Redefining VAE Potential: Very Deep VAE

Seongju Jang, Tung Do, Nikhil Vaidyanathan, Elizabeth Lombard
Department of CEE, ECE, CSE, CMB
University of Michigan
jseongju@umich.edu, tungsdo@umich.edu, nikhillv@umich.edu, lombarde@umich.edu

1 Introduction

1.1 Generative model and VAE

Generative models have emerged as powerful tools for understanding and synthesizing complex data distributions. These models aim to learn the underlying patterns of a dataset, enabling the generation of new data samples that resemble the original dataset. Variational Autoencoders (VAEs) are one of the foundational architectures in this domain. By combining probabilistic modeling with deep neural networks, VAEs offer a structured framework for data generation and latent representation learning. However, as the complexity of the data increases, traditional VAEs often struggle to capture both global and local features effectively. This limited their generative capabilities compared to autoregressive models[3].

A primary limitation is their reliance on a single-layer latent space, which can restrict their ability to model high-dimensional, multi-scale data like images. This results in suboptimal reconstructions and less realistic samples, particularly for data with intricate details. Furthermore, achieving stability during training becomes increasingly difficult as the model depth increases, and the absence of hierarchical representations hampers the disentanglement of latent features [3]. Recent advancements, such as hierarchical VAEs, have attempted to address these issues, but their scalability and efficiency remain constrained [9].

1.2 Problem statement and objective

The goal of the Very Deep Variational Autoencoder (VAE) is to address the mentioned limitations [3]. Very Deep VAE is an architecture that introduces a hierarchical structure of latent variables across multiple layers. This design allows for the capture of global and local data features, enabling superior generative performance on complex datasets. Through a systematic exploration of this architecture, the we aim to: (1) Reproduce an efficient, scalable hierarchical VAE model capable of processing more complex image data. (2) Quantitatively and qualitatively evaluate its performance on standard and custom datasets. (3) Highlight the advantages of the Very Deep VAE in generating realistic samples and reconstructing data with varying levels of detail.

By addressing these objectives, this study contributes to advancing the field of generative modeling, offering insights into the potential of hierarchical architectures for complex data tasks.

2 Related Research

2.1 Autoregressive model

Autoregressive models have demonstrated impressive results in generative tasks, particularly in image and audio synthesis. Models like PixelRNN and PixelCNN model the conditional probability of each pixel in an image, one at a time, achieving high-quality outputs [11]. Subsequent improvements, such as PixelCNN++, refined the discretized likelihood functions and enhanced efficiency [8]. Another significant contribution is WaveNet, which applied autoregressive principles to raw audio generation, setting benchmarks in audio quality [10].

However, these models often suffer from slow sampling times due to their sequential nature, making them less practical for large-scale applications. This limitation motivates the exploration of alternative generative frameworks, such as hierarchical approaches.

2.2 Hierarchical VAE architecture

Variational autoencoders (VAEs) have been widely adopted for generative tasks, and hierarchical extensions have improved their expressivity. Ladder VAEs introduced a multi-layer latent variable hierarchy, enabling the model to capture complex, high-dimensional data distributions [9] . Importance Weighted Autoencoders (IWAE) improved the evidence lower bound (ELBO) approximation, offering better latent variable modeling [2]. Additionally, stochastic backpropagation has been a pivotal technique for optimizing these deep latent variable models [7].

Despite their success, hierarchical VAEs often face challenges in training stability and scalability. For instance, as the number of latent layers increases, the optimization process becomes more sensitive, limiting their practical application in very deep architectures.

2.3 Summarization of limitations of previous research

Autoregressive models excel at fine-grained sequential modeling but are computationally expensive, as they require iterative sampling for each output element ([11]; [8]. On the other hand, while VAEs offer parallelization and scalability, their generative quality can suffer due to the posterior approximation's limitations [5]. Hierarchical VAEs alleviate some of these issues but introduce additional complexity in their design and training ([9]; [2]).

These limitations underline the need for models that combine the generative quality of autoregressive approaches with the scalability of VAEs while addressing training and computational efficiency.

3 Very Deep VAE

3.1 Concept of the model

The Very Deep Variational Autoencoder (VAE) extends the traditional VAE by introducing a hierarchical structure of latent variables, allowing it to model data at multiple levels of detail. This enables the model to capture global and local features effectively, making it suitable for complex, high-dimensional data such as images.

The encoder processes input data through layers that generate latent variables at different resolutions, while the decoder reconstructs the data by sequentially refining details from coarse to fine. Advanced architectural features such as residual connections, bottleneck layers, and multi-scale representations enhance stability and efficiency during training. By aligning the hierarchical structure with a tailored prior distribution, Very Deep VAEs learn compact, meaningful representations and excel in generative modeling tasks.

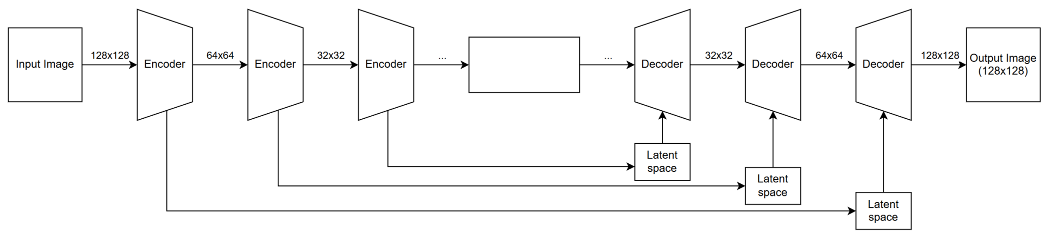

Figure 1: VAE Model Architecture [1]

3.2 Evidence Lower Bound (ELBO)

The foundational idea of the Very Deep VAE is based on the VAE. The structure of VAE is described in Figure 1. In this structure, the Evidence Lower Bound (ELBO) is used as the object function.

$$\log p_\theta(x) \geq \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - D_{KL} \left[q_\phi(z \mid x) \parallel p_\theta(z)\right] \tag{1}$$

ELBO is described in Equation 1 [3]. The equation consist of two parts. First is reconstruction term, which measures how likely the reconstructed image is to match the original image. This is quantified as the log-likelihood of the data given the latent variables. The second part is the KL divergence term, which regularizes the latent space to follow the prior distribution.

$$q_\phi(z|x) = q_\phi(z_0|x) q_\phi(z_1|z_0, x) \cdots q_\phi(z_N|z_{<N}, x) \tag{2}$$

$$p_\theta(z) = p_\theta(z_0) p_\theta(z_1|z_0) \cdots p_\theta(z_N|z_{<N}) \tag{3}$$

In Very Deep VAE, there are multiple layers (N), and each term is calculated by multiplication (Equation 2, 3). The objective function is formulated to maximize its value. In the implementations, the sign is inverted and the problem became minimiztion problem. Calculation of ELBO loss was implemented in source file `vae helpers.py`.

3.3 Novelty of Very Deep VAE and code implementation

The Very Deep VAE introduces several architectural innovations that extend the traditional Variational Autoencoder (VAE) framework. Its primary novelty lies in its hierarchical structure, which organizes latent variables across multiple layers, each capturing increasingly detailed aspects of the data. This approach allows the model to encode and reconstruct complex data distributions more effectively than a single-layer VAE. Moreover, the hierarchical nature also supports multi-scale feature learning, which is important for generating high-quality reconstructions and diverse samples.

The Very Deep VAE incorporates a structured prior distribution that aligns with its hierarchical latent space. This prior encourages the model to learn meaningful and disentangled representations at each hierarchy level. As a result, the model excels in generating realistic data and provides interpretable latent variables that can be used for tasks such as conditional generation and data analysis.

Another key aspect of the Very Deep VAE is its use of advanced architectural components, such as residual blocks, bottleneck layers, and multi-scale feature maps. Residual connections help stabilize training in very deep networks by enabling the flow of gradients across layers, while bottleneck layers reduce the computational cost and ensure compact representations.

Our team studied the structure in detail and reproduced it. In a Very Deep VAE, the encoder and decoder are designed to operate at multiple resolutions. The encoder and decoder are built as separate modules. The modular design allows easy customization of the architecture.

3.3.1 Encoder

The encoder processes the input data through a series of blocks, progressively extracting features at coarser and finer levels. The encoder uses convolutional layers, residual blocks, and down-sampling to encode data into hierarchical latent variables. We defined structure of based block on `base block.py` and utilize this block to organize encoder implementation. The hierarchical processing ensures that global features, such as an image's overall structure, and local details, such as textures and edges, are represented in separate latent variables.

3.3.2 Decoder

The decoder mirrors this process by reconstructing the data layer by layer, starting from coarse representations and gradually adding finer details. To enable the hierarchical structure, the DecoderBlock code was defined, and by stacking these blocks, decoder structure was built. In the decoder module, KL divergence and negative log-likelihood was calculated and combined to acquire ELBO loss. This paper employs diagonal Gaussian distributions for stochastic layers, ensuring stability even in deep architectures, which is one of the novelty of this work.

4 Experiment

4.1 Dataset

In this report, experiments were conducted on three datasets. First, we conducted experiment utilizing the (1) CIFAR-10 dataset and (2) ImageNet32, whcih were also evaluated in the original paper. In addition, we conducted an additional experiment using an external dataset, the (3) Oxford 102 Flower dataset, **which was not used in the original research** [6].

The ImageNet dataset consists of 1.4M of images. In this experiment, we were unable to use entire dataset due to limited computing power. Instead, **we extracted 10,000 images** and conducted the experiment. Unlike CIFAR-10 and ImageNet, which cover a diverse categories ranging from vehicles to animals, the Oxford 102 Flower dataset focuses specifically on images of flowers. The reason for choosing this custom dataset was to compare the model's performance on single-purpose dataset. The amount of data for each dataset and the train/validation/test splitting are recorded in Table 1.

Table 1: Three dataset used in the experiment

	CIFAR-10	ImageNet 32	Oxford 102 Flower
Images Categories	animals, vehicles	various	flowers only
Used in original paper	Yes	Yes	No
Total Data	60,000	10,000	8,189
Train Data	45,000	8,100	6,583
Validation Data	5,000	900	737
Test Data	10,000	1,000	819

4.2 Model Training (and training code implementation)

The training and evaluation codes were implemented for the training process. Based on the original paper's hyperparameters, adjustments were made to fit the custom dataset and our hardware environment. As mentioned earlier, generative model VAE was trained using the ELBO objective function, with the negatvie vlaue of ELBO used as the loss function.

Gradient skipping is an important method for effectively training this model with deep layers. Using this method, the proposed model can be trained more stably. During the training loops, if the gradient norm exceeds a certain predefined threshold, the update step for that iteration is skipped. This gradient norm was calculated by torch.nn.utils module, and the threshold for the skipping was set to 400 for all three dataset. Moreover, an Exponential Moving Average (EMA) of model parameters further enhances evaluation stability. Robust checkpointing stores model and optimizer states, while built-in visualization tools (`visualization.py`) generate reconstructions, and logging code was implemented to monitor progress and assess learned representations.

Table 2: Hyperparameter setting

	Setting	**Comments**
Optimizer	AdamW	implemented by torch.optim.AdamW, linear scheduler was used
Warm up	100	100 iterations
Learning rate	0.0002	Set based on the original paper
Epoch	100	Since our image data was only 32x32 in size, we selected relatively small epoch. Validation was conducted every epoch.
num layers	45	selected based on original paper's parameter

In the original paper, the model's depth and hidden size varied depending on the dataset. For our experiments, since the image size and dataset size were small, we used a relatively small number of layers and a smaller hidden size. Our model had 45 layers and hidden size of 384, with the total of 39M parameters. In this experiment, we used an RTX 4070 GPU for training. While the original paper employed distributed training with the Apex module [4], we implemented the training code using a single GPU.

4.3 Experimental results

$$NLL = -logp_\theta(x) \tag{4}$$

The evaluation of the model is based on Negative Log-Likelihood (NLL) (Eq. 4). NLL evaluates the predictive probability distribution of the model. Therefore, it is often used as an evaluation metric in models such as VAEs and auto regressive models. In the original paper, the author used NLL metrics for validation [3]. Here, the NLL value is indirectly derived using ELBO value. The (-ELBO) is approximated as the NLL value and is used for comparison. Table 3 shows the experiment result. Although there were differences in the numbers, we were able to reproduce the model from the original paper. For comparison, **Gated Pixel CNN achieved 3.83 NLL** in ImageNet32 dataset [12]. Our model had much lower NLL value. However, we used filtered version of the ImageNet32 dataset, and test dataset was much smaller. These differences may be the reason for the performance difference.

Table 3: Experiment results for Training Time and Negative Log-Likelihood (NLL) over different Datasets

	CIFAR-10	ImageNet 32	Oxford 102 Flower
Training time	31 h	6 h	5 h
NLL	2.151	2.491	3.265

After training, we conducted visual verification on two aspects. First, we generated unconditional images and qualitatively assessed them. Second, we reconstruct some test images from different stages in the latent hierarchy.

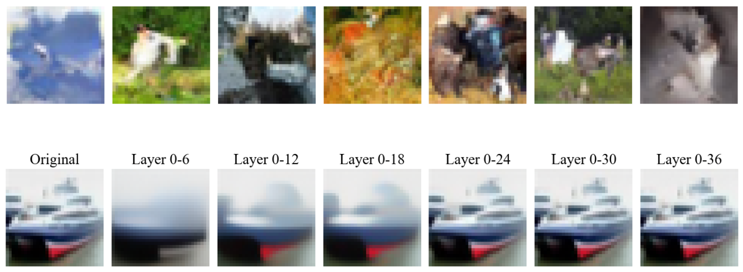

Figure 2: Result on CIFAR-10, Unconditionally generated image (Above), Reconstruction using selected layers (Below)

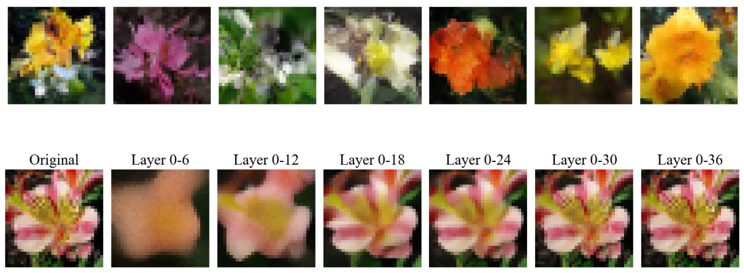

Figure 3: Result on 102 Flower, Unconditionally generated image (Above), Reconstruction using selected layers (Below)

The example samples (CIFAR-10) for unconditional generation are described at the top of the Figure 2. When qualitatively analyzed, the approximate outlines of animal shapes could be observed. However, some images made it difficult to clearly identify the specific animal type or even confirm that it was an animal. These results were discussed in more detail in the discussion section. The seven images in bottom of the Figure 2 is the result comparison on various stages in the latent hierarchy. It can be observed that low-resolution latent variables determine the broad structure of images, while high-resolution variables are enabling finer details. This confirms that the hierarchical structure of the Very Deep VAE captures and generates images features step by step as intended. Figure 3 shows the similar result for Oxford 102 Flower dataset. We can clearly see the images of the flower in this result.

4.4 Discussion on generated images

When examining unconditionally generated images, we can see that the images generated from the Oxford 102 Flower dataset are much clearer compared to those from the CIFAR-10 dataset, even the CIFAR-10 model had smaller (-ELBO) value. A lower (-ELBO) value not necessarily guarantee visually satisfactory image generation from human inspection, especially if we are comparing different datasets.

The differences in quality arises because the Oxford 102 Flower dataset contains only flower images, while the CIFAR-10 dataset includes a much wider variety of images, which are entirely different in nature. This likely limited the ability of the latent variables to accurately represent the data.

5 Conclusion

The Very Deep Variational Autoencoder (VAE) demonstrates the potential of hierarchical architectures in enhancing generative performance for complex datasets. By introducing a structured hierarchy of latent variables and advanced architectural components, the model effectively captures both global and local features of data. The experiments conducted on CIFAR-10 , ImageNet32 and the Oxford 102 Flower dataset highlight the scalability and adaptability of the model, with notable improvements in image generation quality and feature representation.

Despite these successes, challenges remain in optimizing for datasets with diverse categories, such as CIFAR-10, where deeper networks and more computational resources could further enhance performance. Future work could explore these avenues alongside integrating larger datasets and additional hierarchical priors to expand the model's applicability. This study underscores the importance of combining scalability, efficiency, and generative quality in designing next-generation VAEs, paving the way for advancements in generative modeling across a wide array of applications.

6 Contribution by Group Members

Name	Contribution
Seongju	Contributed on explaining Method section on proposal. Mainly focused on implementing training related code (`train.py`, `train setup.py`, and `utils train.py`). Run the experiment code and debug the errors in the code integration. Conduct experiments for each dataset. Wrote experiment and result part on the final report.
Tung	Worked on the proposal's Plan section, configured environment setting to run the repo, worked on `vae.py`, created `train.ipynb` on `Tung Do` branch, added checkpoint load/save and loss save functions to `train.ipynb`, ran the model's training with laptop GPU, reproduce the code in `decoder.py`, `DmolNet.py`, `encoder.py`, and new `vae.py` on main branch, worked on Very Deep VAE section in the final report.
Nikhil	Whiteboarded and planned out the different files required for the project, then the specific functions to streamline the coding process/write efficient code. Implemented the helper functions for vae, train, and other utils, set hyperparameters.py, and `base block.py`. Ran the model's training with Macbook's GPU to ensure replicability of results over multiple systems.
Elizabeth	Researched and found the paper used for replication and experimentation in the project. Performed literature research on relveant works necessary for successful compeletion and understanding of the model. Simplified and reorganized the data.py visualization.py and preprocessing custom dataset.py for a stream-lined and easy to update data loading and preprocessing pipeline allowing for easy experimentation. Worked on introduction and related works.

References

[1] Data Science Blog. Variational autoencoders: Concepts and implementation, 2022. URL `https://data-science-blog.com/blog/2022/04/19/variational-autoencoders/`. Accessed: 2023-12-06.

[2] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015. URL `https://arxiv.org/abs/1509.00519`.

[3] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=RLRXCV6DbEJ`.

[4] Patrick Diehl, Gregor Daiss, Kevin Huck, Dominic Marcello, Sagiv Shiber, Hartmut Kaiser, Juhan Frank, Geoffrey C Clayton, and Dirk Pflueger. Distributed, combined cpu and gpu profiling within hpx using apex. *arXiv e-prints*, pages arXiv–2210, 2022.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. URL `https://arxiv.org/abs/1312.6114`.

[6] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics Image Processing*, pages 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

[7] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/rezende14.html`.

[8] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *ArXiv*, abs/1701.05517, 2017. URL `https://api.semanticscholar.org/CorpusID:12663716`.

[9] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Sigurd Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *arXiv preprint arXiv:1602.02282*, 2016. URL `https://arxiv.org/abs/1602.02282`.

[10] Aaron Van den Oord, Sander Dieleman, Heiga Zen, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. URL `https://arxiv.org/abs/1609.03499`.

[11] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1747–1756. JMLR.org, 2016.

[12] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016. URL `https://arxiv.org/abs/1606.05328`.