Conditional Latent Diffusion Models (cLDMs) for High Resolution Sketch to Image Generation

John Foley 22366741

Darragh Grealish 22347666

Project Overview

We want to create high resolution (256 x 256) lifelike images of shoes from sketches. An existing barrier to synthesizing high resolution images is the computation required. Latent Diffusion Models (LDMs) present themselves as a computationally efficient version of Denoising Diffusion Probabilistic Models (DDPMs). LDMs work by compressing high resolution images to smaller latent representations using Variational Auto-Encoders (VAEs).

In this project we train a conditional LDM, first using a well established pre-trained VAE [1, 2], and then using our own custom VAE. Using a pre-trained state of the art VAE allowed us to test our cLDM's U-Net in isolation. A core aspect of our project is finding a suitable approach to condition our sketches.

Dataset and Task Selection

The task we are focusing on is broadly known as style transfer and more specifically in our case sketch-to-image style transfer. We will be using the Sketch2Shoe [2] dataset, pairing a sketch of a shoe to its real life representation at a 256 x 256 resolution rate.

Background Literature

KL Variational Auto-Encoders

Variational auto-encoders are used to encode images to a lower dimensional space while preserving the images structural information. Their advantage over standard auto-encoders is their ability to cluster data in the latent space [1]. KL VAE's use a KL divergence term in the loss to normalise the distributions from which the encoder samples. VAEs have a wide variety of use cases, including data augmentation, however an area with recent success is their use in LDMs.

LDMs use VAEs to apply DDPM in a latent space. Note it is the ability to cluster data in the latent dimension that makes VAEs well suited, and standard auto-encoders do not work as well.

VQ Variational Auto-Encoders

Vector quantized variational auto-encoders (VQ-VAEs) present themselves as an alternative solution to KL-VAEs. KL-VAEs describe a continuous latent space, while VQ-VAEs use a discrete latent representation. In VQ-VAEs, the encoder can only use values from a codebook [5]. Note that VQ-VAEs produce sharper reconstructions due to their discrete latent representation. KL-VAEs prioritise smoothness between pixels, which can sometimes result in blurry reconstructions.

For our implementation, we have decided to use a Stable Diffusion KL-VAE [1]. We chose this auto-encoder as KL-VAEs are preferred for LDMs in practice. Their continuous latent space loses less information and achieves a better FID score [1, 2]
Talk about choosing one [1] VAE or the other [4]

Latent Diffusion Models

Latent diffusion models (LDMs) present themselves as a computationally efficient approach of DDPM [1]. An existing issue with DDPM, particularly with high-resolution images, is the computation required. LDMs solve this issue by applying DDPM in a lower dimensional latent space. The DDPM learns to diffuse a latent representation, which can then be decoded by the VAE. LDMs are of particular interest to us as our images are high-resolution. Applying DDPM in 256 by 256 resolution on limited computational resources presents a substantial risk to us, therefore we decided to use LDMs.

Conditional Diffusion Models

Conditioning has become the standard approach for guiding diffusion models to produce specific outputs. Conditioning can take place in many forms such as text, class and in our case sketch conditioning. Previous approaches outline conditioning of text via vector embeddings [6,7] using pretrained encoders such as CLIP [6].

A core aspect of this project is to find a conditioning approach well suited for sketches. Previous approaches apply feature wise linear modulation (FiLM), however images are often best suited to spatial conditioning [8]. In our implementation we consider both FiLM along with a number of different spatial approaches.

Implementation Detail

Downsampling Sketch

One of our initial conditioning approaches was to simply concatenate a downsampled sketch to our latent in the U-net. Our sketches are initially 256x256 in resolution, however to condition them in the latent space, we require them to be 32x32. Note that downsampling the sketches using a VAE risks the potential of losing fine grain sketch details. Our initial approach was to downsample the sketch with a MinPool. Note we use a MinPool over a MaxPool as our sketch lines are black. Figure 1 below shows four different min pooling approaches used. The first is a single min pool with an 8x8 min pool and stride of 8, the second is a 4x4 min pool with a stride of 4 applied 2 times, and finally we use a 2x2 min pool with a stride of 2 applied 4 times. Figure 1 below shows how the 8x8 pool produces the best results, although still blocky.

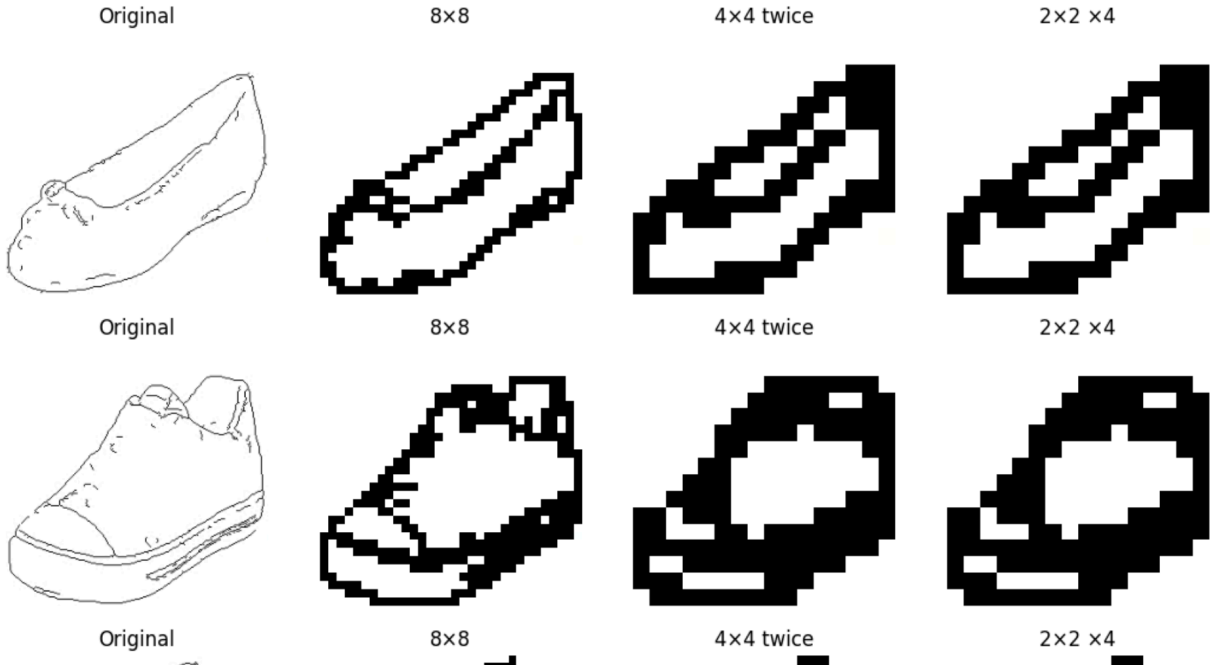


Figure 1: Min Pool Downsampling Approaches

Note that this was one approach used for downsampling, however in the conditioning approaches section below we discuss more complex and performant approaches.

Conditioning Approaches

FiLM Conditioning

The first approach used for conditioning the sketch was FiLM via a MLP. However, the results were poor. The MLP could not learn the spatial dimensions of the sketch, and the resulting diffused images were merely blobs of blackness.

Concatenated Spatial Conditioning

Our next approach was to use a downsampled sketch and concatenate it to the noised latent when it first entered the U-net. This approach demonstrated a significant improvement, and was the first time our model showed significant results. After just 20 epochs, the model could diffuse sketches. However one issue was that the downsampled sketch lost features and was blockier. The model still struggled to understand subtle features like shoe laces. Figure 2 below shows inference results using this spatial conditioning approach on unseen sketches. A key takeaway from this was the success of spatial conditioning on such little epochs.

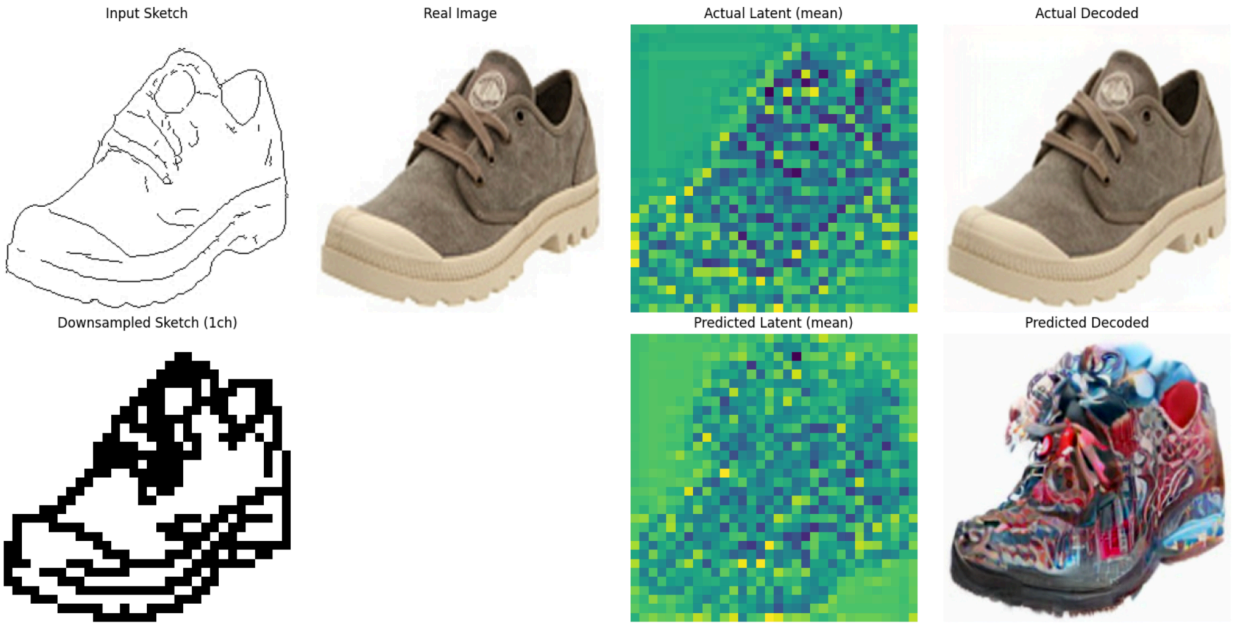


Figure 2: Diffused Image Using Downsampled Concatenation

Multi Scale Conditioning Via Sketch Encoder

The issue with concatenating the downsampled image was it lost features like shoelaces. We needed a way to downsample the sketch to 32x32 while preserving the details of the sketch. Figure 3 below shows our sketch encoder, which learns to encode a sketch into multiple dimension spaces. In order to reinforce the input sketch, we apply multi-scale conditioning. This means adding an add-layer of conditioning at every stage. Figure 3 below details our sketch encoder learning to encode a sketch at different resolutions.

```python
class SketchEncoder(nn.Module):
    """Processes 256x256 sketch into multi-scale feature maps."""
    def __init__(self, base_ch=32):
        super().__init__()
        # 256 -> 128 -> 64 -> 32
        self.enc1 = nn.Sequential(
            nn.Conv2d(1, base_ch, 3, stride=2, padding=1), nn.SiLU(),
            nn.Conv2d(base_ch, base_ch, 3, stride=2, padding=1), nn.SiLU(),
            nn.Conv2d(base_ch, base_ch, 3, stride=2, padding=1), nn.SiLU()
        )
        # 32 -> 16
        self.enc2 = nn.Sequential(
            nn.Conv2d(base_ch, base_ch * 2, 3, stride=2, padding=1), nn.SiLU()
        )
        # 16 -> 8
        self.enc3 = nn.Sequential(
            nn.Conv2d(base_ch * 2, base_ch * 4, 3, stride=2, padding=1), nn.SiLU()
        )

    def forward(self, x):
        feat32 = self.enc1(x)    # [B, 32, 32, 32]
        feat16 = self.enc2(feat32) # [B, 64, 16, 16]
        feat8  = self.enc3(feat16) # [B, 128, 8, 8]
        return feat32, feat16, feat8
```

Figure 3: Sketch Encoder

The results of the multi-scale conditioning can be seen in figure 4. Even after a small number of epochs, the sketch encoder learns to encode a sketch. The generated show captured more fine grain details from the sketch in comparison to concatenating the downsampled sketch.

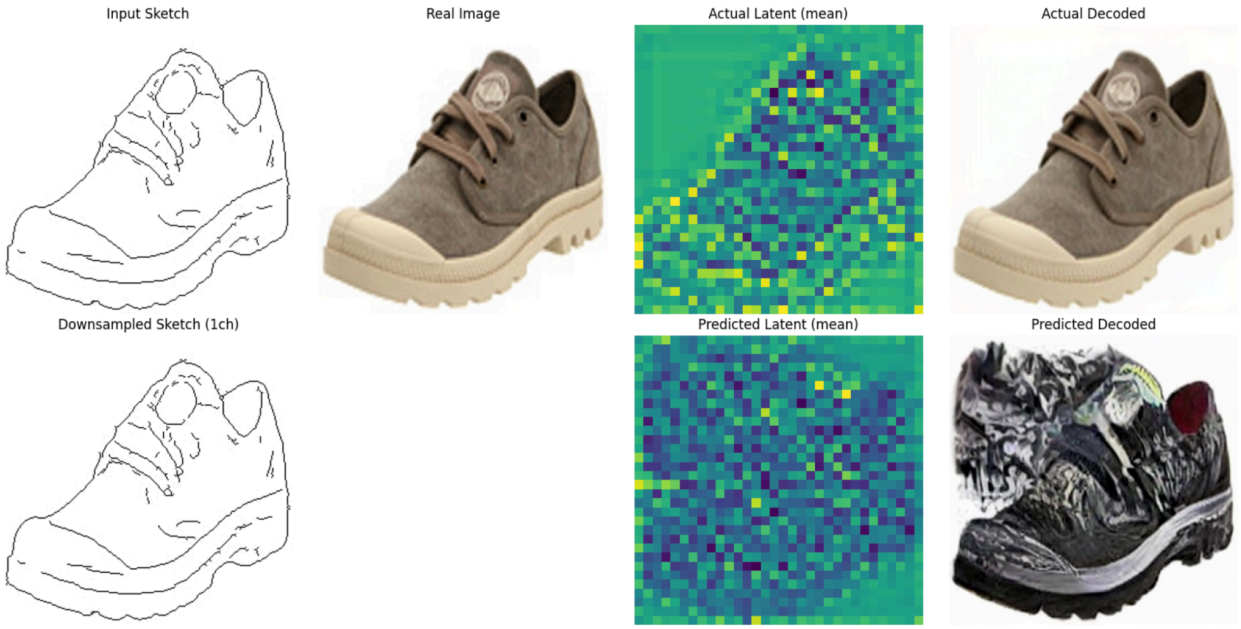


Figure 4: Diffused Image Using Multi-Scale Conditioning

Improvements

The results from figure 4 show significant results, however there is still a high degree of noise present. The following changes were made to our final model to the one creating the images in figure 4.

Dropout

We apply dropout in our training loop as a form of regularization. We implement dropout by removing the entire sketch using a dropout rate of 10%. The model learns robustness and is not dependent on the sketch feature. It also means that our model also learns unconditional diffusion, and can generate a real image without a sketch.

Increasing Timesteps

Another change that saw possibly the most substantial improvements of any was the increase of our timesteps from 200 to 800. A clear issue with the diffused images in figure 2 and 4 were they still have a significant amount of noise. By increasing the timesteps, our model learns a more stable diffusion process and the output image has no noise. Figure 5 below shows the resulting diffused images having implemented dropout and increasing the timesteps.

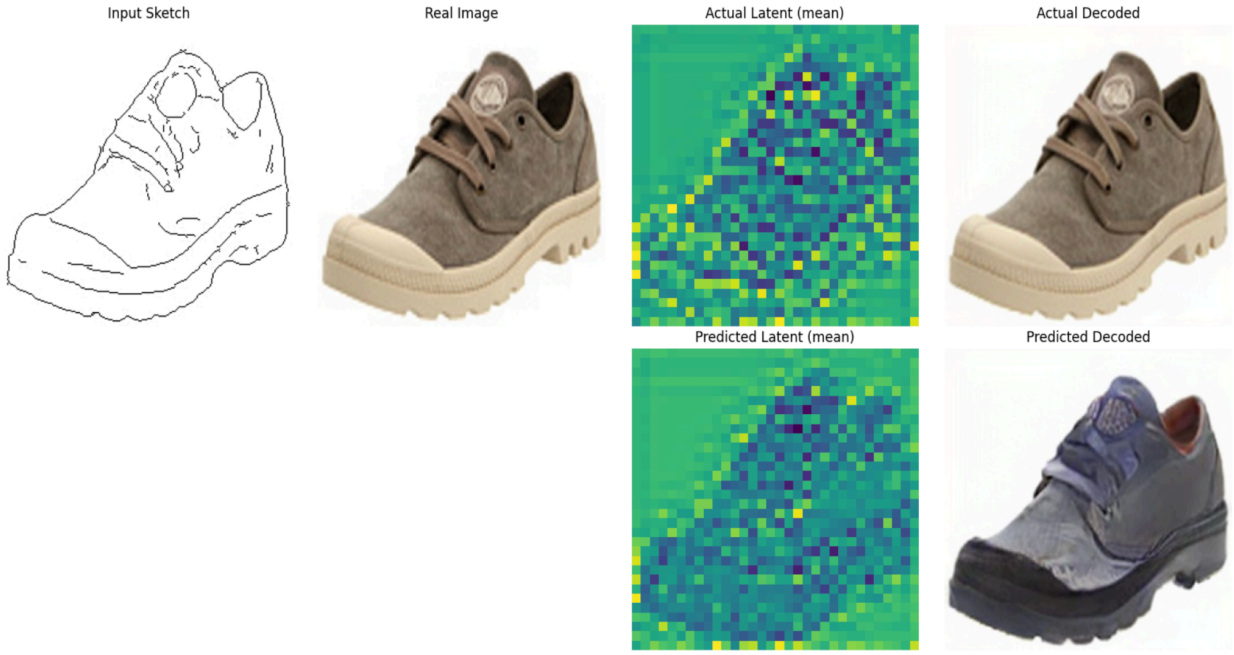


Figure 5: Increased Timesteps and Implemented Dropout

Architecture

KL-VAE Architecture

The initial design of the KL-VAE used three stride-2 convolutions to compress the 3x256x256 shoe images down to a 4x32x32 latent space. This was followed by a symmetric stack of three transposed convolutions, to reconstruct back to the 256x256. This model only took a few minutes per epoch to train, but the reconstructions were noticeably blurry, the shoe colours weren't being used and the FID score was very high. Looking into the setup we had we seen the problem was surrounding the encoder & decoder being too shallow and poorly regularised.

We then implemented the architecture to help improve the FID. First we replaced the single convolution stages, implementing our encoder and decoder using GroupNorm and SiLU. This gave the network more depth and better normalization, which stabilised training and allowed it to capture sharper edges and textures. Secondly, we swapped out the transpose convolutions in the decoder for nearest-neighbor upsampling followed by regular convolutions. Finally, we modified the loss function from using MSE+KL to using a combination of L1 and SSIM plus a KL term with a tunable weight beta. The VAE was tricking the MSE by giving it overly smooth outputs to get a better MSE score. Using L1 & SSIM helped us stop this as it preserved structural details that matter for FID and for the downstream diffusion. After implementing this change, we noticed the visual reconstructions went from soft and noisy, to having sharper features similar to the original shoes.

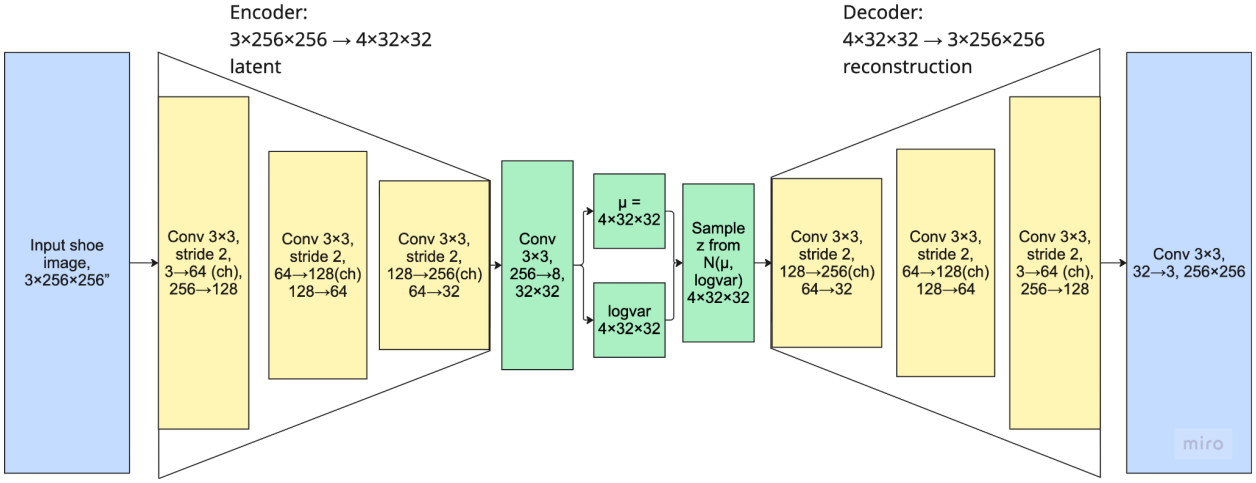


Figure 6: KL-VAE Architecture

U-Net Architecture

Our latent U-Net is kept small with roughly 1.8M parameters, yet it operates at an effective output resolution of 256×256. This is possible because the model never works directly in pixel space, but it predicts noise in a 4×32×32 VAE latent space, and the heavy lifting of decoding back to 256×256 RGB is handled by the VAE. Working in latent space reduces both memory

and the computer per image drastically compared to a 256×256 U-Net with similar depth, while still capturing high-level structures of the shoes. The three-level encoder–decoder (32→16→8) with residual blocks and skip connections gives us enough capacity to model complex shoe textures and shapes, without needing a lot of compute power or having a massive parameter count.

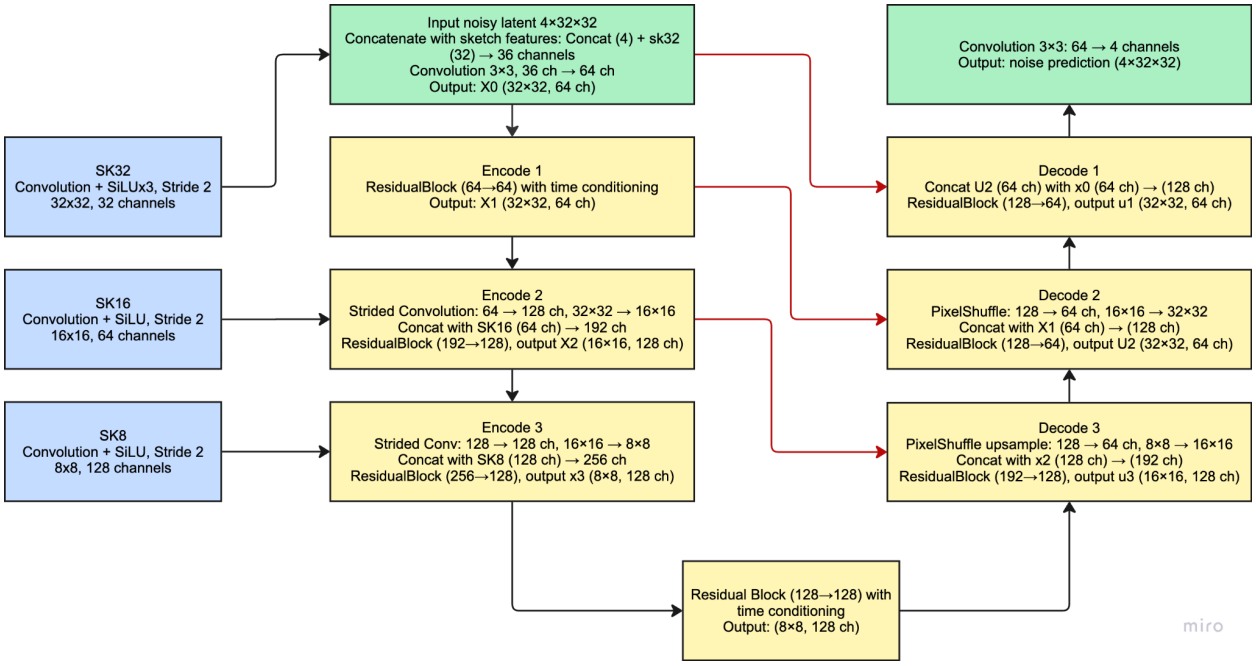


Figure 7: LDM Encoder and Decoder With Multi Scale Conditioning

Inference

Inference in LDMs consists of a randomly noised latent and a sketch. In our training loop, we teach our LDM to predict noise at different timesteps. We train the model on one timestep per image. However during inference, we iteratively remove noise starting at timestep 800 and working our way down to 0.

Inference requires the LDM U-net and the VAE decoder, the VAE encoder is redundant at this stage. Once the U-net predicts noise, we recursively remove the noise from our noised latent until arriving at the denoised latent. At this stage, the denoised latent has dimensions (4, 32, 32) and we use the decoder to upscale it to the higher-resolution 256x256 space.

Results

Custom VAE FID

The initial design of the KL-VAE used three stride-2 convolutions to compress the shoe images down into the latent space. This was followed by a symmetric stack of three transposed convolutions, to reconstruct back to the original size. This model only took a 3.5 minutes per epoch to train. We trained the model for 20 epochs, but the reconstructions were noticeably blurry, the shoe colours were being lost and the FID score was very high ~350. Looking into the setup we had we seen the problem was surrounding the encoder & decoder being too shallow and poorly regularised.

Figure 8: Shoe reconstruction on initial KL-VAE design (20 epochs, FID-350)

We replaced the single convolution stages in the encoder and decoder with deeper blocks using GroupNorm and SiLU activations, which improved normalization and allowed the network to learn sharper edges and textures. We swapped the decoder's transposed convolutions for a combination of nearest-neighbour upsampling and standard convolutions to reduce checkerboard artifacts. Finally, we changed the reconstruction objective from MSE + KL to a combination of L1 and SSIM, and a tunable weight Beta, which discouraged overly smooth outputs (which tricked the MSE into giving better scores) and encouraged preservation of structural details. Together, these changes brought the FID down from ~350 to 74.8 after 50 epochs, and the reconstructions went from soft and noisy to visually much closer to the original shoes.

Figure 9: Shoe reconstruction KL-VAE (50 epochs, FID-74.8)

Figure 10: KL-VAE Reconstruction and Latent

LDM U-Net

Our cLDM demonstrates high quality sketch to image at high resolution. Using a pre-trained VAE such as Stable Diffusion allows us to focus entirely on the performance of the U-net. As shown in figure 11 below, the u-net successfully captures the structural dimension of the sketch to produce a clear latent image.

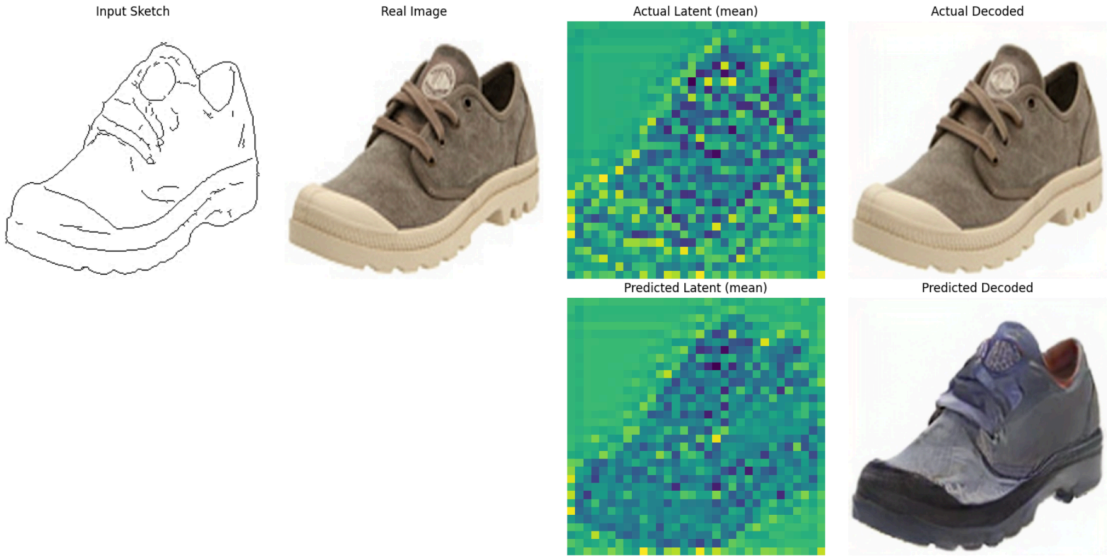


Figure 11: LDM Predicted Latent

Figure 12 below shows the generated image from the same sketch for a second time. What we notice is that the structural integrity of the shoes remains, while coloring and characteristics not described in the image are subject to change. The structural integrity of a shoe is a telling characteristic of our successful conditioning.

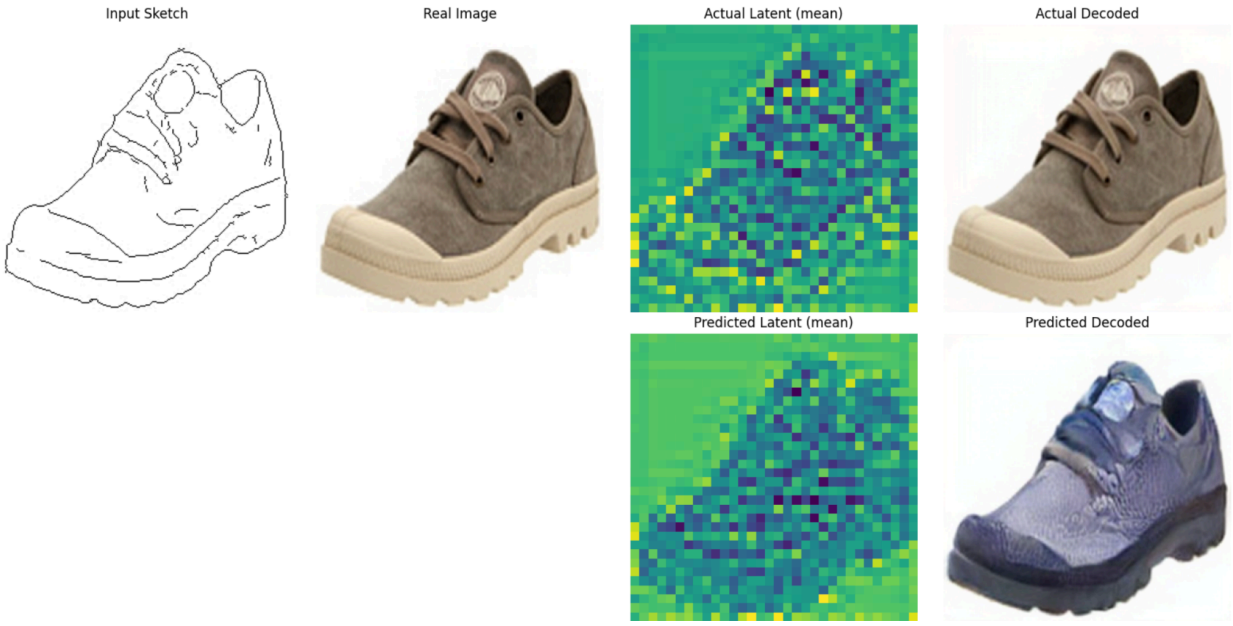
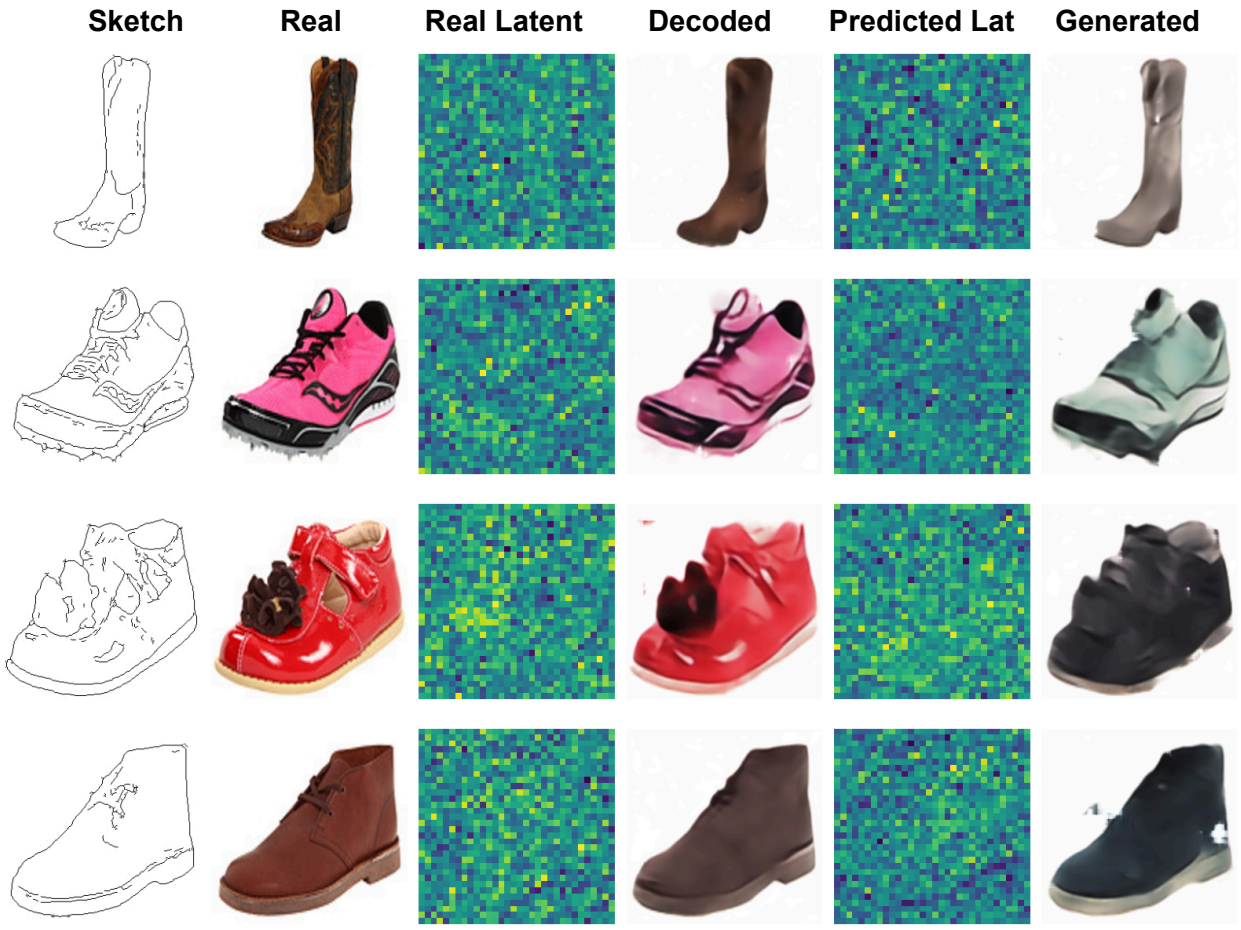


Figure 12: LDM Second Predicted Latent

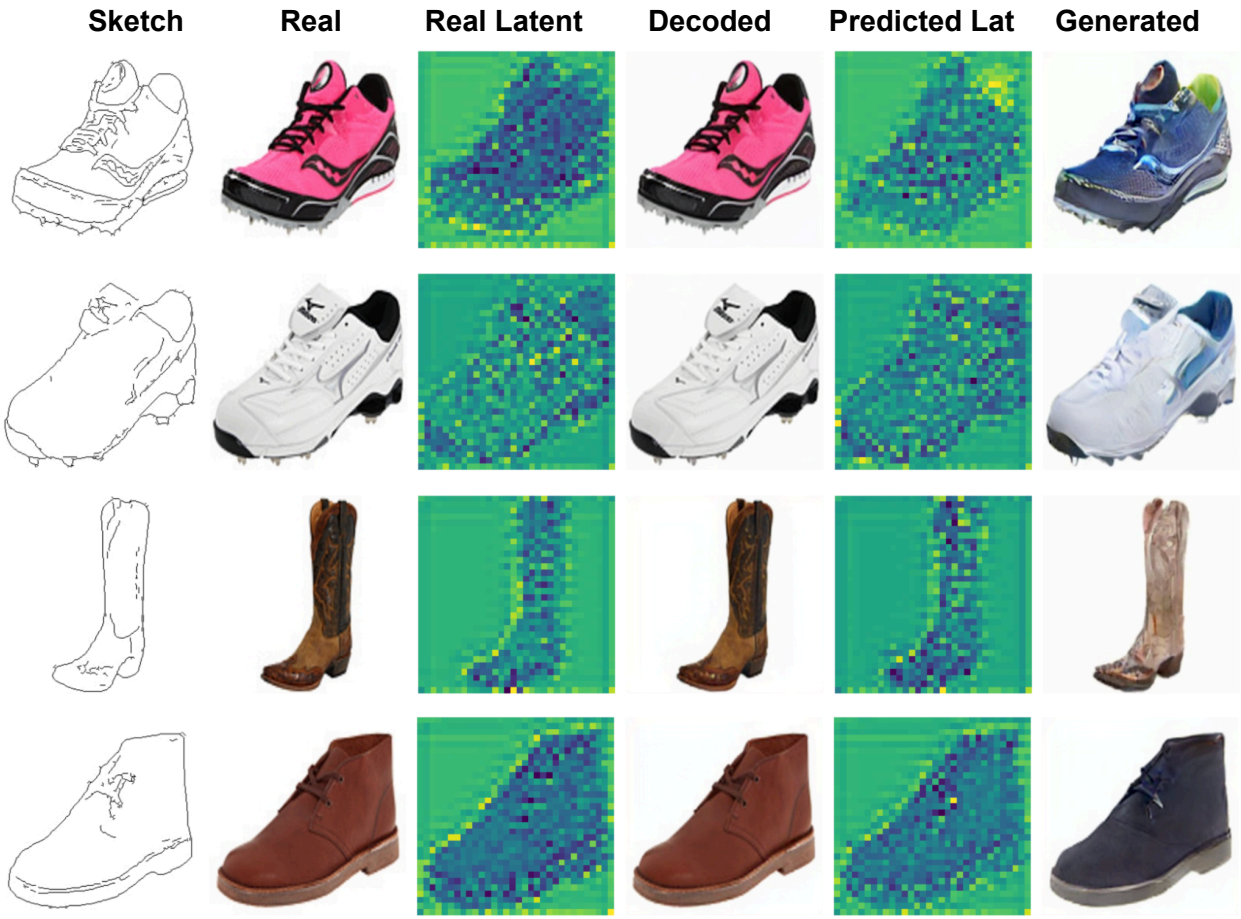
With regard to an FID score, our U-net trained with the pre-trained VAE generated images with an FID of 131. Despite producing high resolution images on the eye, this FID score would be considered quite high. A potential reason for this high FID is that the sketches only maintain the structural integrity but the model is open to color changes.

In comparison with this, our U-net trained with our custom VAE received an FID of 164. This value demonstrates that our VAE performs quite well on this metric. Having said this, looking at the images below with our custom VAE, we can see the KL-VAE struggles with sharpness. We attribute this primarily to the model's shallow architecture and lack of training epochs.

Example Generations With Custom VAE



Example Generations With Pre-trained VAE



References

1. Rombach, Robin, et al. “High-Resolution Image Synthesis with Latent Diffusion Models.” arXiv:2112.10752, arXiv, 13 Apr. 2022. *arXiv.org*, https://doi.org/10.48550/arXiv.2112.10752.
2. Blattmann, Andreas, et al. “Retrieval-Augmented Diffusion Models.” arXiv, 2022, https://doi.org/10.48550/ARXIV.2204.11824.
3. Isola, Phillip, et al. “Image-to-Image Translation with Conditional Adversarial Networks.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–34.
4. Subramanian, Anand K. *AntixK/PyTorch-VAE*. 10 Jan. 2020, Python. 22 Nov. 2025. *GitHub*, https://github.com/AntixK/PyTorch-VAE.
5. van den Oord, Aaron, et al. “Neural Discrete Representation Learning.” *Advances in Neural Information Processing Systems*, vol. 30, 2017. *Neural Information Processing Systems*, https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.
6. Saharia, Chitwan, et al. “Palette: Image-to-Image Diffusion Models.” *CoRR*, abs/2111.05826, 2021, https://arxiv.org/abs/2111.05826.
7. Nichol, Alex, et al. “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models.” arXiv:2112.10741, arXiv, 8 Mar. 2022. *arXiv.org*, https://doi.org/10.48550/arXiv.2112.10741.
8. Peng, Yichen, et al. “Sketch-Guided Latent Diffusion Model for High-Fidelity Face Image Synthesis.” *IEEE Access*, vol. 12, 2024, pp. 5770–80, https://doi.org/10.1109/ACCESS.2023.3346408.
9. Esser, Patrick, et al. “Taming Transformers for High-Resolution Image Synthesis.” 2021, pp. 12873–83. *openaccess.thecvf.com*, https://openaccess.thecvf.com/content/CVPR2021/html/Esser_Taming_Transformers_for_High-Resolution_Image_Synthesis_CVPR_2021_paper.html?ref=