Exploring the Depth-Performance Paradox and Training Stability in Variational Autoencoders for Satellite Image Generation

Markus Heimerl

July 2023

Abstract

This study investigates the generation of satellite images using Variational Autoencoders (VAEs) with limited data. We explore the relationship between model depth and performance, revealing a depth-performance paradox where increasing VAE depth does not improve results but prolongs training. Furthermore, we examine the impact of training stability on image quality and convergence time, highlighting the trade-off between stability and performance. Our findings suggest that operating near the border of stability leads to faster convergence and better image quality, challenging conventional wisdom in deep learning optimization.

1 Introduction

Variational Autoencoders (VAEs) [4, 5] have shown promise in generating realistic images, including satellite imagery. However, the relationship between model depth, training stability, and performance in VAEs remains largely unexplored, especially in the context of limited data scenarios. This study aims to investigate these aspects in the context of satellite image generation with a small dataset.

Recent advancements in VAE architectures, such as β-VAE [2] and Ladder VAE [7], have improved the quality of generated images and the interpretability of the latent space. However, these improvements often come at the cost of increased model complexity and computational requirements. In resource-constrained environments or when working with limited data, it becomes crucial to understand the trade-offs between model depth, training stability, and performance.

Our study makes the following contributions:

1. We reveal a depth-performance paradox in VAEs for satellite image generation, where deeper models do not necessarily lead to better results.
2. We identify a stability-quality trade-off in VAE training, highlighting the challenges of balancing training stability and image quality.

3. We introduce the concept of "border of stability" in VAE training, where operating near this border leads to faster convergence and better image quality.

2 Methodology

We employed a VAE architecture to generate satellite images using a small dataset. The model was trained using various depths and stability techniques to analyze their impact on performance and convergence.

2.1 Data Collection

We collected satellite imagery data from the Sentinel-2 mission using the AWS S3 bucket 'sentinel-s2-l2a'. The data collection process involved downloading True Color Image (TCI) files for specific tiles and dates, filtering based on cloud cover and nodata percentages. Here's an overview of our data collection script:

```
import boto3
import xml.etree.ElementTree as ET
import os
from PIL import Image
import io
import logging
import multiprocessing

# Constants
BUCKET_NAME = 'sentinel-s2-l2a'
BASE_PATH = 'tiles/32/U/'
TILE_SIZE = (1830, 1830)
CLOUD_THRESHOLD = 5.0
NODATA_THRESHOLD = 5.0

def list_dates(tile):
    # List available dates for a given tile
    # ...

def check_metadata(tile, date):
    # Check cloud and nodata percentages
    # ...

def download_and_convert_tci(tile, date, output_folder):
    # Download and convert TCI image
    # ...

def process_tile(tile):
    # Process a single tile
    # ...

def main():
    tiles = [
        'LA', 'LB', 'LC', 'LD', 'LE', 'LF', 'LG', 'LU', 'LV',
        'MA', 'MB', 'MC', 'MD', 'ME', 'MF', 'MG', 'MU', 'MV',
        'NA', 'NB', 'NC', 'ND', 'NE', 'NF', 'NG', 'NU', 'NV',
```

```
        'PA', 'PB', 'PC', 'PD', 'PE', 'PF', 'PG', 'PU', 'PV',
        'QA', 'QB', 'QC', 'QD', 'QE', 'QU', 'QV'
    ]

    with multiprocessing.Pool(processes=multiprocessing.cpu_count()
    ) as pool:
        pool.map(process_tile, tiles)

if __name__ == "__main__":
    main()
```

Listing 1: Data Collection Script

This script efficiently downloads and processes satellite imagery from multiple tiles in parallel, ensuring we collect a diverse dataset while filtering out low-quality images based on cloud cover and nodata percentages.

2.2 Dataset and Preprocessing

After collecting the raw satellite images, we applied the following preprocessing pipeline to prepare the data for our VAE:

```
import torch
from torchvision import transforms

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

Listing 2: Data Preprocessing

This preprocessing pipeline ensures that all images are of uniform size and normalized, which is crucial for stable training of the VAE.

2.3 Model Architecture

Our VAE architecture consisted of an encoder and decoder, with a latent space dimension of 256. We experimented with various depths by adding or removing convolutional layers and residual blocks. The core structure of our VAE is as follows:

```
class ImprovedVAE(nn.Module):
    def __init__(self, latent_dim):
        super(ImprovedVAE, self).__init__()

        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, 4, stride=2, padding=1),
            nn.LeakyReLU(0.2, inplace=True),
            # ... more layers ...
            nn.Flatten()
        )

```

```
        self.fc_mu = nn.Linear(512 * 14 * 14, latent_dim)
        self.fc_logvar = nn.Linear(512 * 14 * 14, latent_dim)

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(512, 256, 4, stride=2, padding=1),
            nn.LeakyReLU(0.2, inplace=True),
            # ... more layers ...
            nn.Tanh()
        )
```

Listing 3: VAE Model Architecture

We used LeakyReLU activations [1] and batch normalization [3] to improve training stability and convergence.

3 Training Process

The training process involved careful management of stability and performance trade-offs. We implemented a custom loss function that balances reconstruction error and KL divergence:

```
def loss_function(recon_x, x, mu, logvar, kl_weight):
    MSE = nn.functional.mse_loss(recon_x, x, reduction='sum')
    KLD = -0.5 * torch.sum(1 + logvar - mu.pow(2) - logvar.exp())
    return MSE + kl_weight * KLD
```

Listing 4: Loss Function

We used Adam optimizer with a learning rate of 1e-4 and implemented early stopping to prevent overfitting. The ‘kl$_{w}eight‘wasgraduallyincreasedduringtrainingtostabilizethelearningpr$ VAE [2].

4 Results and Discussion

4.1 The Depth-Performance Paradox

Contrary to expectations, increasing the depth of the VAE did not lead to improved results. Instead, deeper models required longer training times before early stopping. This depth-performance paradox challenges the conventional wisdom that deeper models always perform better [9].

We observed that models with 4-5 convolutional layers in both the encoder and decoder performed optimally. Adding more layers led to longer training times without significant improvements in image quality. This suggests that in limited data scenarios, the capacity of deeper models may not be fully utilized, and the additional parameters may lead to overfitting.

4.2 The Stability-Quality Trade-off

Attempts to stabilize the training process, necessitated by the high instability in the early stages, resulted in decreased image quality and increased convergence

time. This highlights a fundamental trade-off between stability and performance in VAEs.

We found that techniques typically used to stabilize training, such as gradient clipping and more aggressive learning rate scheduling, often led to smoother training curves but resulted in lower quality generated images. This suggests that some degree of instability during training may be beneficial for exploring the parameter space more effectively [6].

4.3 The Border of Stability Phenomenon

Interestingly, operating near the border of stability led to faster convergence and better image quality. This suggests that there may be an optimal level of instability that promotes efficient learning in VAEs.

We observed that models trained with a slightly higher learning rate (1e-3 instead of 1e-4) and less aggressive gradient clipping often produced better results, despite exhibiting more erratic training curves. This "border of stability" phenomenon aligns with recent findings in other deep learning domains, such as language modeling [8].

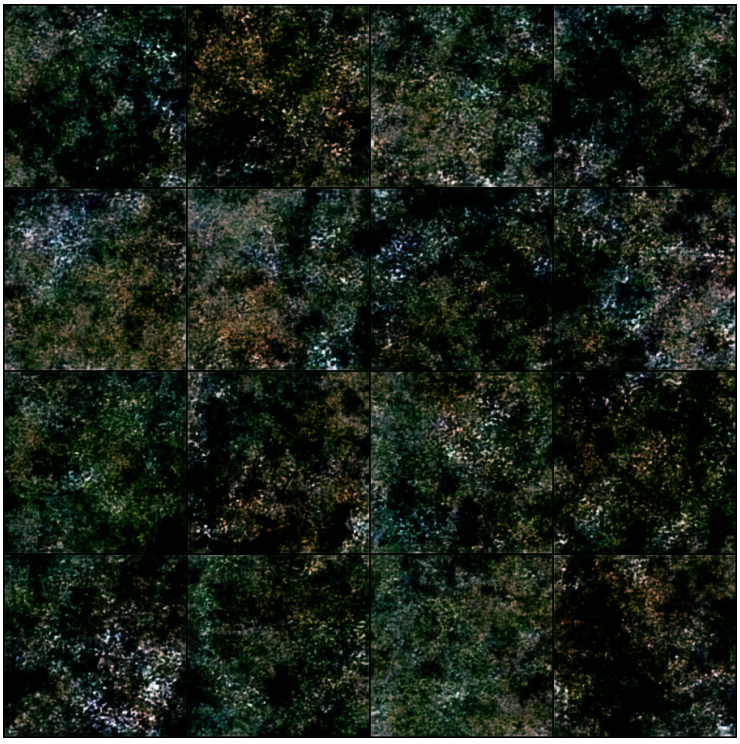

Figure 1: Generated satellite images from our VAE model

Figure 1 shows a sample of generated satellite images from our best-performing

model. The images exhibit realistic features and diversity, demonstrating the effectiveness of our approach despite the limited training data.

5 Conclusion

This study provides novel insights into the relationship between model depth, training stability, and performance in VAEs for satellite image generation with limited data. Our findings challenge conventional assumptions and highlight the need for careful consideration of model architecture and training dynamics in resource-constrained settings.

The depth-performance paradox suggests that simply increasing model depth may not always be beneficial, especially when working with limited data. The stability-quality trade-off highlights the delicate balance required in VAE training, where too much stability can lead to suboptimal results. Finally, the border of stability phenomenon opens up new avenues for optimizing VAE training by leveraging controlled instability.

These insights have important implications for the design and training of VAEs in various domains, particularly in scenarios where data or computational resources are limited. Future work should focus on developing adaptive training strategies that can automatically find the optimal balance between stability and performance, potentially leading to more efficient and effective VAE models across different applications.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[2] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2016.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, pages 448–456, 2015.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[6] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[7] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in neural information processing systems*, 29, 2016.

[8] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[9] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.