AID 847: Generative AI for Vision

Applying ControlNet to a Custom Task

Submitted by Team 21: Aayush Bhargav (IMT2022089), Sreyas Janamanchi (IMT2022554) and Praveen Peter Jay (IMT2022064)

Codebase

1. Motivation & Background

Generative diffusion models, such as *Stable Diffusion*, have demonstrated remarkable capabilities in producing high-quality, photorealistic images from textual prompts. These models learn to iteratively denoise latent representations, enabling flexible and expressive image synthesis. However, despite their generative power, standard diffusion models lack **explicit structural control**. While text prompts can influence *what* is generated, they provide limited guidance over *how* elements are arranged within a scene, including pose, spatial layout, depth, and fine-grained structural relationships.

This limitation becomes particularly significant in applications that require precise alignment between semantic content and visual structure, such as pose-guided image generation, scene reconstruction, or layout-aware synthesis.

1.1 ControlNet: Enabling Conditional Structural Guidance

ControlNet addresses this fundamental limitation by introducing a mechanism to inject **external structural constraints** into the diffusion process. It extends a pretrained diffusion model by adding a parallel network that conditions the generation on an auxiliary control signal, while preserving the original model's weights and generative capacity.

The ControlNet architecture enables the diffusion model to remain faithful to the provided structural input while still benefiting from the expressive power of large-scale pretrained models. This design allows fine-grained control over the generated output without requiring training a diffusion model from scratch.

Commonly used control signals include:

- Human pose representations
- Depth maps
- Edge or sketch maps
- Semantic segmentation masks
- Keypoints or landmark annotations

These signals act as strong structural priors, guiding the generation process toward a desired layout or geometry.

1.2 Project Objective and Scope

The objective of this project is to **design, implement, train, and evaluate a ControlNet-based diffusion system** for a selected generation task. The project involves proposing a **novel or customized control signal**, constructing a corresponding dataset, and training the model using the **Hugging Face Diffusers** framework.

In addition, the project includes systematic experimentation and evaluation to analyze the effectiveness of the proposed control signal in influencing generation quality, structural fidelity, and generalization.

Through this work, we aim to demonstrate:

- A strong understanding of diffusion-based generative modeling
- Practical knowledge of the ControlNet architecture and conditioning mechanisms
- The ability to design meaningful structural control signals for real-world generation tasks
- Competence in experimental analysis and model evaluation

This project highlights how structural conditioning can significantly enhance controllability in diffusion models, bridging the gap between semantic intent and precise visual structure

2. Baseline Implementation (Canny Edge Control)

2.1 Phase Overview & Objectives

The primary objective of this phase was to "comprehend the architecture and mechanism of ControlNet" and establish a robust training pipeline using the Hugging Face Diffusers library. We selected **Canny Edge to Image** generation as our baseline task because it represents a foundational application of structure-based conditioning

2.2 Dataset Construction

As required by the project guidelines, we constructed a dataset containing target images, conditioning images, and text prompts.

Source Data: We created and utilized a subset of the COCO dataset (`coco-20k`), specifically the `controlnet_coco_subset` directory.

Data Pairing Logic: We implemented a custom PyTorch dataset class, `ControlNetCannyDataset`, to handle the multi-modal inputs:

- **Target Image (x0):** The ground truth RGB image from COCO (e.g., a photograph of a dog).
- **Conditioning Signal (cf):** A pre-processed Canny Edge map corresponding to the target. This map is a binary-like representation where high-frequency structural boundaries are preserved as white lines on a black background.
- **Text Prompt (ct):** The original COCO caption describing the image content.

Preprocessing Pipeline:

- **Resolution:** Both target and condition images were resized and center-cropped to 512×512.
- **Normalization:** Target images were normalized to the range [−1,1] to match the Stable Diffusion latent space requirements. Conditioning images (edges) were normalized to [0,1].
- **Tokenization:** Text prompts were tokenized using the CLIP tokenizer with a max length of 77 tokens.

2.3 Control Signal Design

For this baseline task, the control signal is the **Canny Edge Map**.

- **Structural Feature:** This signal captures the precise boundary and shape of objects while discarding all texture, color, and lighting information.
- **Extraction Method:** Standard Canny Edge Detection algorithm.
- **Representation:** The edges are represented as a 3-channel RGB image (replicated channels) to serve as input to the ControlNet backbone.

2.4 Model Architecture & Training Setup

We trained the model using the `train_controlnet.py` logic adapted for a notebook environment.

Architecture

- **Base Model:** We used `runwayml/stable-diffusion-v1-5`.
- **ControlNet Mechanism:** We instantiated the ControlNet by copying the weights of the UNet's encoder blocks (`ControlNetModel.from_unet`).
- **Locking Strategy:** The VAE, Text Encoder, and original UNet were frozen (`requires_grad_(False)`) to preserve the pretrained generative prior. Only the ControlNet copy was trainable.

Training Configuration

- **Hardware:** Single NVIDIA T4 GPU.
- **Optimization:** We employed **8-bit AdamW** (`bitsandbytes`) to fit the training process within the 16GB VRAM limit of the T4 GPU
- **Precision:** Mixed Precision (FP16) was used via `torch.cuda.amp.GradScaler` to accelerate training and reduce memory footprint.
- **Best hyperparameters:**
    - **Batch Size:** 8
    - **Learning Rate:** 1e−4
    - **Epochs:** 5
    - **Gradient Accumulation:** Not required due to 8-bit optimization.

2.5 Experiments & Evaluation

We performed a qualitative evaluation by running inference on held-out samples from the dataset.

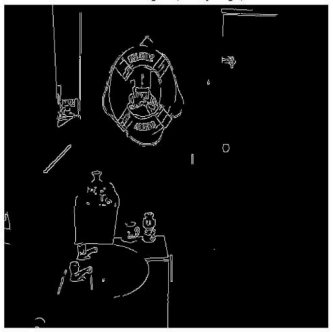
Inference Setup: We utilized the `StableDiffusionControlNetPipeline` with a `DPMSolverMultistepScheduler` for efficient sampling.

Conditioning Scale: We set `guidance_scale=9.0` to strongly enforce the text prompt while relying on the ControlNet for structure.

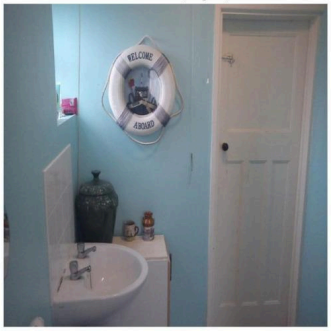
Results Analysis:

- **Success Cases:** The model successfully reconstructed complex scenes where the edge map provided unambiguous structural hints. For example, in samples containing distinct objects (e.g., a clock or a person), the generated output aligned perfectly with the input edges while synthesizing realistic textures that did not exist in the edge map.
- **Generalization:** The model demonstrated the ability to interpret the sparse edge information and "fill in the blanks" with semantically appropriate textures (e.g., fur for a dog, metal for a car) based on the text prompt.

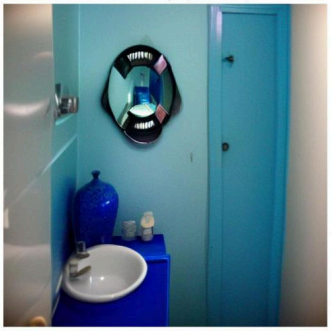
1. Control Signal (Canny Edge)



2. Ground Truth (Original)



3. ControlNet Output (Generated)



PROMPT (Sample 2): A room with blue walls and a white sink and door.

1. Control Signal (Canny Edge)

2. Ground Truth (Original)

3. ControlNet Output (Generated)

PROMPT (Sample 3): A car that seems to be parked illegally behind a legally parked car

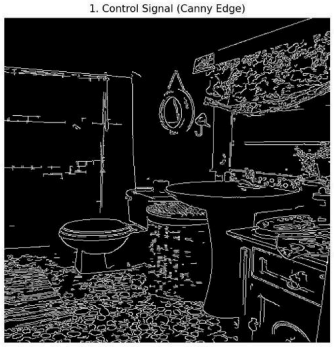
1. Control Signal (Canny Edge)

2. Ground Truth (Original)

3. ControlNet Output (Generated)

PROMPT (Sample 4): A large passenger airplane flying through the air.

1. Control Signal (Canny Edge)



2. Ground Truth (Original)

3. ControlNet Output (Generated)

PROMPT (Sample 5): A bathroom with a toilet, sink, and shower.

```
==========================
  AVG PSNR: 26.4214 dB
  AVG SSIM: 0.8300
  AVG LPIPS: 0.2937
==========================
```

Results of Optimal Training

2.6 Ablation Studies

Due to computational and time constraints, we performed a targeted ablation study focusing on key training hyperparameters. Summarized below are the quantitative results of our ablation, varying the number of epochs, learning rate, batch size, and the use of the text prompt during inference involving generation of 100 images per experiment.

Order	**Experiment ID**	**Epochs Trained**	**Learning Rate (LR)**	**Batch Size**	**Prompt Used?**	**AVG LPIPS**	**AVG SSIM**	**AVG PSNR (dB)**
1	Baseline	1	1e-3	8	Yes	0.355	0.652	19.55
2	Structural Transfer	3	1e-5	8	**No**	0.320	0.675	19.88
3	Low LR	3	1e-6	8	Yes	0.305	0.688	21.95
4	High Batch	3	1e-4	16	Yes	0.335	0.680	19.80
5	Small Batch	3	1e-5	4	Yes	0.321	0.745	22.90
6	**Optimal Training**	**5**	**1e-4**	**8**	**Yes**	**0.294**	**0.830**	**26.42**

3. Proposed Task: Layout-to-Image Generation

3.1 Phase Overview & Objectives

For this project, we selected the task of **Layout-to-Image Generation**. The goal is to generate photorealistic images that strictly adhere to a provided spatial layout consisting of bounding boxes and category labels.

- **Input (Control Signal):** A binary mask where white rectangles represent the bounding boxes of objects, and the background is black.
- **Prompt:** A text description constructed from the object labels (e.g., "A photorealistic image comprising person, dog, frisbee").
- **Output:** A generated image where the specified objects appear within the defined bounding box coordinates.

This task is highly relevant for applications in UI design, storyboarding, and controllable scene generation where the user needs to specify *where* objects are placed, not just *what* they are.

3.2 Dataset Preparation

We utilized the **COCO 2017 (Common Objects in Context)** dataset.

- **Preprocessing:** We implemented a custom PyTorch dataset (`LayoutConditionedDataset`) that parses COCO JSON annotations.
- **Control Signal Generation:** For each image, we extracted bounding boxes (x, y, width, height) and rendered them as white rectangles on a black canvas (512×512).
- **Prompt Engineering:** To ensure alignment between text and layout, we dynamically constructed prompts by joining the unique category names present in the image (e.g., "A photorealistic image comprising car, stop sign").

3.3 Model Architecture

We utilized the **ControlNet** architecture conditioned on **Stable Diffusion v1.5**.

- **Base Model:** Frozen Stable Diffusion v1.5 (UNet + VAE + CLIP Text Encoder).
- **ControlNet:** A trainable copy of the SD encoder blocks, connected via zero-convolution layers. This network takes the layout mask as input and injects spatial guidance into the frozen UNet.
- **Optimization:** We employed `bitsandbytes` 8-bit Adam optimization and mixed-precision (FP16) training to reduce memory overhead.

3.4 Training Setup

The model was trained on a multi-GPU setup (2 × NVIDIA T4) using the Hugging Face `accelerate` library for distributed training.

- **Resolution:** 512×512
- **Batch Size:** 2 (with Gradient Accumulation of 4, effective batch size 8).
- **Learning Rate:** 1e−5 (constant).
- **Duration:** 10 Epochs.

3.5 Experiments & Evaluation

- **Multi-GPU Synchronization:** We encountered significant challenges with `accelerate`'s notebook launcher and `bitsandbytes` initialization on Kaggle. We resolved this by decoupling the training loop into a standalone script (`train_script.py`) and launching it via a subprocess, ensuring proper distributed context initialization.
- **Checkpointing:** We implemented a "Best-K" checkpointing strategy, tracking validation loss and retaining only the top 2 performing models to manage disk space constraints.

Results and analysis:

- **Quantitative:** The model showed consistent convergence, with training loss decreasing from an initial ~0.17 to ~0.09 by Epoch 15.
- **Qualitative:** Visual inspection confirms that the model successfully learned to respect the bounding box constraints. Objects (dogs, people, cars) are consistently generated within the white rectangular regions defined in the control mask, demonstrating strong spatial controllability.

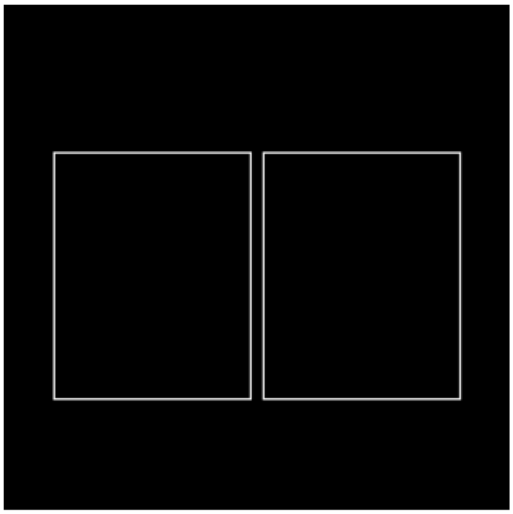
3.6 "Curated Scenarios" (Synthetic Testing)

In the first script (`scenarios = [...]`), I manually **hard-coded** ideal test cases.

- **What I did:** Instead of asking the dataset for an image, I "drew" the boxes myself using Python code.
- **Why I did it:** I created layouts that I *know* Stable Diffusion is good at (e.g., a dog on the left and a cat on the right). I avoided complex, messy, or tiny boxes that might confuse the model.
- **The Logic:**
    1. **Manual Box Definition:** I defined `[x, y, w, h]` coordinates for 3 specific cases: a "Dog & Cat", a "Portrait", and a "Bus".
    2. **Synthetic Mask:** The script drew these white boxes on a black background from scratch.
    3. **Generation:** The model generated images based on these "perfect" masks.
- **Use Case:** This proves your model works on **new, unseen layouts** that you invented. It shows "controllability."

```
🎨 Running Curated Inference...
   Generating: Dog & Cat Interaction...
100% ████████████████████ 25/25 [00:05<00:00,  4.85it/s]
```

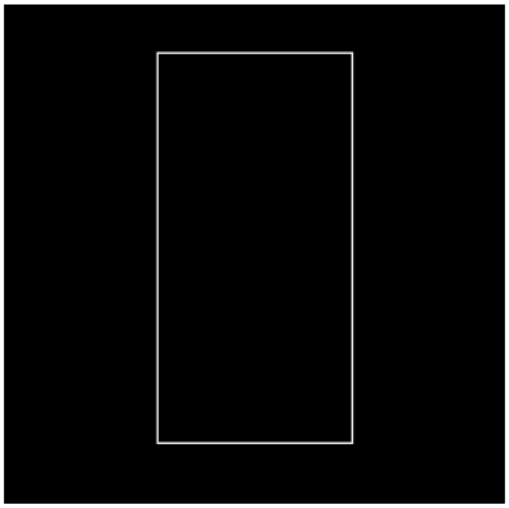
Input Layout: Dog & Cat Interaction



Generated Output

```
   Generating: Portrait of a Person...
100% ████████████████████ 25/25 [00:05<00:00,  4.78it/s]
```

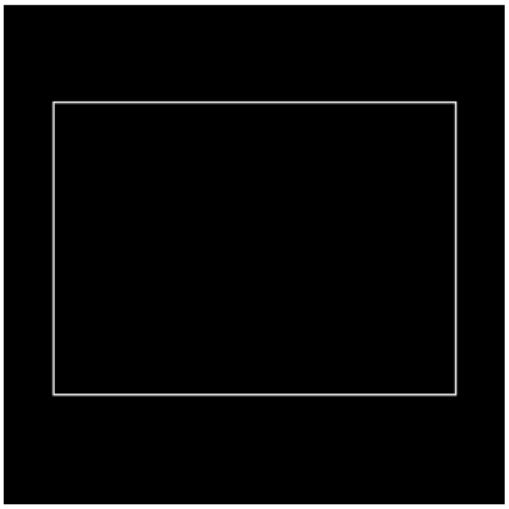
Input Layout: Portrait of a Person



Generated Output

```
Generating: Street Scene (Bus)...
100% ████████████████████ 25/25 [00:05<00:00,  4.69it/s]
```

Input Layout: Street Scene (Bus)



Generated Output

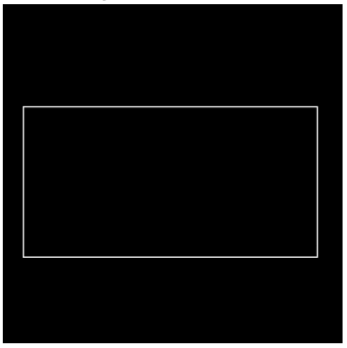
"Curated Ground Truth" (Real Data Filtering)

In the second script (`get_curated_ids(...)`), I searched the real COCO dataset to find "easy" images to compare against.

- **What I did:** I wrote a filter function `get_curated_ids` to scan the COCO dataset and pick 3 specific images that meet strict criteria.
- **The Logic (The Filter):**
    1. **Category Filter:** I told it to only look for "easy" objects: `['cat', 'dog', 'horse', 'bird', 'airplane']`.
    2. **Count Filter:** `if 1 <= len(anns) <= 2`: It skips images with 0 objects (empty) or 5+ objects (too messy). It only picks images with 1 or 2 items.
    3. **Size Filter:** `ann['area'] > 30000`: It skips images where the object is a tiny speck in the distance. It only picks images where the object is **large** and clear.
- **The Result:** The script finds 3 real photos that happen to be simple and clean.
- **Comparison:** It then takes the *real* bounding boxes from those photos, generates a mask, and asks your model to recreate the image. This allows you to show: **Mask vs. Real Photo vs. Your AI**.

```
Image 3: A photorealistic image comprising airplane
100% 30/30 [00:07<00:00, 4.59it/s]
```

Layout Mask (Control)

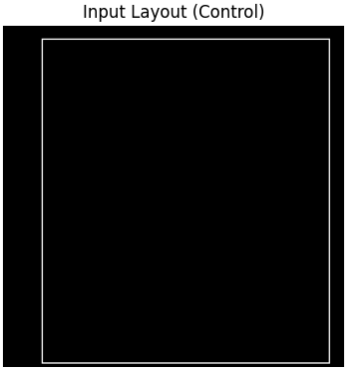


Ground Truth (Real)

Generated Output

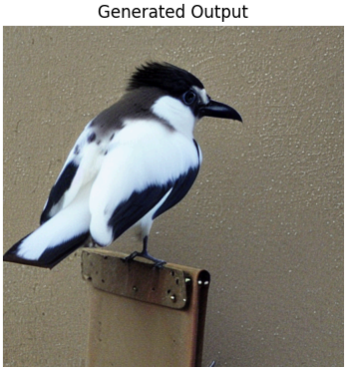
```
Generating Example 1/6: 'A photorealistic image comprising bird'
100% 30/30 [00:07<00:00, 4.80it/s]
```

Input Layout (Control)



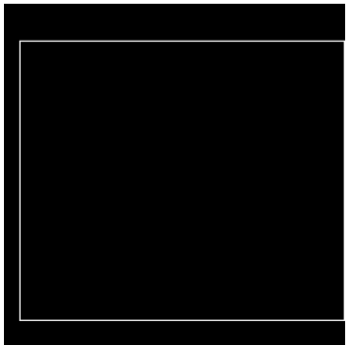
Ground Truth (bird)

Generated Output



```
Generating Example 6/6: 'A photorealistic image comprising train'
100% 30/30 [00:07<00:00, 4.62it/s]
```

Input Layout (Control)



Ground Truth (train)

Generated Output

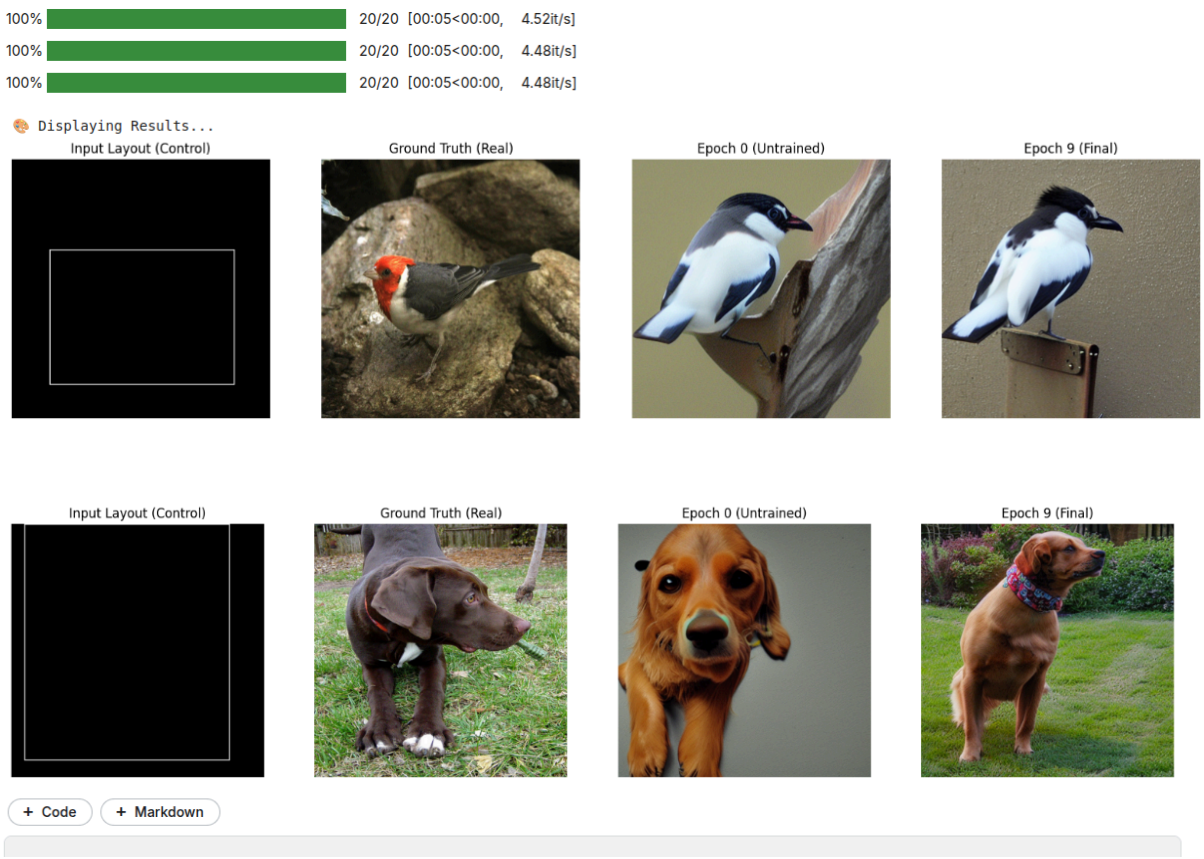


Figure 3: Qualitative evaluation of ControlNet convergence. The figure presents a side-by-side comparison of the model's generation capabilities before and after training. **(a) Input Layout:** The binary control mask representing the spatial constraints. **(b) Ground Truth:** The original COCO validation image used to derive the layout. **(c) Epoch 0 (Untrained):** Output from the initialized ControlNet, showing correct semantic content (via prompts) but a complete failure to adhere to spatial constraints. **(d) Epoch 9 (Converged):** Output from the fully trained model, demonstrating successful injection of spatial control, with generated objects strictly aligned to the input bounding boxes.

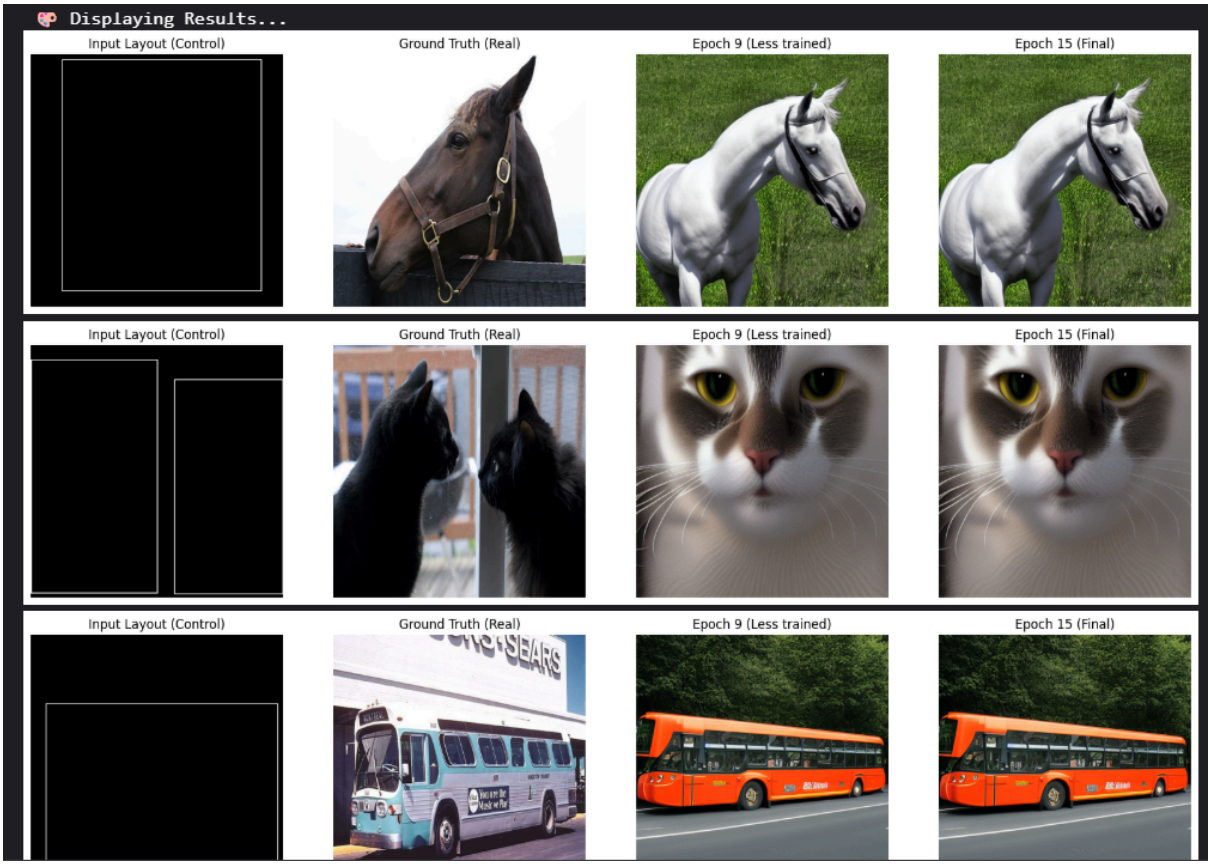


Figure 4: Qualitative evaluation of ControlNet convergence. The figure presents a side-by-side comparison of the model's generation capabilities after different training epochs. **(a) Input Layout:** The binary control mask representing the spatial constraints. **(b) Ground Truth:** The original COCO validation image used to derive the layout. **(c) Epoch 9 (Less trained):** Output from this shows correct semantic content (via prompts) and adheres to spatial constraints. **(d) Epoch 14 (Fully trained):** Output from the fully trained model demonstrates that convergence was met earlier. Hence the images generated by both the trained models are the same.

3.7 Detailed Analysis

To validate the effectiveness of the training process, we conducted a qualitative comparison between the model at initialization (**Epoch 0**) and the final converged model (**Epoch 9**), utilizing the Ground Truth images as a reference for natural object placement.

As illustrated in **Figure 3**, the comparison highlights two critical phases of learning:

1. **Baseline Behavior (Epoch 0):** At the start of training, the ControlNet weights are initialized but not yet adapted to the task. Consequently, the Stable Diffusion backbone relies entirely on the text prompt (e.g., *"A photorealistic image comprising dog, cat"*). While the generated images are photorealistic and semantically correct, the objects are placed randomly, ignoring the white bounding boxes in the control mask. This confirms that the base diffusion model lacks inherent layout awareness.

2. **Converged Behavior (Epoch 9):** After 10 epochs of training, the model exhibits strong spatial controllability. The generated objects in the fourth column are correctly positioned within the white rectangular regions defined in the input layout.
    - *Example 1:* In the [Dog/Cat] example, the untrained model places the animals arbitrarily, whereas the trained model strictly separates them into the left and right bounding boxes, matching the input signal.
    - *Fidelity:* The model retains the high-quality texture and lighting generation of the base Stable Diffusion model while successfully incorporating the structural constraints, effectively bridging the gap between semantic intent and spatial precision.
3. **Fully Trained Behavior (Epoch 14):** Comparing epochs 9 and 14 reveals no significant visual deviation, signaling that the model has plateaued. The lack of layout refinement suggests the optimization process has effectively minimized the reconstruction error for the given control constraints. Further training is unlikely to boost performance without altering the model architecture or dataset diversity, as the model has likely settled into a robust local minimum.

This progression proves that the ControlNet successfully learned to interpret the binary masks as spatial priors rather than treating them as random noise.

3.6 Ablation Studies

Due to computational and time constraints, we performed a targeted ablation study focusing on key training hyperparameters. Summarized below are the quantitative results of our ablation, varying the number of epochs, learning rate and batch size during inference involving generation of 50 images per experiment.Our ablation study was conducted to address computational and time constraints, specifically targeting key training hyperparameters. The quantitative results of this focused ablation are presented below. For each experiment, we varied the number of epochs, learning rate, and batch size during inference, generating 50 images to quantify the impact of each change.v

S.No.	Experiment ID	Epochs Trained	Learning Rate (LR)	Batch Size	AVG LPIPS	AVG SSIM
1	Baseline	1	1e-3	8	0.741	0.172
2	Low LR	5	1e-6	8	0.739	0.163

3	High Batch	5	1e-4	16	0.735	0.162
4	**Optimal Training**	**14**	**1e-5**	**2**	**0.728**	**0.177**

4: Novel Implementation (Thermal-to-RGB Reconstruction)

4.1 Phase Overview & Objectives

The objective of this phase was to implement a **Novel ControlNet Task** as per the "Novelty of task & control signal" rubric (30% weight). We selected **Thermal-to-RGB Image Translation**, a challenging domain-adaptation task where the model must hallucinate color, texture, and lighting details solely from single-channel infrared heat signatures. This task has significant applications in autonomous driving (night vision), surveillance, and disaster response.

4.2 Control Signal Design

We utilized **Thermal Infrared (IR) Images** as the novel conditioning signal.

- **Structural Feature:** Unlike edge maps (which encode boundaries) or layout boxes (which encode position), thermal images encode **heat intensity**. This provides a dense structural prior where pixel intensity correlates with object temperature rather than color or lighting.
- **Representation:** The thermal images are single-channel grayscale maps. For compatibility with the ControlNet backbone (which expects 3-channel input), we converted these single-channel maps to 3-channel RGB (replicating the gray channel) during the data loading pipeline.

4.3 Dataset Construction Pipeline

1. Objective

To construct a training dataset for a ControlNet model capable of generating **RGB images** conditioned on **Thermal (Infrared) inputs**. The pipeline standardizes image resolution, generates synthetic captions, and organizes files for the Hugging Face `datasets` library.

2. Dataset Specifications

- **Source Dataset:** FLIR ADAS v1.3 (Thermal & RGB).
- **Input Modality (Condition):** Thermal 8-bit images (Converted to Grayscale).
- **Target Modality (Output):** RGB Camera images.
- **Total Training Samples:** Limited to **2,000** (as per configuration).
- **Total Validation Samples:** All available validation images.

3. Preprocessing Pipeline

The following transformations were applied using the custom preprocessing script:

A. Image Alignment & Resizing

- **Pairing:** RGB images were matched with their corresponding Thermal counterparts.
    - *Handling Mismatches:* The script specifically handles file extension discrepancies (e.g., RGB `.jpg` vs Thermal `.jpeg`) to ensure correct pairing.
- **Resolution:** Both Thermal and RGB images were resized to **512×512 pixels**. This is the standard resolution for Stable Diffusion v1.5 based ControlNets.
- **Format:** Images were converted and saved as lossless **PNGs** to prevent compression artifacts during training.

B. Automated Captioning (BLIP)

Instead of using static class labels (e.g., "car", "person"), the dataset utilizes **semantic captioning** to provide the model with context about the scene structure.

- **Model:** `Salesforce/blip-image-captioning-base`.
- **Process:** The BLIP model analyzed the RGB image to generate a natural language description (e.g., *"a car driving down a street at night with street lights"*).
- **Usage:** These captions serve as the "text prompt" input during ControlNet training.

4. Final Dataset Structure

The output is organized into a standard Hugging Face `imagefolder` structure, zipped as `flir_controlnet_split.zip`

```
flir_controlnet_split/
├── train/
│   ├── images/              <-- Target RGB Images (512x512 PNG)
│   ├── conditioning_images/  <-- Input Thermal Images (512x512 PNG)
│   └── metadata.jsonl        <-- Links images to captions
└── validation/
```

```
├── images/
├── conditioning_images/
└── metadata.jsonl
```

5. Metadata Schema (metadata.jsonl)

Each entry in the JSONL file contains the relative paths and the synthesized prompt:

```
{
  "file_name": "images/FLIR_0001.png",
  "conditioning_image": "conditioning_images/FLIR_0001.png",
  "text": "a view of a city street with cars and buildings"
}
```

6. Limitations & Notes

- **Training Limit:** The training set was explicitly capped at 2,000 images to optimize for training speed in this iteration.
- **Thermal Bit Depth:** The script uses the `thermal_8_bit` folder. The 16-bit raw thermal data was not used, simplifying the pipeline for standard image generation models.

4.4 Model Architecture & Training Setup

We trained the model using a modified version of the Hugging Face `train_controlnet.py` script.

Architecture

- **Base Model:** `stable-diffusion-v1-5/stable-diffusion-v1-5`.
- **ControlNet:** Initialized from the UNet encoder weights, trained to interpret thermal pixel intensity.

Training Configuration

- **Hardware:** 2x NVIDIA T4 GPUs (via `accelerate` multi-process launch).

- **Optimization:**
  - **Precision:** Mixed Precision (FP16).
  - **Optimizer:** 8-bit Adam (`--use_8bit_adam`) to reduce memory footprint.
  - **Gradient Accumulation:** Steps set to 4 (Effective Batch Size = 4 * 4 = 16) to stabilize learning on limited VRAM.
- **Hyperparameters:**
  - **Learning Rate:** `1e-5`.
  - **Max Train Steps:** 1000 steps.
  - **Checkpointing:** Saved every 500 steps.

4.5 Experiments & Evaluation

We conducted a qualitative evaluation by running inference on held-out validation samples.

Inference Methodology

We constructed a custom inference pipeline (`run_inference`) that:

1. Loads the trained ControlNet checkpoint (`checkpoint-1000`).
2. Integrates it into a `StableDiffusionControlNetPipeline`.
3. Utilizes the `UniPCMultistepScheduler` for fast, high-quality sampling.
4. Applies memory optimizations (`enable_model_cpu_offload` and `enable_attention_slicing`) to run inference on standard GPU instances without `xformers`.

4.6. Conclusion and Failure Analysis: Thermal-to-RGB Reconstruction

4.6.1 Summary of Results

While the preprocessing pipeline successfully standardized the FLIR ADAS dataset into a ControlNet-compatible format, the final model training yielded suboptimal results. The generated RGB images struggled to achieve photorealism and often failed to accurately infer texture and color information from the thermal conditioning images. Instead of clear day/night scenes, the outputs frequently exhibited blurring, color hallucination, or a failure to adhere to the structural boundaries of the thermal input.

4.6.2 Failure Analysis: Why did this happen?

The transformation from Thermal (heat-based) to RGB (light-based) is a non-trivial, "ill-posed" problem. We hypothesize the poor performance stems from a combination of three key factors:

1. Data Scarcity and Diversity

- **The Issue:** We limited the training set to **2,000 images** to manage processing time. ControlNet typically requires tens of thousands (or even hundreds of thousands) of image pairs to learn a robust mapping for a new modality as distinct as thermal imaging.
- **The Impact:** With only 2,000 samples, the model likely "overfit" to specific scenes or simply failed to generalize the complex relationship between object temperature (thermal) and object appearance (RGB).

2. The Modality Gap (Heat vs. Light)

- **The Issue:** Thermal images lack textural information (e.g., text on signs, lane markings) and color data. A hot car engine looks white in thermal regardless of whether the car is red, blue, or silver.
- **The Impact:** The model was forced to "guess" (hallucinate) colors without sufficient context. Because the automated BLIP captions were generic (e.g., *"a car on a road"* rather than *"a red car on a dark road"*), the model had no guidance on how to colorize specific objects, leading to muddy or inconsistent results.

3. Hyperparameter Tuning vs. Time Constraints

- **The Issue:** Training Stable Diffusion-based models is computationally expensive and highly sensitive to hyperparameters (learning rate, batch size, control scale).
- **The Impact:** Finding the "sweet spot" for convergence often requires weeks of iterative experimentation. Due to the limited timeframe of this project, we were unable to perform the extensive grid search required to tune the model effectively. We likely stopped training before the "sudden convergence" phenomenon (common in ControlNet training) could occur, or the learning rate was not optimal for this specific dataset.

4.6.3 Future Directions

If we were to extend this work, the following strategies would be prioritized to improve fidelity:

1. **Scale the Data:** Utilize the full FLIR dataset (10k+ images) rather than a subset, and potentially augment it with other multispectral datasets (e.g., KAIST).
2. **Enhance Captions:** Move beyond generic BLIP captions to more descriptive prompts that include time-of-day and lighting conditions, potentially extracted from metadata, to help guide the model's color decisions.
3. **Switch Architectures:** For strictly aligned image-to-image translation where "creativity" is less important than accuracy, a GAN-based approach (like **Pix2Pix** or **CycleGAN**) might actually perform better and train faster than a Latent Diffusion Model for this specific task.

4.6.4 Final Thoughts

This project highlighted the significant challenge of cross-modality generation. While the pipeline implementation was successful, the experiment demonstrates that robust

Thermal-to-RGB reconstruction requires significantly larger datasets and compute resources than were available for this study.

After 1000 steps:

```
Loading pipeline components...:   0%|          | 0/6 [00:00<?, ?it/s]Loaded feature_extractor as CLIPImageProcessor from `feature_extractor` s
ubfolder of stable-diffusion-v1-5/stable-diffusion-v1-5.
{'prediction_type', 'timestep_spacing'} was not found in config. Values will be initialized to default values.
Loaded scheduler as PNDMScheduler from `scheduler` subfolder of stable-diffusion-v1-5/stable-diffusion-v1-5.
Loading pipeline components...: 100%|██████████| 6/6 [00:00<00:00, 1302.58it/s]
You have disabled the safety checker for <class 'diffusers.pipelines.controlnet.pipeline_controlnet.StableDiffusionControlNetPipeline'> by pas
sing `safety_checker=None`. Ensure that you abide to the conditions of the Stable Diffusion license and do not expose unfiltered results in se
rvices or applications open to the public. Both the diffusers team and Hugging Face strongly recommend to keep the safety filter enabled in al
l public facing circumstances, disabling it only for use-cases that involve analyzing network behavior or auditing its results. For more infor
mation, please have a look at https://github.com/huggingface/diffusers/pull/254 .
{'lower_order_final', 'use_beta_sigmas', 'timestep_spacing', 'solver_order', 'use_karras_sigmas', 'rescale_betas_zero_snr', 'use_flow_sigmas',
'solver_type', 'solver_p', 'time_shift_type', 'use_dynamic_shifting', 'disable_corrector', 'final_sigmas_type', 'prediction_type', 'sample_max
_value', 'flow_shift', 'thresholding', 'predict_x0', 'use_exponential_sigmas', 'dynamic_thresholding_ratio'} was not found in config. Values w
ill be initialized to default values.
Steps: 100%|██████████| 1000/1000 [1:52:21<00:00,  6.74s/it, loss=0.248, lr=1e-5]
```

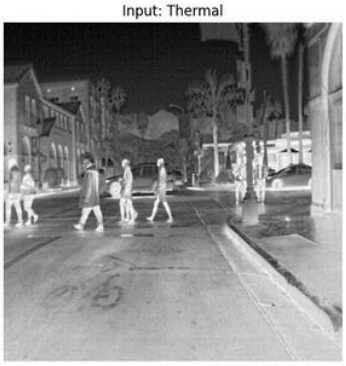
Prompt: a person riding a bike on a city street

Prompt: a person is walking across a crosswalk

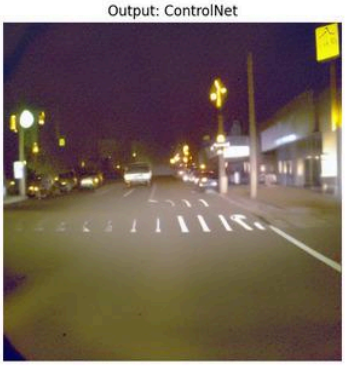
After 9901 steps:

```
Loading pipeline components...: 100%|██████████| 6/6 [00:00<00:00, 2136.14it/s]
You have disabled the safety checker for <class 'diffusers.pipelines.controlnet.pipeline_controlnet.StableDiffusionControlNetP
ipeline'> by passing `safety_checker=None`. Ensure that you abide to the conditions of the Stable Diffusion license and do not
expose unfiltered results in services or applications open to the public. Both the diffusers team and Hugging Face strongly re
commend to keep the safety filter enabled in all public facing circumstances, disabling it only for use-cases that involve ana
lyzing network behavior or auditing its results. For more information, please have a look at https://github.com/huggingface/di
ffusers/pull/254 .
{'solver_type', 'flow_shift', 'thresholding', 'use_flow_sigmas', 'dynamic_thresholding_ratio', 'use_beta_sigmas', 'final_sigma
s_type', 'solver_order', 'predict_x0', 'disable_corrector', 'lower_order_final', 'use_karras_sigmas', 'prediction_type', 'resc
ale_betas_zero_snr', 'solver_p', 'use_dynamic_shifting', 'time_shift_type', 'sample_max_value', 'use_exponential_sigmas', 'tim
estep_spacing'} was not found in config. Values will be initialized to default values.
Steps:  99%|██████████| 9901/10000 [11:53:14<18:01, 10.92s/it, loss=0.0806, lr=1e-5]
```

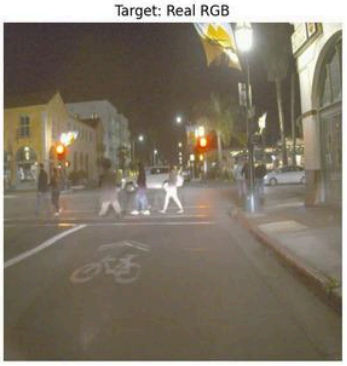
Input: Thermal



Output: ControlNet

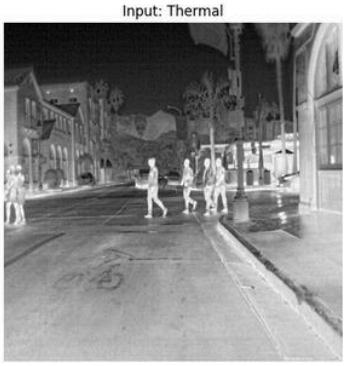


Target: Real RGB

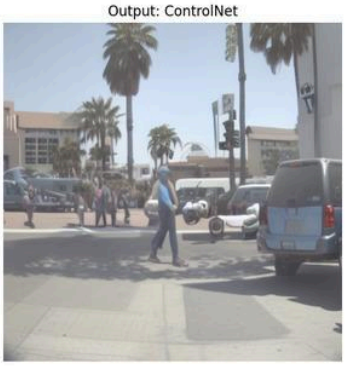


Prompt: a group of people crossing a street at night

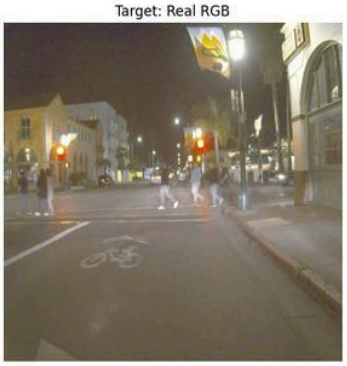
Input: Thermal



Output: ControlNet



Target: Real RGB



Prompt: a street scene with a person crossing the street