Machine Learning Approaches in Structural Mechanics

Final Report

Muhammed Burak Görmüs

5584907

burakkgrms@gmail.com

August 13, 2023

Contents

1

Introduction

Generative AI refers to the models that have the capability to generate new content, such as images and texts based on existing examples. These methods have the ability to learn general patterns and features based on input data, and they can generate unique outputs similar to the input data.

There are three common methods in the field of generative AI: (i) generative adversarial networks (GANs), (ii) denoising diffusion models, and (iii) variational autoencoders (VAEs). Each of them their pros and cons, and when choosing one method over another, the following three aspects could be useful: (i) high-quality samples, (ii) fast sampling, and (iii) diversity [1].

In this report, variational autoencoders (VAEs) are used to generate images of microstructures in a material, demonstrating different grain sizes and shapes. Variational autoencoders (VAEs) are preferred due to their fast sampling process, as they provide both rapid sampling and diversity in outputs [1].

Variational autoencoders (VAEs) transform the data into a smaller, more manageable set of dimensions, and they learn the fundamental features of the data in an unsupervised manner [2]. In detail, this process contains two fundamental operations: (i) encoding, and (ii) decoding. During encoding, the aim is to yield one value for each encoding dimension, thus creating a bottleneck-like layer. On the other hand, the decoder uses these values to try and generate the original input [3].

The difference between autoencoders and variational autoencoders (VAEs) is that variational autoencoders employ stochastical variables *(mean and variance)* to create sample latent to be fed into the decoder instead of a simple bottleneck. Further, in general, the loss during the implementation of variational autoencoders (VAEs) is assumed to be the sum of cross entropy losses and KullbackLeibler (KL) divergence losses [4].

In this report, the effect of the latent space dimension is examined as well as the effect of the number of layers in the encoder and decoder, while trying to generate pictures of microstructures in a material. The code for the report can be found on the GitHub page [1]. In the following sections, the detailed implementation and results are shared.

[1] https://github.com/MuhammedBurakGormus/VariationalAutoEncoders

2

Implementation

In this section, the methodologies that are employed to investigate the impact of varying latent space dimensions and the number of layers in both the encoder and decoder on the process of generation are shared. To create various variational autoencoders (VAEs) architectures, TensorFlow and its API, TensorFlow.keras are used within the Python programming environment [1].

For both studies, 1000 pictures having a size of 64x64 pixels are used as input data. These pictures are, then, stored as arrays with only their RGB channels. During this step, the PIL module on Python is used [2]. Then, the batch size is determined as 64, and input data is shuffled to provide a diverse and randomized order of elements.

During the training phase, the assumption is that the losses originate from two separate sources: Kullback-Leibler (KL) divergence losses, and binary cross-entropy losses. While tf.keras.losses.BinaryCrossentropy() command is used to find the construction loss, the KL loss is calculated by:

$$\text{KL\_loss} = 0.012 \left(-0.5 \cdot \left(1 + z_{\log_{\text{var}}} - (z_{\text{mean}})^2 - \exp(z_{\log_{\text{var}}}) \right) \right) \tag{2.1}$$

Further, the number of epochs for one training session is chosen as 151.

2.1 Examining the effect of latent space dimension on generation

For this study, the latent space dimensions are chosen to be 2,4,8,16, and 32, and 5 different variational autoencoder architectures are created where the same encoder and decoder architectures are used. For the encoder, a series of convolutional and max-pooling layers are applied. While the layers progressively reduce the spatial dimensions, the number of filters increases in each layer such that:

- 16 filters in the first layer,
- 32 filters in the second layer,
- 64 filters in the third layer,
- 128 filters in the fourth layer

Each convolutional layer is followed by a max-pooling layer with a (2, 2) pooling size. The output from the last pooling layer is flattened to create a 1D vector. Then, mean and log-variance corresponding to the latent space dimensions of 2,4,8,16, and 32 are created.

[1] https://www.tensorflow.org/guide/keras
[2] https://python-pillow.org/

The decoder is chosen to be the opposite of the encoder. The decoder takes a latent space vector created by mean and log-variance as an input. A series of convolutional and upsampling layers are applied to gradually reconstruct an output image with dimensions (64, 64, 3). The layers progressively increase the spatial dimensions while reducing the number of filters such that:

- 128 filters in the first layer,
- 64 filters in the second layer,
- 32 filters in the third layer,
- 16 filters in the fourth layer

Each convolutional layer is followed by batch normalization and a Leaky ReLU activation. The final layer produces the reconstructed image.

2.2 Examining the number of layers in the encoder and decoder

For this study, the latent space dimension is determined to be 16, while different numbers of layers are used for 3 variational autoencoder architectures. In detail,

- First architecture with 4 layers in both encoder and decoder:
  - Encoder: 16 filters, followed by 32, 64, and finally 128 filters in sequence.
  - Decoder: 128 filters, followed by 64, 32, and finally 16 filters in sequence.
- Second architecture with 3 layers in both encoder and decoder:
  - Encoder: 16 filters, followed by 32, and finally 64 filters in sequence.
  - Decoder: 64 filters, followed by 32, and finally 16 filters in sequence.
- Second architecture with 2 layers in both encoder and decoder:
  - Encoder: 16 filters, followed by 32 filters in sequence.
  - Decoder: 32 filters, followed by 16 filters in sequence.

For each encoder, similar to the architectures in section 2.1, each convolutional layer is followed by a max-pooling layer with a (2, 2) pooling size. The output from the last pooling layer is flattened to create a 1D vector. During decoding, a series of convolutional and upsampling layers are applied to gradually reconstruct an output image with dimensions (64, 64, 3).

3

Results

In this chapter, the results of the two studies described in the previous chapter are given.

3.1 The effect of latent space dimension on generation

As said before, 5 variational autoencoders (VAEs) architectures with different latent space dimensions *(2,4,8,16,32)* are used to examine the effect of latent space dimension on generation. The number of parameters that are obtained for these architectures are shared below in Table 3.1.

Latent space dimension	Total paramaters
2	98,087
4	103,723
8	114,995
16	137,539
32	182,627

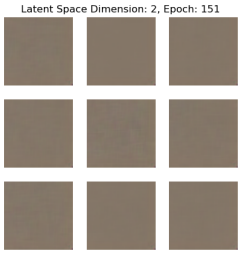
Table 3.1: Latent space dimension vs. number of parameters to be trained

Examining the final generated pictures for each architecture as given in Figure 3.1, it is seen that only the latent space dimensions of 8 and 16 generate similar pictures to the input pictures. Throughout the training sessions for each architecture corresponding to different latent spaces, the total loss is recorded for each epoch. The comparison of the loss functions is given in Figure 3.2.

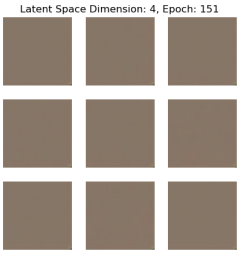
As a result, it can be concluded that working with a small latent space could not be sufficient to extract fundamental features. On the other hand, it does not necessarily mean increasing the latent space always yields better results. As seen in Figure 3.1.e, the latent space of 32 is not able to generate a picture. Comparing it with the latent spaces of 8 and 16, the average loss is higher than both of them, as seen in Figure 3.2.b. Besides, working with a bigger latent space can cause a longer time to reduce the average loss as shown in Figure 3.2.c.

3.2 The effect of number of layers in the encoder and decoder on generation

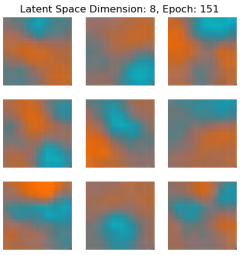
3 variational autoencoder architectures with different numbers of layers are created to examine the effect of layers on generation. The loss values are observed to demonstrate similar trends for the created architectures, and it is shown in Figure 3.3.



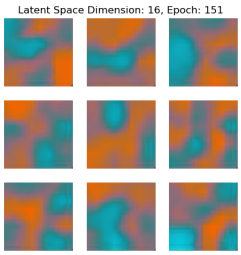
(a)Latent space dim.: 2



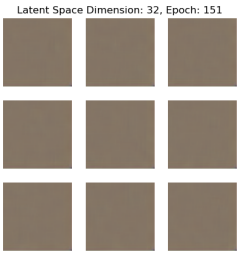
(b)Latent space dim.: 4



(c)Latent space dim.: 8

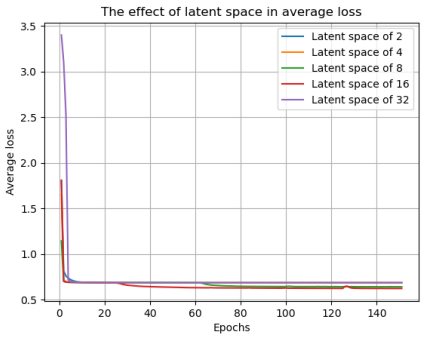


(d)Latent space dim.: 16

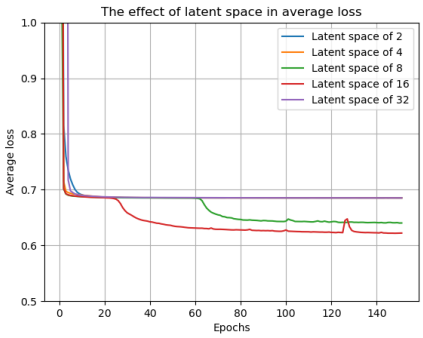


(e)Latent space dim.: 32

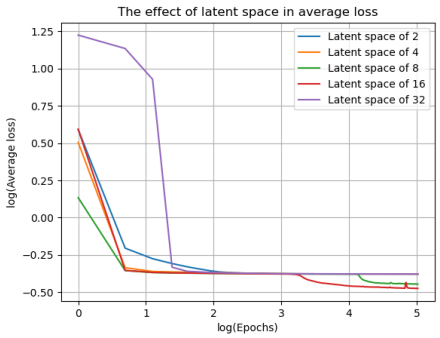
Figure 3.1: Generation of pictures containing microstructures with Variational Autoencoders (VAEs) using latent spaces of sizes: (a) 2, (b) 4, (c) 8, (d) 16, (e) 32.



(a)Average loss vs epochs



(b)Zoomed plot



(c)Log-log scale

Figure 3.2: Comparison of average loss for different latent spaces of 2,4,8,16, and 32. Note that (b) is the zoomed plot of (a) with the limit of y = [0.5,1], c) is the log-log scale of the plot (a).

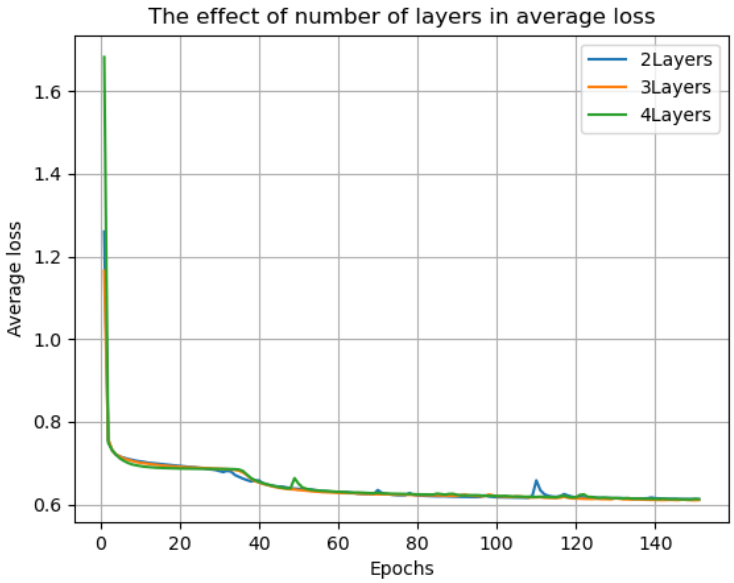
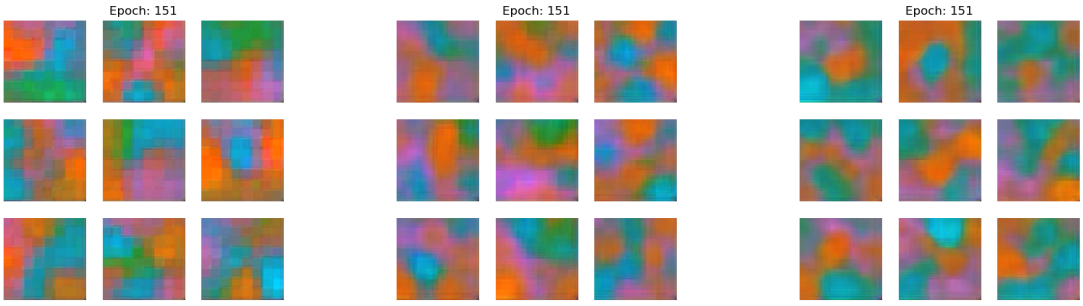


Figure 3.3: The effect of number of layers in generation for 3 different variational autoencoder architectures. The legend demonstrates how many layers are used in encoders as well as in decoders. *For example, the legend 2 Layers illustrates that 2 layers in encoder and 2 layers in decoder are used.*

Although the trends in the loss are similar, it is observed that each architecture captures different fundamental characteristics of an image, as shown in Figure 3.4. When using less number of layers (Figure 3.4.a), more colorful plots are obtained, although the generated image is quite noisy. On the other hand, increasing the layers (Figure 3.4.c) could lead to less noise and more distinct borders, but the color information is suppressed.



(a)2+2 layers (b)3+3 layers (c)4+4 layers

Figure 3.4: Comparison of final generated pictures.

4

Conclusion

Variational autoencoders (VAEs) are one of the powerful methods to generate new content. For example, in this report, images representing the microstructure of a material are tried to be generated. While trying the generate the microstructure of a material, the size of latent space is studied, and it is seen that a small number of latent space could be enough to capture the fundamental characteristics of images, unless it is too small. Also, it is seen that increasing the latent space dimension can lead to undesired content generation. Therefore, while trying to generate new images, the sweet spot should be found to optimize the generation. Further, the number of layers in encoders and decoders is examined. It is concluded that each architecture is unique to extract different characteristics of the content.

Bibliography

[1] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans, 2022.

[2] Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review, 2008.

[3] Jeremy Jordan. Variational autoencoders, 2018.

[4] Taehyeon Kim, Jaehoon Oh, Nak Yil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation, 2021.