Computational Modelling for Materials Engineering

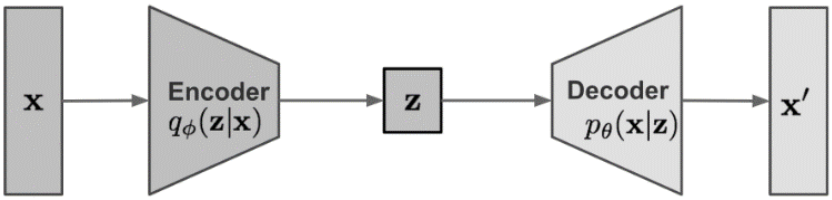
Final Project

Project: Deep Generative Models for Polysilicon Microstructures

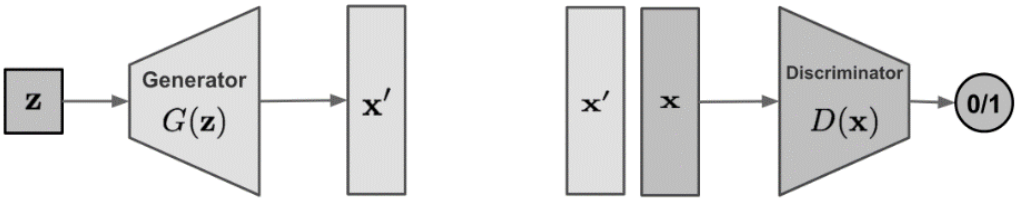
Background

One of the main challenges of implementing effective Deep Learning models is the availability of sufficient data, which can be limited not only in terms of quantity but also diversity. As a result, the development of techniques for the generation of high-fidelity synthetic data has gained relevance in recent years. Deep generative models have demonstrated great capabilities to accomplish this task and produce highly realistic data. The most prominent generative models are the variational autoencoder (VAE) [1] and the generative adversarial network (GAN) [2]. See Figure 1.

In this project, students are incentivized to explore among different deep generative strategies to implement a model that effectively learns how to generate highly realistic image data. For this an image dataset will be facilitated for the training, where samples represent microstructures typical from epitaxially-grown polysilicon thin-films. See Figure 2.



(a)



(b)

Figure 1. Schematic representation of unsupervised deep generative models. (a) VAE. (b) GAN.

X and **X′** represent the original input data and generated data, respectively. **Z** represents the compressed representation of the data, also referred to as the latent representation; a space of significantly lower dimensionality compared with the original input.

In contrast to traditional autoencoders[1], VAEs learn latent variables with continuous distributions, a particularly useful property for generative modeling tasks. VAE encoding returns a distribution over the latent space rather than discrete values. More specifically, the encoder produces a set of two vectors including a vector of means (μ), and another vector of standard deviations (σ). As such, the VAE attempts to learn the distributions of latent variables based on the mean values and their variances, instead of learning a deterministic mapping, as in traditional autoencoders. A comprehensive tutorial of the VAE approach is presented in [3].

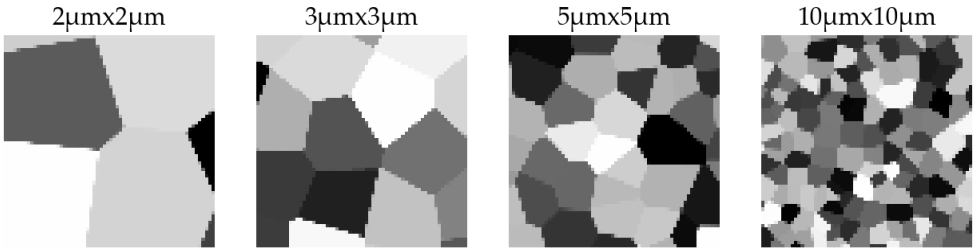


Figure 2. Exemplary microstructures representing epitaxially-grown polysilicon thin-films. Each image is linked to a label (classes 2, 3, 5, 10) as well as continuous target values given by the homogenized Youngs modulus, E.

Conversely, in GANs, the generative network generates candidates while the discriminative network evaluates them. The generative network learns to map from a latent space to a data distribution of interest (e.g. generated microstructure image), while the discriminative network distinguishes candidates produced by the generator from the true data distribution (e.g. original microstructure image). The generator´s training objective is to produce novel candidates that the discriminator thinks are part of the true data distribution, while the discriminator´s training objective is to differentiate between samples extracted from each distribution. Typically, the generator is seeded with a randomized input that is sampled from a predefined latent space (e.g. a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator. Independent backpropagation procedures are applied to both networks so that the generator produces better samples, while the discriminator becomes more skilled at flagging synthetic samples [4].

Students are required to select a generative model, implement, and train it. Once trained to automatically learn the patterns in the original data, students are requested to analyze the relationship between the latent space variables and the generated images. **Students are encouraged to illustrate how different values (or ranges of variation) of the latent space variables impact the generated microstructure image**. The main purpose of this point is to explore if the implemented generative model can learn disentangled representations i.e. if it is capable to explicitly represent salient attributes of a data instance. Additionally, to complete the

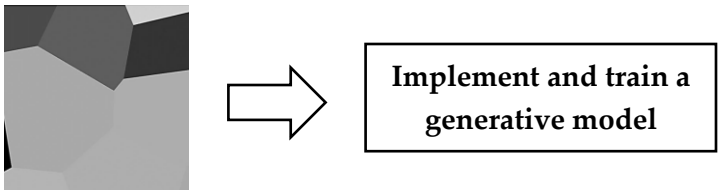
[1] Mostly used for tasks related to data compression, feature extraction or denoising.

evaluation of the results, students must generate a completely new set of 200 synthetic samples which will be evaluated indirectly by the reviewer, using a discriminative model based on a Convolutional Neural Network (CNN) that is trained as a baseline using the original data.

Step-by-step. Instructions for the Project

STEP 1

Original dataset

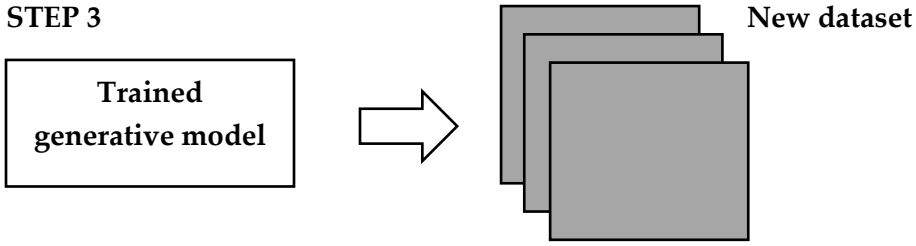


- Develop and implement a generative model (VAE or any GAN variant) capable to learn a representation of the input data. **Note: The resolution of the images can be adjusted if needed.**

STEP 2

- Explore and report in detail the observed relationship between the latent space variables and the characteristics of the generated microstructures. Illustrate different cases demonstrating **how sampling different values from the latent space affect the generated image**.

STEP 3



- Using the trained generative model, generate a representative dataset of 200 new synthetic microstructures to be submitted together with the written report and code implementation files (.ipynb and .h5)

STEP 4 (By the evaluator)

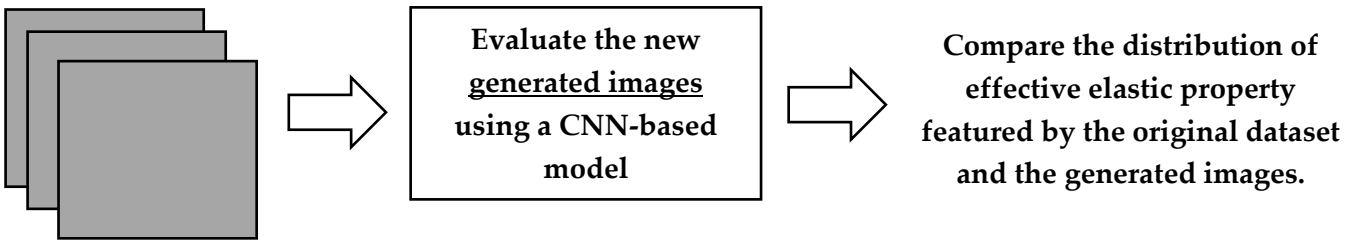


Figure 3: Steps for the implementation and posterior evaluation.

- Download all files available at https://github.com/josepabloquesadamolina/CMME2022
- Extract the .zip files before attempting to run *generative.ipynb*.
- The file called *generative.ipynb* is provided as the starting point for the project development and contains the required code for importing the input data (images), the corresponding labels (classes) and target values (Youngs Modulus, E11).
- Implement the deep generative model (e.g. VAE or any of the GAN variants). Check available online resources for generative modelling code implementation examples.
- Train the model and optimize the hyperparameters to generate highly realistic synthetic images.
- Write a final report. Include the details regarding the architecture and hyperparameters of the best performing model. Add a diagram to illustrate the architecture implemented.
- Explain the tested configuration(s) and provide a discussion of the main results and conclusions. Include the analysis regarding the observed relationship between specific variations in the latent space variables and the generated images.
- Submit (via email) the final report together with the files of the code implementation of the generative models and final model weights (*.ipynb* and *.h5*). Also include the folder with the 200 generated images from the best performing generative model.

Relevant Publications

1. https://doi.org/10.1038/s41524-020-0340-7
2. https://doi.org/10.1038/s43588-021-00045-8
3. https://doi.org/10.1038/s41598-020-70149-0
4. https://doi.org/10.48550/arXiv.1805.02791
5. https://doi.org/10.1007/s11837-020-04484-y
6. https://doi.org/10.1002/eng2.12274

References

[1] D. P. Kingma ; M. Welling. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114. Available online: https://doi.org/10.48550/arXiv.1312.6114

[2] Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *arXiv* **2014**, arXiv:1406.2661. Available online: https://arxiv.org/abs/1406.2661

[3] Kingma, D.P.; Welling, M. An introduction to variational autoencoders. *arXiv* **2019**, arXiv:1906.02691. Available online: https://arxiv.org/abs/1906.02691

[4] Karpathy, A.; Abbeel,P. ; Brockman, G.; Chen, P.; Cheung, V.; Duan,R.; Goodfellow, I.; Kingma, D.P.; Ho,J. ; Houthooft, R.; Salimans,T. ; Schulman, J.; Sutskever,I.; Zaremba,W. Generative Models. OpenAI. Available online: https://openai.com/blog/generative-models/