Generative CLIP-Conditioned Variational Autoencoder

Ariana Azarbal, Ayman Benjelloun Touimi, Sofia Tazi

Abstract:

In recent years, the intersection between language and imagery has garnered increasing attention, particularly with the development of CLIP by OpenAI. The CLIP model innovatively maps both text and image data into a shared latent space, opening new avenues for multimodal research. Instead of employing a computationally intensive diffusion model, our group has integrated CLIP embeddings with a Variational Autoencoder (VAE). We leverage well-established techniques for image-generation using Conditional Variational Autoencoders, yet we explore conditioning the model with CLIP's semantically rich textual/visual output. We perform experiments conditioning with both image embeddings and text embeddings. We show that the model is able to generate images which are sensitive to the generation prompt when conditioned on either text or image encodings during training. But, surprisingly, the model does best when conditioned only on image encodings, even when prompted with the caption's text encoding at generation time. This reflects and confirms the incredibly rich latent space of CLIP.

Introduction:

Contrastive Language Image Pretraining (CLIP)[1] opens up vast possibilities for multimodal learning by producing an aligned representation of text and image data in a shared latent space. Not only has this model demonstrated impressive zero-shot capabilities, but it has also been incorporated as an effective image encoder in Vision-Language Models (VLMs), which perform tasks such as visual question and answering, object detection, etc.

In the context of diffusion model-based image generation, the paper "Hierarchical Text-Conditional Image Generation with CLIP Latents" [2] introduces a method of conditioning on CLIP embeddings. Given the computational resource constraints of our group, we have chosen to integrate CLIP embeddings with a Variational Autoencoder (VAE). This approach allows us to explore image generation without the computational overhead associated with diffusion models.
We experimented with various image sizes, architectures, and CLIP encoding modes (image embeddings vs. text embeddings) leveraging CLIP's encoding of semantic and structural visual/textual information.

Goals:

Our base and stretch goals underwent two distinct versions as we realistically discovered our computational capacity and what was achievable within our allotted time. We initially set unrealistic goals, with a base goal aimed at reproducing the diffusion model on which unClip is based, a target goal of conditioning the image using CLIP, and a stretch goal of influencing the image created with a second prompt afterward.
Further research led us to understand that those goals were unreasonable given our computational resources, which prompted us to reconsider the scope of our project and set new, more realistic goals.
Our final base goal was to generate an image using a CVAE that was more than pure noise, and that presented a comprehensive structure picturing a bucolic landscape. Our target goal was to detect a semantic influence of the given caption when generating an image, while our stretch goal was to slightly shift away from the bucolic theme, expanding the dataset to include other themes to see if we could still achieve a recognizable structure.

Dataset:

We chose to use MS COCO[4] to train our model which is commonly used for studying the relationship between image captions and the images themselves. Given that MS COCO captures a wide range of images and contexts, using the entire dataset as-is was neither feasible nor ideal for our specific goals. Consequently, we chose to refine a subset of the MS COCO and specifically parsed the dataset for bucolic and natural themes, such as "cow," "sheep," "grass," and "mountains." This approach was taken to emphasize the vivid color contrasts typically found in pastoral scenes—like the stark white of snow or the rich green of grass. By focusing on such themes, we aimed to explore the distinct visual patterns and color differentiations that could provide deeper insights into the semantic understanding of our model.
The images were formatted in either 64x64 or 128x128 resolutions to maintain a balance between detail and computational efficiency, enhancing the model's ability to differentiate and generate based on the subtle distinctions between these elements.

For training purposes, we utilized two distinct datasets. The first, a smaller dataset containing 3400 images, was exclusively parsed for images tagged with "cow" and "sheep," and served as the initial training ground for all types of models we explored. This allowed us to focus on fine-tuning the models on a smaller set of images.
The second, a larger dataset, included a broader range of tags such as "cow," "sheep," "mountain," "hill," "countryside," "grass," "forest," "farm," "alpaca," "horse," "landscape," and "fence." This dataset contained about 20,000 images and was used to train the most promising architecture. This approach allowed us to scale up our training as we moved from a narrow focus to a more comprehensive set of natural and pastoral scenes, enhancing the model's ability to generalize across a wider variety of images. Examples of the images used in both datasets can be found in Figure 1.

Small dataset	
Large dataset	

Figure 1. 4 random samples from the two datasets used

Methodology: Architecture

I. Process

In order to achieve our goals, we initially focused on identifying an architectural complexity that would align with our objectives while remaining within our computational capabilities. We started with a basic Variational Autoencoder (VAE) model inspired by a GitHub repository found [3]. In our first architecture configuration, we opted not to flatten the output of the encoder. The decoder was composed of three Conv2DTranspose layers and one final Conv2D layer. However, due to the consistent noise in the model output and inadequate training, we concluded that the initial complexity was not sufficient.

We then explored ways to increase the complexity of both components:
- **Encoder**: We added dense layers, flattened the inputs, and experimented with leaky ReLUs.
- **Decoder**: We adjusted the number of Conv2DTranspose layers and batch normalization processes, occasionally adding or removing a Conv2D layer at the end.

Conditioning the Model: We also experimented with various methods of conditioning the model with embeddings. Initially, we simply concatenated the embeddings to the output of the encoder before computing Z for the VAE's latent space. However, this approach proved inconclusive. We reasoned that if we wanted the embeddings to be the sole input for our model, it would be more logical to

concatenate them with Z before the decoding process. Consequently, we adopted a dual concatenation approach: once at the encoder to condition the VAE latent space, and again before the decoder to utilize the raw information during decoding.

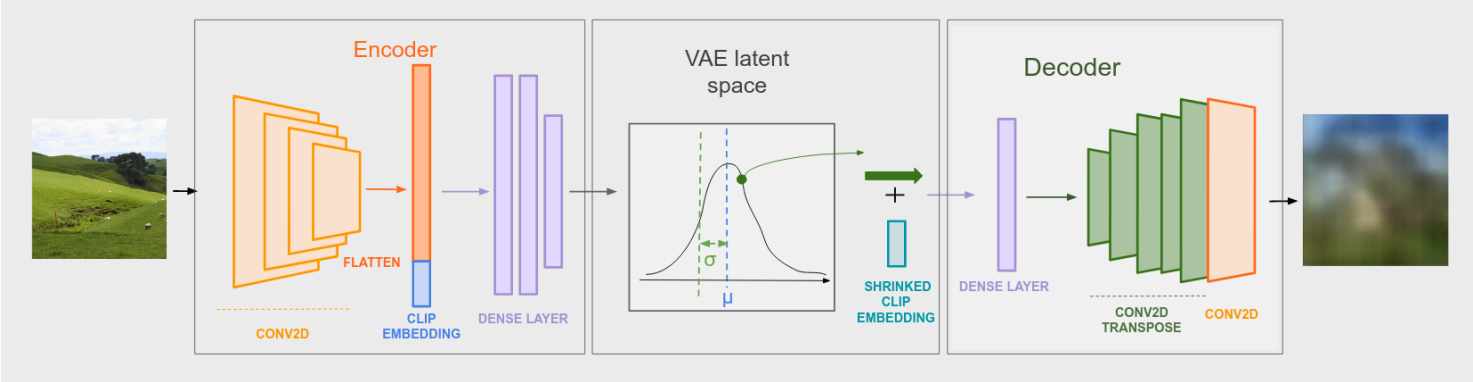
Exploring these elements enabled us to settle on an architecture with sufficient complexity to discern a clear structure in the generated images. However, we found that increased complexity often led to less stable results, higher losses, or configurations that were unmanageable on our computers.

Below, we have illustrated the visual evolution of the outputs from our architecture development.

Complexifying VAE until finding structure

Complexifying a CVAE with caption “cow in the grass with blue sky”

II. Final Architecture



For this project, we have experimented with multiple architectures of Conditional Variational Autoencoders (CVAE). Simply put, a CVAE is a type of generative model extending the regular VAE framework with conditional capabilities. It is composed of two main parts: an encoder and a decoder. The encoder uses image data, along with CLIP image or text embeddings, and reduces the data to a smaller latent representation consisting of parameters of a probability distribution representing the

data. The decoder, on the other hand, uses a point randomly sampled from this distribution to reconstruct the original image.

Our **encoder** consists of two parts:

- The first part is composed of 4 convolutional layers with a decreasing number of filters (256 3x3, 256 5x5, 128 3x3, 128 3x3), strides of 2, same padding, and Kaiming-initialized weights. Batch Normalization is applied to the output of each layer and activated with LeakyReLU. It is designed to receive the image data as an input.
- The second part has three fully connected layers (output sizes: 2048, 2048, 1024), activated using LeakyReLU with Batch Normalization. Its input is a latent representation of the image, concatenated with a shrunk CLIP embedding, obtained by passing it through a Dense layer with output size 32.

Between our encoder and decoder, two Dense layers with output size 128 and Kaiming-initialized weights compute two vectors, mu, and sigma.

Our **decoder** consists of:

- A Dense layer with output size 16,384, followed by Batch Normalization, activated using LeakyReLU and Kaiming-initialized weights.
- There are five 2D Transpose Convolutional layers, with a decreasing number of filters (256, 256, 128, 128, 32), same padding, and strides varying from 1 to 2. Batch Normalization is applied to the output of each layer and activated with LeakyReLU.
- A final convolutional layer, with 3 5x5 filters, with same padding, Kaiming-initialized weights, and sigmoid-activated, to scale the image back to its original dimension

The **loss** chosen is expressed below:

$$L = L_{BCE}(x, \hat{x}) + 0.5D_{KL}(N(\mu, \sigma), N(0, 1)) + 0.1(1 - SSIM(x, \hat{x}))$$

Where :

$$L_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left(x_i \cdot \log(\hat{x}_i) + (1 - x_i) \cdot \log(1 - \hat{x}_i)\right)$$

$$D_{KL}(N(\mu, \sigma) \| N(0, 1)) = \frac{1}{2}\left(\sigma^2 + \mu^2 - 1 - \log(\sigma^2)\right)$$

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

The function combines Binary Cross Entropy (BCE) to determine reconstruction loss, KL-divergence, and Structural Similarity. In the formula, the BCE ensures that the pixels are accurately reconstructed, KL-divergence allows the latent spaces to encode generalizable representations of the data, and SSIM enhances the quality of the reconstruction, making sure that the structural integrity of the output matches the input. Note that both D_{kl}, and $(1 - SSIM(x, \hat{x}))$ are multiplied by two weights (0.5 and 0.1, respectively). These weights happened to give us the best results and were left as tunable hyperparameters

Results:

With the final architecture described above as a basis, we tried different variations in order to find the most semantically influenced model. Those training were all made on the smaller data set and compared in order to determine the most effective tuning. The result's insights can be seen in Table 2 and found in Table 1. Note that for all of these, our batch size was 16 or 32.

Model	Best Validation Loss	Qualitative Observations	Number in Table 2
64x64 Im. Embed-Conditioned	2376 (epoch 25)	Images are highly pixelated and blurry, but good with high-level features!	[1]
64x64 Text Embed-Conditioned	2526 (epoch 8)	Poor. Unable to represent colors or basic shapes well.	[2]
128x128 Im. Embed-Conditioned	9227 (epoch 25)	Best model, qualitatively Effective with colors, shapes, features like "sky", "grass", "sheep".	[3]
128x128 Text Embed-Conditioned	9701 (epoch 9)	Poor. Unable to represent colors or basic shapes well.	[4]
128x128 Im Embed-Conditioned (No Dropout)	9386 (epoch 10)	Effective at high-level features. Blurry.	[5]

Figure 2: Qualitative textual description of the results obtained from training on the smaller data set with different design characteristics

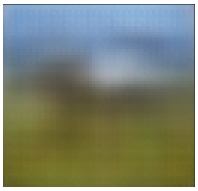
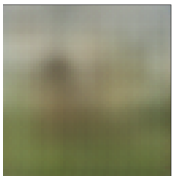
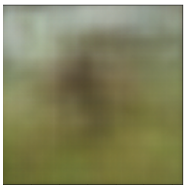
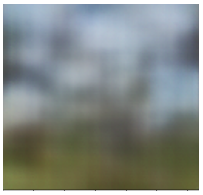
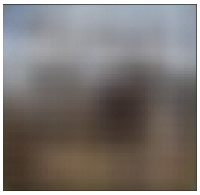
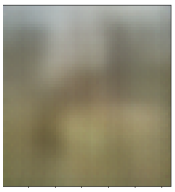
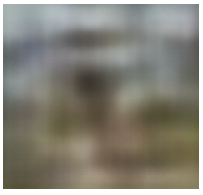
Captions	[1]	[2]	[3]	[4]	[5]
"Cow in grass with sky"					
"Cow in mountain with sky"					

Figure 3: Images generated for two chosen captions in the 5 different versions of the architecture described in Figure 2

Given those first results, we figured that the model that trained with image embeddings and with 128x128 images was the one that demonstrated the sharpest shapes and featured the most coherent colors. We consequently decided to train this model on the largest data set in order to explore the generation of our model.

As we can see in Figure 4, the train and the validation were pretty similar across the epochs showing that we did not reach a point of overfitting. However, even though the loss was still slightly decreasing, it converged early in the training. It is worth noting that this plateau was the smallest one we reached after tuning the hyperparameters.

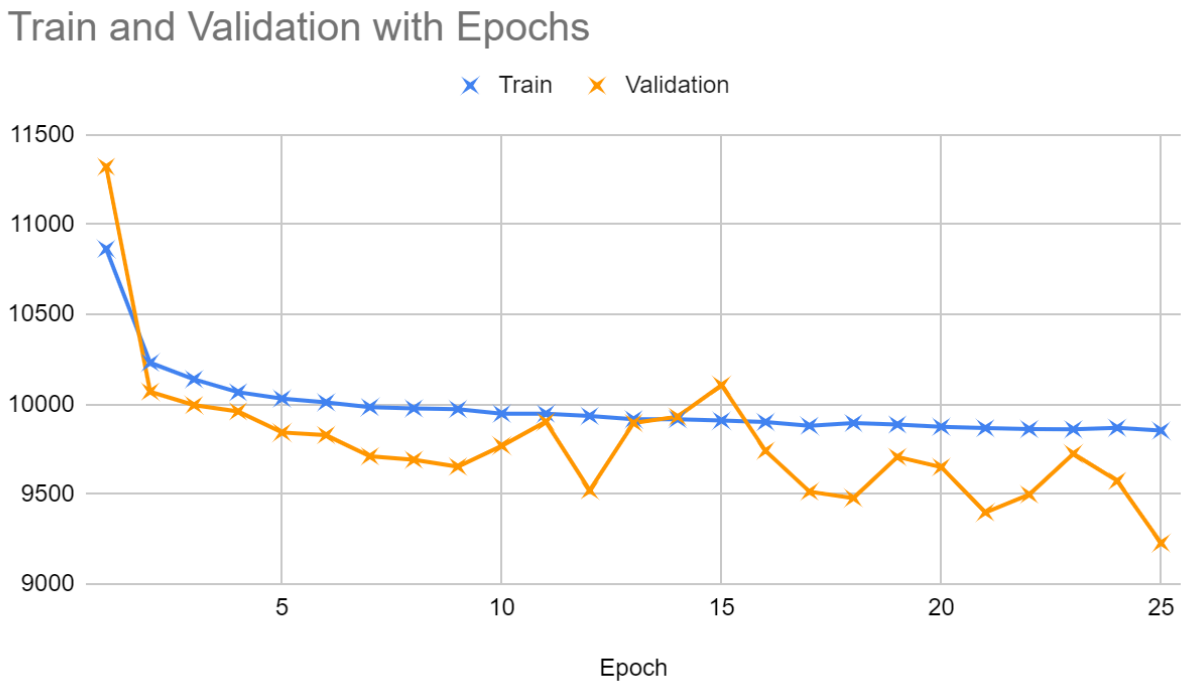


Figure 4. Train and Validation loss of the final model across 25 epochs

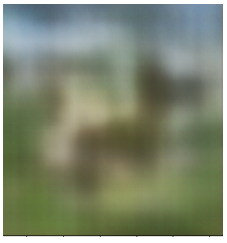
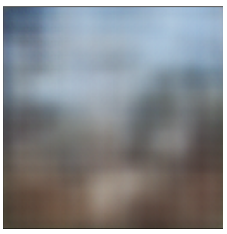
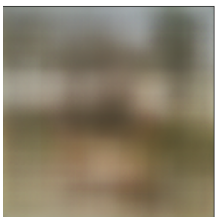
				
Cow in grass with blue sky	*Cow in grass*	*Sheep in snow*	*White sheep on a mountain with blue sky*	*Farm*

Figure 5: Images generated by the final architecture for 5 different captions

We can observe that, although the images are blurry, there are distinct, high-level patterns that emerge. For instance, when "grass" is mentioned, green appears; if "sky" is not mentioned, there is not consistently a blue bar at the top. However, when it is mentioned, a blue bar is consistently present. Additionally, there is a gradient of color depending on the type of landscape: white for snow, maroon for mountains, and so on. This reflects a form of semantic understanding within the model.

Challenges

While carrying out our vision for this project, we faced a variety of difficulties and had to reroute multiple times. Our initial hope was to re-implement unCLIP, which we later realized is DALL-E 2. Thankfully, we soon realized this endeavor likely requires computational resources beyond what we'd have access to for this project. We pivoted to implementing a simplified version of this basic idea, a CLIP-Conditioned VAE, which retained the aspects of this model that we found to be very intriguing—leveraging CLIP's aligned visual and linguistic representations for text-to-image generation.

Still, it took time and effort to tune a model that could generate images that weren't merely noise. As discussed in the process section, using a decoder and encoder with increased complexity but not too much complexity yielded our best results. We also had to strike a balance between working with image sizes capable of representing sharp images yet small enough to allow the model to effectively represent simple, broad features (like a blue sky, green grass, etc.).

We struggled with excessively blurry images throughout the process. In the end, we were not able to generate images that were very sharp at all, but we were satisfied that broad features were represented well and the model is clearly sensitive to changes in the prompt: for example, our effective models did not fail to register color information provided or whether we wanted a "sky" present in the picture.

Reflection :

Project Takeaways :

We believe that our results lend themselves to two main takeaways :

1. *Effectiveness of CLIP embeddings:* CLIP embeddings proved to be a cost-effective method for encoding basic linguistic knowledge into a relatively simple model like a Variational Autoencoder (VAE). This demonstrates the versatility and robustness of CLIP's embeddings for various types of encoding.
2. *Richness of CLIP's latent space:* CLIP's latent space was substantial enough to allow our model to make visual sense of text encodings from prompts, even though it was primarily trained on image encodings. Interestingly, the model showed improved performance when trained solely

on image encodings, suggesting that visual information alone might suffice for this kind of task under certain conditions.

The fact that our model performed well with only visual information from CLIP embeddings is intriguing, as it suggests that the complexity of integrating text and image was not necessary for it to have a solid understanding of the caption. However, it is worth emphasizing that with significantly more resources, and especially a more complex and nuanced model, the importance of the information given by textual data might've been more palpable; a more complex, text-embedding conditioned model could prove itself superior.

Project Outcomes:

We initially expected to have somewhat blurry images, and we believe we generally met our target goals structure-wise. However, we did not have time to explore our stretch goal. Observing how specific features like mountains or grass influenced the color palette of the images (from maroon to green) was particularly fulfilling. More crucially, discovering that the model recognized semantic information with image embeddings alone was a key insight. It allowed us to unexpectedly explore the power of the CLIP latent space, which was the underlying objective of our project. This was honestly thrilling!

Areas of Improvement:

The loss metric plateaued very early (within 7-8 epochs) and was consistently fairly high no matter the complexity we chose (within our computational capacities), leading to persistent blurriness in the images. This indicates a clear limitation in the complexity achievable with our current CVAE architecture. We considered enhancing the model's architecture by incorporating U-nets to tackle blurriness and improve image quality. However, our initial attempts to integrate U-net architecture encountered significant challenges, notably in adapting its typical use in image classification to the demands of image generation. U-nets are predominantly utilized for tasks like segmentation, where the goal is classification within images. Our project required not just recognizing features within images but generating entirely new images based on conditioned inputs from CLIP embeddings. We struggled to devise a method to adapt U-nets for this type of conditional image generation. The main stumbling block was figuring out how to effectively integrate the CLIP embeddings with the U-net architecture to maintain both the conditioning and generative aspects. Due to time constraints, we were unable to resolve these issues fully, but this remains a promising area for future research. With more time, we would love to further explore a modified approach to U-nets that could potentially lower the loss magnitude and lead to clearer, more detailed images.

Another approach we would like to explore further, which Raymond, the class's HTA, suggested at DL Day, is to "reverse" the pipeline: give an image we generated to a model and try to generate a caption from it or classify it. This approach would involve analyzing the semantic accuracy and coherence of

the captions generated from our model's images, providing valuable insights into the model's understanding of the visual content it creates. We believe that exploring this process could be a very interesting area for the future and would've been explored if we had more time.

Code repository :

Our code is accessible on Github at: https://github.com/ayma-n/dl-fp.

Acknowledgements:

We would like to thank our professor Ritambhara Singh, our mentor TA, Michael Lu, and Calvin Luo for giving us tremendous support and guidance. We would also like to thank Willow Baker for kindly and generously printing and giving precious design feedback on our poster!

References:

[1] "Learning Transferable Visual Models From Natural Language Supervision", 26 Feb 2021, Alec Radford, Jong Wool Kim, Chris Hallacy, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark Gretchen Krueger, Ilya Sutskever
[2] "Hierarchical Text Conditional Image Generation with CLIP Latents", Apr 2022, by Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen
[3] GitHub repository, url: https://github.com/MariaPdg/image-autoencoding
[4] "Microsoft COCO: Common objects in context. In Proceedings of the European Conference on Computer Vision (ECCV)", Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014).