeletons. This makes it easier to guide the diffusion model to create specific, detailed outputs while still preserving the core abilities of the underlying model. In this project, ControlNet is demonstrated using Canny edge control on a vanilla 'unconditional' Denoising Diffusion Probabilistic Model (DDPM) as a proof of concept using limited compute. The MNIST dataset, known for its handwritten digit images, is used as a robust benchmark to evaluate ControlNet's capabilities in conditional image manipulation with Canny edges. By leveraging diffusion models trained using this dataset and incorporating conditional control inputs, ControlNet enables more precise conditioning of outputs based on the input data.

Code — https://anonymous.4open.science/r/ControlNet-ECE570-D5C5

Introduction

One of the key strengths of ControlNet is its ability to balance stability and flexibility. It introduces new conditioning layers to the model without disturbing the large diffusion model's generalization capabilities. This allows for efficient training and adaptation without overfitting or forgetting previously learned information, making it a powerful tool even when working with smaller datasets. The system is remarkably scalable and competitive with much larger models, yet it manages this without significantly increasing computational costs. Before we dive deeper into how ControlNet works, lets fitst discuss about Stable Diffusion, which serves as the foundation for ControlNet. Stable Diffusion is a deep learning model that excels at generating high-quality images from text prompts by using a stable training process. Its ability to transform detailed or abstract descriptions into realistic visuals has made it a versatile tool for everything from digital art and scientific research to video game development. For instance, game developers can quickly create in-game characters or environments from simple text descriptions, while ecommerce platforms can use the technology to design products based on written inputs.

ControlNet builds on this by locking the weights of large, pretrained text-to-image models like Stable Diffusion and adding "zero convolution" layers. These layers progressively learn new parameters in a way that avoids introducing harmful noise into the model. As a result, ControlNet can be fine-tuned with new data without losing the stability and generalization capabilities of the original model. It's particularly effective in tasks like depth-to-image and pose-to-image generation, where users can combine multiple conditioning inputs alongside or without text prompts. This flexibility opens up new possibilities for text-to-image models across a range of fields, allowing for more precise control of the image generation process while maintaining the high quality and diversity of the outputs. Here's a refined version of your paragraph for improved clarity and coherence:

In this work, we first train a Denoising Diffusion Probabilistic Model (DDPM) using the MNIST dataset. The methodology then involves generating Canny edge maps from the original MNIST digit images, which are used as input conditions for ControlNet. By leveraging these edge maps, ControlNet can reconstruct and generate digit images that align with the structural patterns defined by the edges while preserving the core characteristics of the original dataset. This approach introduces a novel method for dataset augmentation, producing new variations of handwritten digits that adhere to predefined structural constraints.

Related Work

In recent years, numerous strategies have been developed to fine-tune neural networks for various applications, including natural language processing (NLP) and computer vision (CV). Some notable examples include:

- HyperNetwork- Introduced in 2016 (Ha, Dai, and Le 2016), HyperNetwork aims to train a smaller recurrent neural network that influences the weights of a larger one. This approach has been used in applications like generative adversarial networks (GANs) and diffusion models.
- Adapter Methods- In both NLP and computer vision,

adapters have become a widely used technique for customizing pretrained models. Adapter methods add lightweight module layers that allow models to adapt to new tasks. For text-to-image models, adapters can integrate external conditioning inputs, such as edge or depth maps, enabling more controlled image outputs. In the field of NLP, adapters are useful for incremental learning (adapting to new classes without forgetting) and domain adaptation (transferring models to new domains). Notable works in this space include the CLIP model by Radford et al. (2021) from OpenAI (Radford et al. 2021), which combines visual and text data for improved image classification. Another example is T2I-Adapter, developed by Mou et al. in 2024 (Mou et al. 2024), which extends Stable Diffusion's capabilities.

- Additive learning- This addresses a common challenge known as catastrophic forgetting, which occurs when fine-tuning large models on new data. This issue is managed by freezing the original model's weights and introducing a small set of new parameters using techniques such as learned weight masks, pruning, or hard attention (Rosenfeld and Tsotsos 2018) (Rosenfeld and Tsotsos 2018). These methods allow the model to retain its previously learned knowledge while focusing on learning new tasks.

- Low-Rank Adaptation (LoRA)- This is another new importanrt technique. LoRA prevents catastrophic forgetting by focusing on learning the offset of parameters through low-rank matrices (Hu et al. 2021) (Hu et al. 2021). This method uses the idea that many large, overparameterized models, such as transformers, can be expressed in a lower-dimensional subspace. By targeting only the low-rank component, LoRA reduces the number of parameters required for fine-tuning, enabling more efficient adaptation to new tasks with minimal memory and less computational needs.

- Zero-Initiated Layers- This is another approach, though initializing weights with zeros is typically avoided because it can hamper the learning process. However, in ControlNet, Zhang et al. make use of zero convolution layers, where the weights are initially set to zero and gradually grow during training. This allows ControlNet to fine-tune large pretrained models like Latent Stable Diffusion without introducing disruptive noise early in the process.

In summary, techniques like adapter methods, additive learning, side-tuning, LoRA, and zero-initialized layers all contribute to making large pretrained models more adaptable across both natural language processing and computer vision world. ControlNet's architecture takes advantage of these advances, especially in how it manages spatial conditioning in text-to-image diffusion models like Stable Diffusion. These strategies provide flexible, efficient, and scalable fine-tuning, making them applicable to a variety of tasks while reducing computational costs.

Problem Definition

The key challenge addressed in this work is enhancing spatial control in text-to-image diffusion models, and ControlNet, a novel neural network model, is the solution. The authors lists out a couple of the problem statements in their paper, in no particular order.

- Finer Spatial Control- While current text-to-image diffusion models are powerful, they struggle with fine-grained spatial control. It can be challenging for users to specify detailed attributes like object positioning, shapes, or human poses using only text prompts. This leads to a time-consuming trial-and-error process, where users repeatedly modify prompts to achieve the desired result. There is a growing need for models that allow users to input additional visual data—like edge maps, pose skeletons, or segmentation maps—as conditioning inputs. These visual guides would help the model generate images with more precision and customization, significantly reducing the effort required to reach the desired output.
- End-to-End Learning- The aim is to develop an end-to-end learning system that can seamlessly integrate additional conditional controls like edge maps or poses—into the diffusion process. This system should retain the generative power of large, pretrained text-to-image models while adding spatial conditioning capabilities.
- Data Scarcity- One of the major challenges in implementing these conditional controls is the limited availability of large, high-quality datasets. For instance, datasets for edge maps or pose skeletons are much smaller—up to 50,000 times smaller—than the vast datasets (like LAION-5B) typically used to train general text-to-image models. Overcoming this data scarcity is essential for effectively integrating conditioning inputs.
- Overfitting and Catastrophic Forgetting- When fine-tuning large pretrained neural network models with limited conditional data, there's always a risk of overfitting to the small dataset or experiencing "catastrophic forgetting" of the original model's capabilities. The challenge lies in ensuring that the model's fine-tuning does not degrade its general performance, while still learning to incorporate new conditional controls. This overfitting happens mainly, when latent diffusion models trained with large dataset is used to train relatively smaller datasets.
- Architectural Design for Handling Diverse Inputs- Another significant challenge is designing a neural network architecture capable of processing diverse types of conditioning inputs, such as edge maps, shapes, or high-level semantic features, while maintaining the image quality and generative power of the pretrained model. The architecture needs to be flexible enough to integrate both textual and spatial information seamlessly.

Below is how ControlNet is differentiable in solving these problems. It introduces a method for adding spatial control to diffusion models, particularly Stable Diffusion, using various conditioning inputs like Canny edges, Hough lines, user scribbles, human key points, segmentation maps, depth, and cartoon line drawings. ControlNet ensures that fine-tuning the model does not lead to

overfitting or catastrophic forgetting, providing a robust solution for generating complex and detailed images with better spatial control. To validate this approach, ControlNet locks the parameters of the large pretrained model and introduces a trainable copy with zero-initialized convolution layers, minimizing noise during training. The research shows that ControlNet maintains the stability and capabilities of the large diffusion model while adding flexible conditioning. It is also robust and scalable across various experimental settings and requires minimal computational resources, running efficiently on a single A100 GPU. Despite being smaller in scale, ControlNet competes with industrial-scale models by efficiently managing diverse conditioning inputs. By addressing these challenges, ControlNet enables more precise and customizable image generation, enhancing the spatial control of text-to-image diffusion models.

Methodology

- Model, Algorithm and Technique: For this work, a vanilla denoising diffusion probabilistic model (DDPM) is used. This model implements a variant of the UNet architecture that processes image data with the added capability of handling time-dependent features via sinusoidal time embeddings. The network includes multiple downsampling, mid-level, and upsampling blocks, each containing ResNet-style layers and attention mechanisms to ensure efficient feature learning and focus. Diffusion models originally depicted by (Ho, Jain, and Abbeel 2020) in 2020 at a high level, in Diffusion Models the data is transformed into a latent space through Markov chain designed to add noise incrementally, making the data increasingly random until it becomes indistinguishable from pure Gaussian noise in the forward process. In the backward process, it learns how to reverse this process to reconstruct the original data. Once trained, Diffusion Models can generate new data by starting from random noise and applying this learned denoising process in reverse. , the forward process (noising) is modeled as a Markov chain where each step $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ adds noise according to a predefined schedule. In the Reverse Diffusion (denoising) the goal of the Diffusion Model is to learn the reverse of the above process, which is also a Markov chain but in reverse. This reverse process is denoted as $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$.

  Let $\mathbf{x}_0$ be the original data sample (e.g., an image). where, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ represent the sequence of increasingly noisy samples produced by the forward diffusion process. - The forward process is defined as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

  where β_t controls the amount of noise added at each step. These values are typically scheduled across the timesteps and usually square root or linear schedulers are used for simpler datasets and coside beta schedulers from OpenAI for more popular large datasets (Sohl-Dickstein et al. 2015), (Ho, Jain, and Abbeel 2020), (Nichol and Dhariwal 2021). - The model learns the reverse process:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

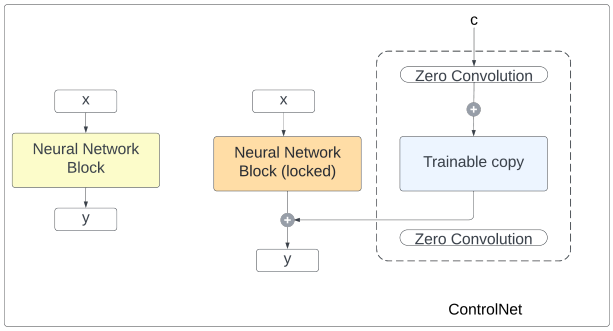

Figure 1: A neural block takes a feature map x as input and produces another feature map y (left). To add a ControlNet, the original block is locked, and a trainable copy is created. They are connected using zero convolution layers (1x1 convolution with zero weights and bias). A conditioning vector c is added to the network. (right)

ControlNet was introduced by Lvmin Zhang and Maneesh Agrawala from Stanford University in their February 2023 paper titled "Adding Conditional Control to Text-to-Image Diffusion Models." (Zhang, Rao, and Agrawala 2023) It works by creating two copies of the model's weights: a "trainable copy" and a "locked copy." The locked copy retains the knowledge from the pretrained model, while the trainable copy is fine-tuned on specific tasks to adapt to new conditions.

The key architecture 'as depicted' in Figure 1 similar to (Zhang, Rao, and Agrawala 2023). is its use of zero convolution to connect the trainable and locked copies. This layer starts with weights initialized to zero and learns optimal values without introducing noise to the model's features. A neural block, denoted as $F(x;\Theta)$, transforms an input feature map x into an output y,

$$y = F(x;\Theta)$$

ControlNet adds a trainable copy of the original block, with parameters Θ_c, while freezing the original block's parameters Θ. The trainable copy takes an external condition c and is connected to the locked model via zero convolution layers, denoted $Z(\cdot;\Theta_z)$, initialized with weights and biases set to zero.

The full ControlNet computation is,

$$y_c = F(x;\Theta) + Z(F(x + Z(c;\Theta_{z1});\Theta_c);\Theta_{z2})$$

Initially, the zero convolution terms are zero,

$$y_c = y$$

This initialization prevents random noise from influencing the network's hidden states at the start of training, while retaining the original pretrained model's functionality as a strong backbone for further task-specific

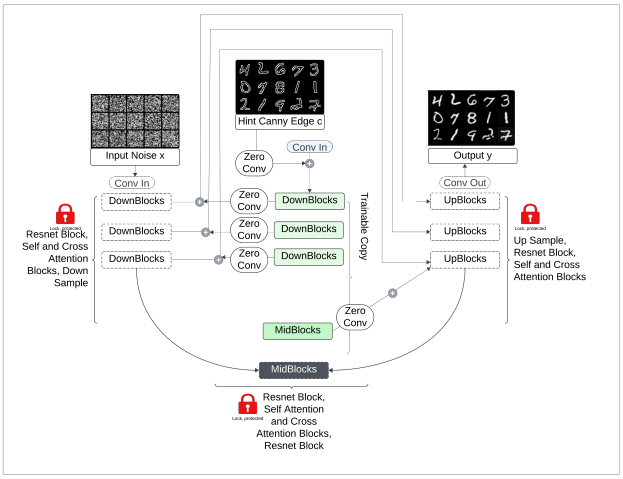

Figure 2: A rough, schematic of ControlNet architechture with DDPM on MNIST using canny edges

learning. Image diffusion generative models as we know, learns to progressively denoise images The remarkable aspect of ControlNet is that it allows fine control over how extra conditions influence denoising diffusion process. Stable Diffusion relies on Classifier-Free Guidance (CFG), where the final output ϵ_{prd} is computed as,

$$\epsilon_{\text{prd}} = \epsilon_{\text{uc}} + \beta_{\text{cfg}}(\epsilon_{\text{c}} - \epsilon_{\text{uc}})$$

Here ϵ_{uc} is the unconditional output, ϵ_{c} is the conditional output, and β_{cfg} is a weight controlling the guidance strength. With ControlNet, conditioning images can be added to both ϵ_{uc} and ϵ_{c} or only to ϵ_{c}, which impacts the strength of the guidance. In cases where no prompts are given, adding conditions to both outputs can reduce guidance (resulting in weaker outputs), while applying conditions only to ϵ_{c} can strengthen the guidance significantly.

In image diffusion algorithm, given an initial image z_0, noise is progressively added to generate a noisy image z_t. A network ϵ_θ is trained to predict the noise added, using conditions like time step t, text prompts c_t, and task-specific conditions c_f. The learning objective is:

$$L = E_{z_0,t,c_t,c_f,\epsilon\sim N(0,1)}\left[\|\epsilon - \epsilon_\theta(z_t, t, c_t, c_f)\|_2^2\right]$$

In our particular case, the ControlNet architecture details is shown in Figure2. The unconditional Unet model used here is designed for image generation tasks where both spatial and temporal context are essential. By incorporating time embeddings derived from a sinusoidal function and integrating them into a UNet architecture with attention mechanisms, the model aims to generate realistic images conditioned on a sequence of time steps, making it particularly useful in tasks such as denoising diffusion probabilistic models (DDPM) and video generation.

The architecture consists of three main components: DownBlocks, a MidBlock, and UpBlocks. In addition, a time embedding is used to condition the model on the time step, influencing the image generation process at each stage.

The time embedding t_{emb} is computed using a sinusoidal function that encodes time steps into a continuous representation. Given the time step t, the sinusoidal time embedding is defined as:

$$t_{\text{emb}}(t) = \left[\sin\left(\frac{t}{10000^{i/d}}\right), \cos\left(\frac{t}{10000^{i/d}}\right)\right],$$

$$i = 0, \ldots, \frac{d}{2} - 1$$

where $t \in Z$ is the time step, and d is the dimension of the embedding. This embedding is subsequently used in the network to modulate the processing of the image at each time step.

The DownBlock progressively downscales the input image while increasing the depth of the feature maps. Each DownBlock consists of ResNet and Attention Mechanisms and downsampling layer. The ResNet block applies a series of convolutions, each followed by SiLU activation, with the time embedding added to modulate the transformation at each step. After the ResNet block, a multi-head self-attention mechanism is applied. Given an input feature map $\mathbf{X} \in R^{B\times C\times H\times W}$, the attention mechanism reshapes the map into a sequence of patches, performs attention, and reshapes it back:

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

where Q, K, V are the query, key, and value matrices derived from the input features, and d_k is the dimension of the key vectors.

A downsampling operation is then applied, typically via a strided convolution or average pooling, to reduce the spatial resolution of the feature maps. The output of the DownBlock is a feature map $\mathbf{X}_{\text{down}}$ that captures abstract features at a lower resolution.

The MidBlock processes the output of the final DownBlock and refines the feature representations. Two ResNet Blocks with time embedding. similar to the DownBlocks, two ResNet blocks are applied, each followed by time-conditional transformations. A multi-head attention mechanism is applied to capture long-range dependencies in the feature map, similar to the DownBlock.

The output of the MidBlock is a refined feature map that captures the most abstract representations of the image.

The UpBlock progressively upscales the image, combining the high-level feature representations from the DownBlocks to reconstruct the image. An upsampling operation is applied to increase the spatial resolution. A ResNet block is applied to the concatenated feature map, followed by a time-conditional transformation similar to the DownBlock and MidBlock, a multi-head attention mechanism is applied to capture long-range dependencies across the image.

The full unconditional UNet architecture consists of an encoder-decoder structure, where the input image is processed through a series of DownBlocks, followed by the MidBlock, and finally upsampled using the UpBlocks. The time embedding is passed through each block to guide the image generation process.

$$\mathbf{X}_{\text{in}} \rightarrow \text{DownBlocks} \rightarrow \text{MidBlock} \rightarrow \text{UpBlocks} \rightarrow \mathbf{X}_{\text{out}}$$

where $\mathbf{X}_{\text{in}}$ and $\mathbf{X}_{\text{out}}$ represent the input and output images, respectively.

- Method and Experimental Setup: The aim of this work is to implement ControlNet on a dataset that can be trained and inferred using a DDPM with constraint Compute GPU availability (A100 Nvidia GPU) on google colab as demonstration of proof of concept. The training a diffusion model and implement ControlNet successfully following github code presented by (Zhang, Rao, and Agrawala 2023) here

  — https://github.com/lllyasviel/ControlNet

  All Pre trained stable Diffusion Models from (Zhang, Rao, and Agrawala 2023) work are often trained on large data set e.g. LAION-5B with 2 billion images. Thus instead of using such computation heavy dataset, ControlNet is being implemented on MNIST image dataset. The test and training label csv files were obtained from Kaggle.

  — https:
  //www.kaggle.com/datasets/oddrationale/mnist-in-csv

  Its a widely-used benchmark in machine learning, consisting of 70,000 grayscale images of handwritten digits (0-9) with a resolution of 28x28 pixels. It includes 60,000 training samples and 10,000 test samples. Due to its simplicity and well-labeled structure, MNIST serves as an excellent dataset for training and evaluating models like ControlNet. YAML file is defined with dataset, diffusion, model and training parameters below.

Experimental Result

First we train the Unet diffusion model without ControlNet, with 60k Mnist Dataset training samples and using Nvidia A100 cluster from google colab. A few smaller samples were initially used to estimate the training and loss. Using linear beta schedular, Adam Optimizer and MSE loss training took 1 hr 37 min 42 sec or 5861.88 seconds to complete with 40 epochs. For each image batch, model adds random noise to images, predicts the noise using the U-Net model, computes the loss between predicted noise and actual noise, backpropagates the loss and updates model weights, saves the model checkpoint after each epoch. The iterative losses are depicted in Figure 3.

Next, the now trained DDPM MNIST model is sampled and inferred where reverse diffusion is performed to progressively denoise a random Gaussian noise image,

Parameter	**Value**
dataset_params	{'im_path': 'data/mnist/train/images', 'im_test_path': 'data/mnist/test/images', 'canny_im_size': 28}
diffusion_params	{'num_timesteps': 1000, 'beta_start': 0.0001, 'beta_end': 0.02}
model_params	{'im_channels': 1, 'im_size': 28, 'hint_channels': 3, 'down_channels': [32, 64, 128, 256], 'mid_channels': [256, 256, 128], 'down_sample': [True, True, False], 'time_emb_dim': 128, 'num_down_layers': 2, 'num_mid_layers': 2, 'num_up_layers': 2, 'num_heads': 4}
train_params	{'task_name': 'mnist', 'batch_size': 64, 'num_epochs': 40, 'controlnet_epochs': 1, 'num_samples': 25, 'num_grid_rows': 5, 'ddpm_lr': 0.0001, 'controlnet_lr': 0.0001, 'ddpm_ckpt_name': 'ddpm_ckpt.pth', 'controlnet_ckpt_name': 'ddpm_controlnet_ckpt.pth'}

Table 1: Parameters used for the U-Net Diffusion Model

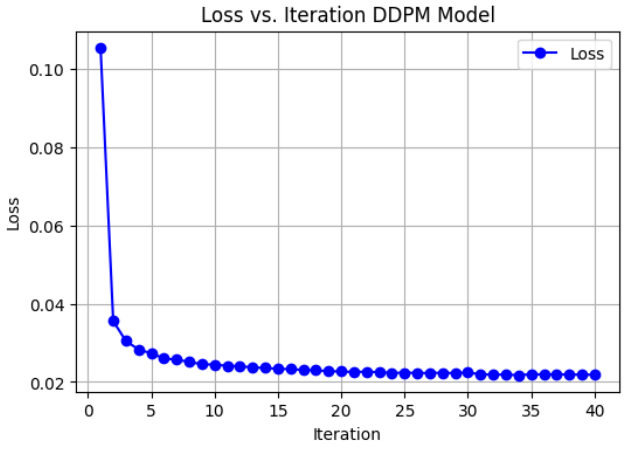

Figure 3: DDPM losses vs iteration

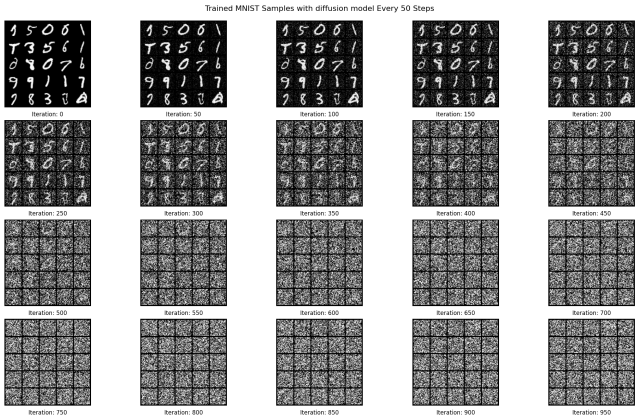

Figure 4: Reverse diffusion process on the unconditional DDPM trained with MNIST. The model starts with a noisy image at iteration $t = 1000$ and progressively denoises it to recover a clear MNIST digit image at $t = 0$. The initial noisy input image here is shown at (x_{950}), while at $t = 0$ image represents the reconstructed image (x_0).

transforming it into a high-quality output image. During the sampling and inference process, it initializes a noisy image tensor $\mathbf{x}_t$ with random Gaussian noise of the specified shape, iterates through 1000 timesteps in reverse order up to $t = 0$, predicts the noise using the U-Net model, and uses the scheduler to estimate $\mathbf{x}_0$ (the original image) and $\mathbf{x}_{t-1}$ (a less noisy image). It saves the predicted $\mathbf{x}_0$ (and intermediate predictions) as a grid of images at each timestep. The inference generates new images using a pre-trained diffusion model.

Next, ControlNet is implemented on the trained DDPM U-Net model and "ControlNet copy" of the U-Net is created, where certain layers are removed to focus on the control mechanism, which incorporates hints into the generation process. A hint block is added to process conditioning images through several convolutional layers. Next, Zero convolution layers are applied to the encoder (down blocks) and mid-blocks to ensure that the control network's influence on the model is isolated. This is achieved by using zero weights and biases in these layers, which do not contribute directly to model parameters during training. The forward method involves passing the noisy image through the trained U-Net while freezing its parameters. Passing the control signal (hint) through the control copy U-Net and hint block. Combining outputs from both the trained U-Net and control copy U-Net at various stages of the network, particularly during the mid-blocks and upblocks. The final output is a denoised image that integrates the conditioning information provided by the control network.

Next, the ControlNet model was trained for diffusion-based image generation, It incorporated a Linear noise scheduler and utilizes random noise and hints to guide the model's learning process. The samples are generated by performing reverse diffusion over time steps, utilizing a trained 'ControlNet' model and a noise scheduler.

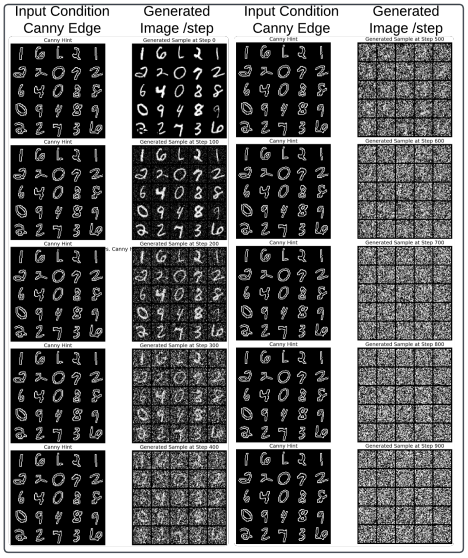

Figure 5: Controlling Diffusion Model using ControlNet, on each pair of columns, left row is input condition (canny edge) while the right row is time evolution of reverse diffusion process.

It started with random noise, progressively denoised the image while incorporating hints from the dataset, and saved intermediate $\mathbf{x}_0$ predictions. . The inference function loaded the ControlNet DDPM checkpoint and produced and saved samples from the trained model. Figure 5 shows final plot of the generated samples and their Canny hints at specified intervals.

The time evolution video is shown here -

— https://anonymous.4open.science/r/ControlNet-ECE570-D5C5/controlnet_time_evolution.mp4

Conclusion and Future Directions

In this work, ControlNet was successfully demonstrated on the MNIST dataset using Canny edge maps as hints for the diffusion model. By leveraging the ControlNet framework, which incorporates additional hint information during the generation process, we showed that the model could effectively guide the denoising process, producing high-quality samples. The comparison of generated images with their corresponding Canny edges at different timesteps highlighted the model's ability to preserve important edge structures while generating realistic digits. This approach opens new possibilities for controlled image generation tasks where edge or structural information is critical. For future work, one approach is to add a separate convolutional layer that incorporates artistic filters (e.g., edge detection, color transformations) or style-based filters inspired by style transfer techniques. One example would be to implement Impressionistic Customization, adding textures and soft brushstroke-like effects to the input control (using GaussianBlur and adding different noise levels in CV2). This ap-

proach will ensure that the project starts with manageable tasks and gradually scales up in complexity, allowing for a deep understanding of ControlNet before attempting more advanced extensions.

Note- LLM model Chatgpt was used to make the text & code content more concise and coherent and used for troubleshooting.

Acknowledgment- Current reimplementaton is adopted from the original Codebase are found in below repositories-

https://github.com/lllyasviel/ControlNet,
https://github.com/explainingai-code/VAE-Pytorchdata-preparation, https://github.com/milesial/Pytorch-UNet,
https://github.com/explainingai-code/ControlNet-PyTorch,
https://github.com/zhixuhao/unet

References

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. arXiv:2006.11239.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Mou, C.; Wang, X.; Xie, L.; Wu, Y.; Zhang, J.; Qi, Z.; and Shan, Y. 2024. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 4296–4304.

Nichol, A.; and Dhariwal, P. 2021. Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. 8748–8763. PMLR.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. 10684–10695.

Rosenfeld, A.; and Tsotsos, J. K. 2018. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 42(3): 651–663.

Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. arXiv:1503.03585.

Zhang, L.; Rao, A.; and Agrawala, M. 2023. Adding conditional control to text-to-image diffusion models. 3836–3847.