Maxwell's Generative Intelligence Lab/Paired Data Research Synopsis

Maxwell's Final Contributions to: "Customizing Text-to-Image Models with a Single Image Pair"

- Code
  - wrote all pipelines and testing code to generate final results/metrics for final results
  - Wrote all pipelines for rebuttal
  - Wrote a github repository so that others can also use the method (https://github.com/PairCustomization/PairCustomization)
    - Training code integrated into diffusers to train models using our method
    - Diffusers pipeline to run style guidance after models have been trained
    - Integration of style guidance with controlnet
    - Real image editing with style guidance
  - Wrote a nice website with results (https://paircustomization.github.io/)
  - Helped write code to make figures with help from Sheng-Yu
  - Helped write code to run
- User Study
  - Conducted all users studies and put results in the table
- Writing
  - Wrote initial version of method section, experiments section, and results section
  - Wrote initial version of all supplementary text
  - Helped edit all sections during editing process + after rebuttals

Second Project that I tried working on for Generative Intelligence Lab: Learning Image filters from large scale paired data

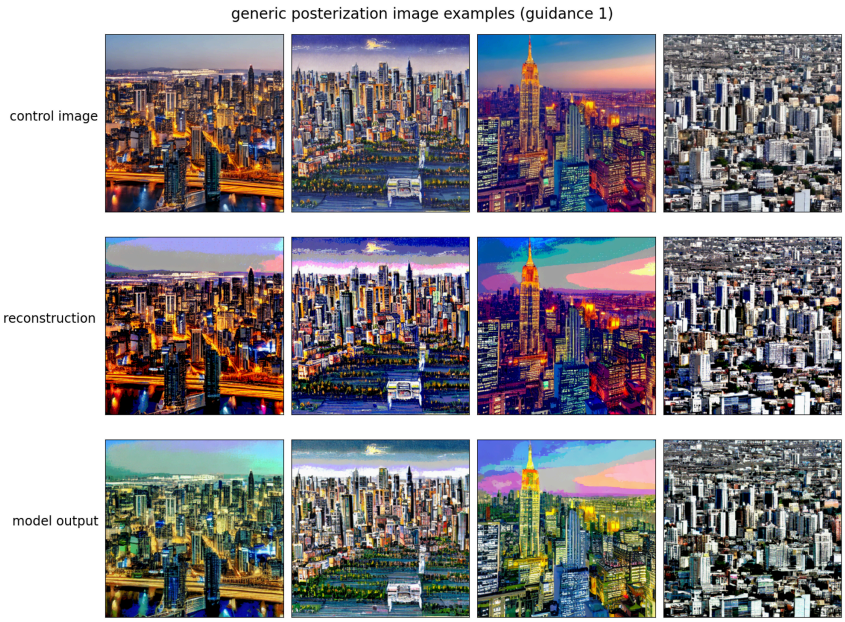
Idea: Can we learn some image filter transformation from a set of image pairs (original image -> filtered image)

Images and their posterized versions

Problem: Training a controlnet and running inference with classifier free guidance leads to some artifacts:

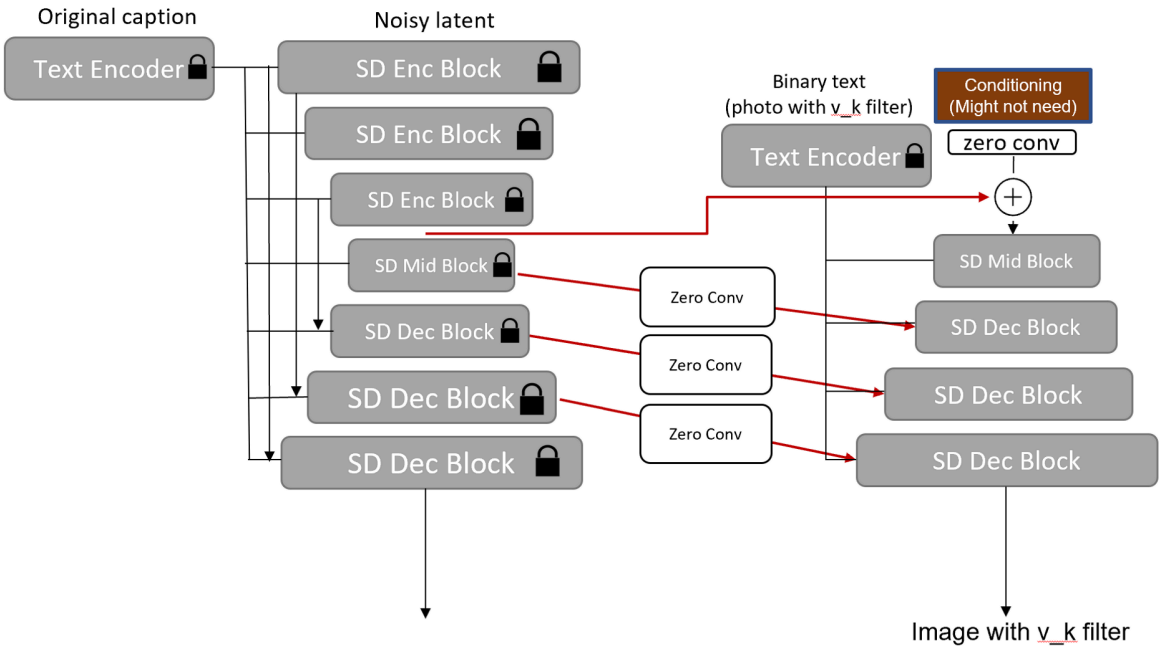
Posterization expected vs output:

Solution: Turning off cfg fixed some of these artifacts, however models still took days to train and ~50k or more images. This worked in some cases:



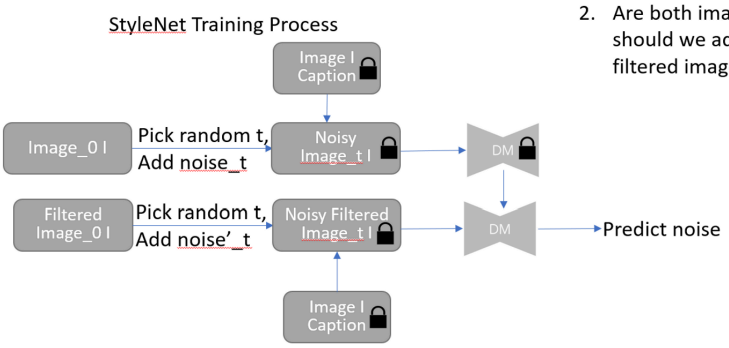
Reconstruction here represents the filtered image encoded and decoded by the VAE

Next step: Try and train a network that takes in both noisy image and noisy styled image, and uses the outputs of the noisy image to help denoise the styled image:

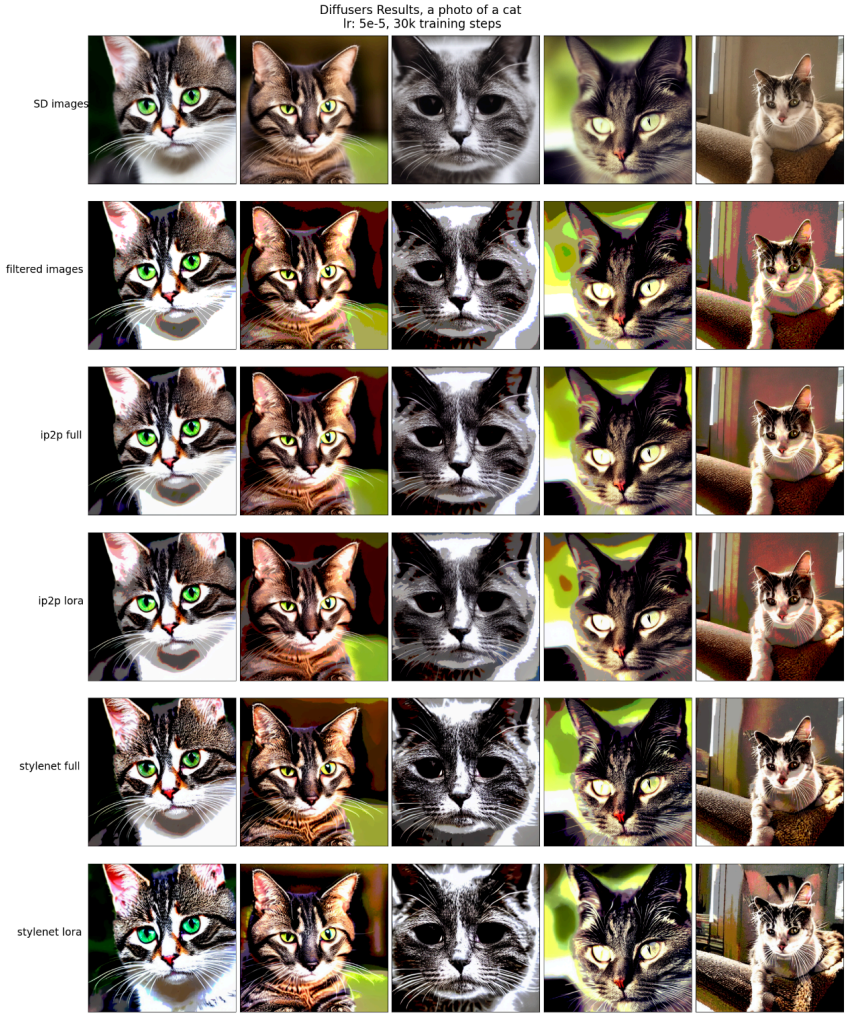


Design Decisions:

1. Are noise_t and noise'_t the same or different? Are they fundamentally indpenedent?
2. Are both image captions the same, or should we add ", with a v* filter" to the filtered image caption



Problem: This works, but it works less well than simply fine tuning instruct pix2pix with low rank adapters (better performance and faster convergence)

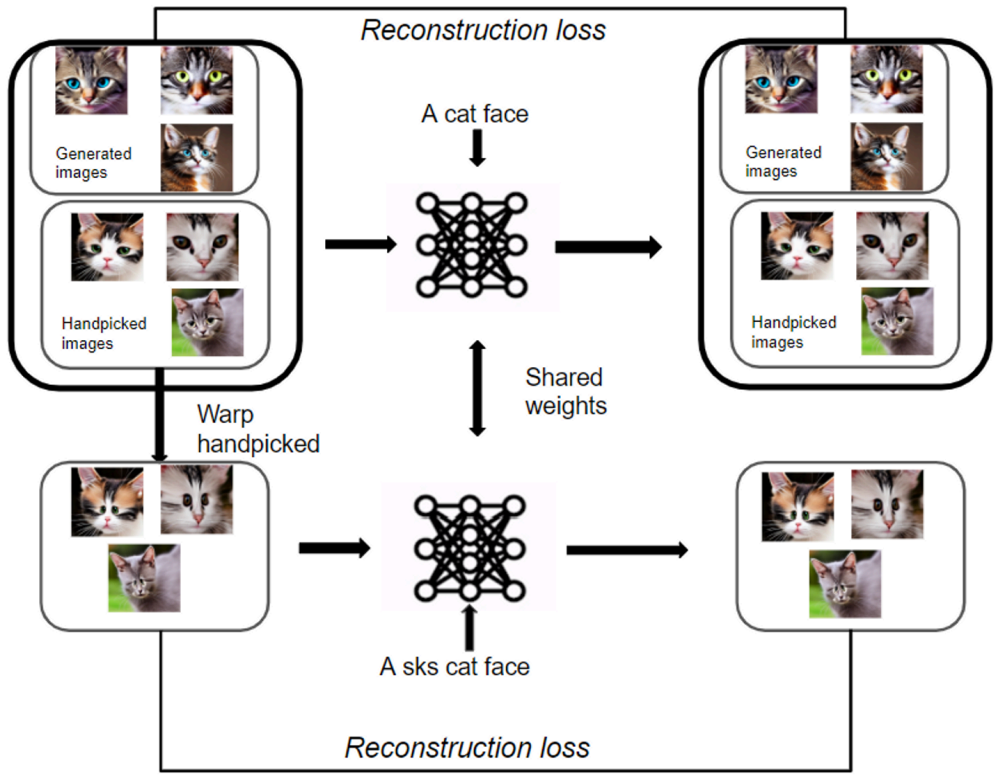


Learnings:

- How to use pytorch lightning and distributed training to train models.
- Working with both CompVis and Diffusers libraries and being able to write training and validation code that works the same on both
- Writing new models wiith new architectures in both pytorch lightning and diffusers syntax and getting them to train correctly
- Working with Hugging Face datasets and models (like instruct pix2pix and the ip2p dataset) as well as implementing low rank adapters for fine tuning

First Project that I tried working on for Generative Intelligence Lab: Learning a warp from a few examples of warping

- Strategy 1 (Dreambooth strategy). Warp a handpicked selection of cats, and train a dreambooth model on that selection, with prior preservation loss coming from all different types of cats:

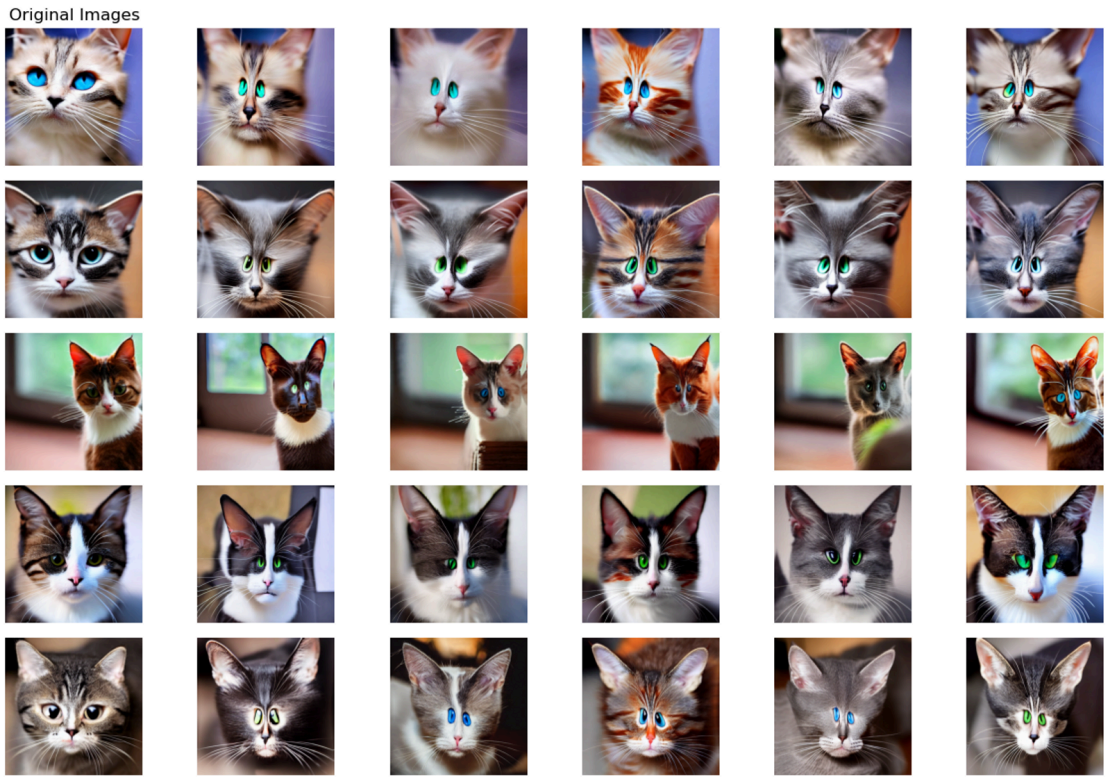


- Dataset Augmentation:
    - Take an image of a cat, use SDEdit to change the fur color, then apply a single warp to all of the SDEdited images

The original cat and the original warp application

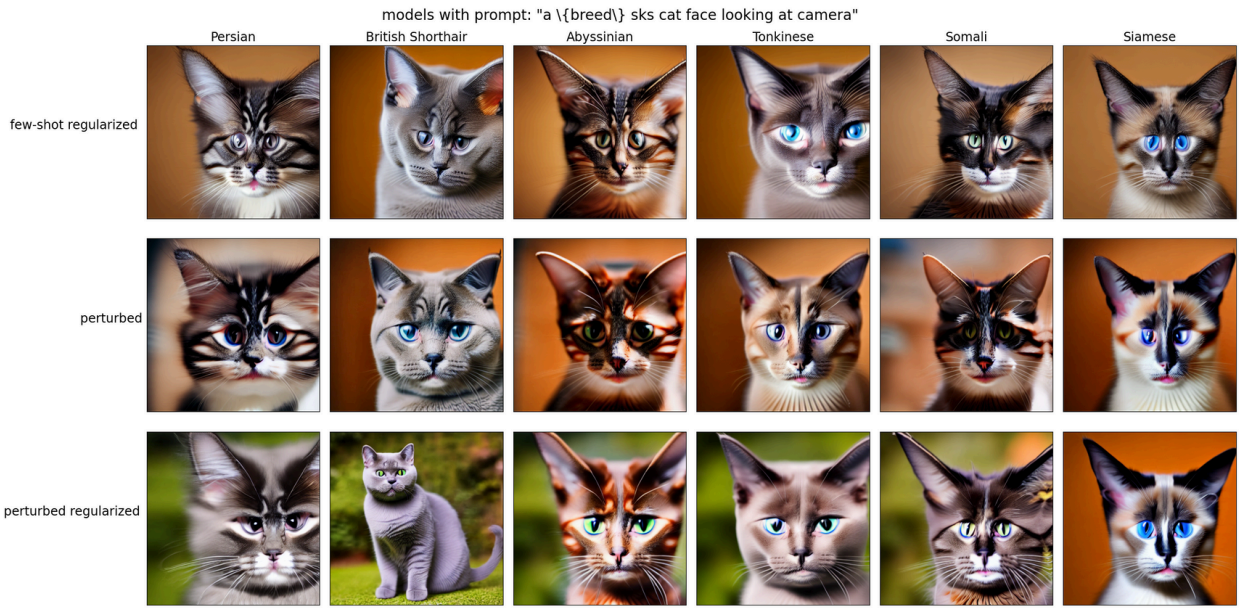
Warped + SDEdited Cats. First SDEdit is applied to the initial cat, then the warp is applied to the resulting images

Problem 1: SDEdit cannot get the eyes to stay in the exact same position, leading to warping issues:



Notice how the warped eyes in columns 3-6 have slight asymmetries as compared to column 1. Training with this data leads to bad warping output from the model:

Problem 2: The edit does not propagate to cats in scenes other than the cat face:



Notice how in all 3 rows, none of the cats have a very warped face (these 3 rows correspond to slightly different training strategies)

Learning through phase 1:

1. How to run training for diffusion models on GPUs and test their results
2. How to run different tests with different paper results (Rewritting Geometric Rules of A GAN, Dreambooth, Custom Diffusion)
3. Creating figures and data visualizations

Developed data perturbation training/evaluating/testing pipeline in Python, leveraging Pytorch for main testing - Tested probabilistic models and evaluated models on perturbed image data