Exploring Variational Autoencoders for 2D Microstructure Generation

Ajith Moola, Chiran Binnu Cherian and Sourav Das

This study investigates the application of Variational Autoencoders (VAEs) to the synthesis of 2D microstructure images for computational materials design. A custom data-loading pipeline was developed in PyTorch to handle binary microstructure images and to generate paired clean/noisy examples via three distinct perturbation models: salt-and-pepper flips, structured sinusoidal grid overlays, and frequency-domain masking using FFT. The resulting dataset feeds into a denoising VAE framework that learns latent representations robust to diverse noise patterns. We further refine samples by training a Latent Diffusion Model (LDM) in the learned latent space, demonstrating that the diffusion process improves generation fidelity. Multi-modal noise augmentation and latent-space diffusion jointly yield high-quality synthetic microstructure realizations.

I. Introduction

The generation of realistic microstructural images is a critical step in linking material processing parameters to macroscopic properties. Traditional approaches rely on statistical sampling or physics-based simulation, which can be computationally expensive and may not capture the full variability of natural microstructures. Recent advances in deep generative models, particularly Variational Autoencoders (VAEs) and diffusion models, offer a data-driven alternative capable of learning compact latent representations and refining them via iterative denoising.

In this work, we explore a two-stage pipeline: a denoising VAE trained on clean/noisy image pairs, followed by a Latent Diffusion Model (LDM) operating in the VAE's latent space. We introduce three noise-injection strategies—salt-and-pepper flips, structured grid noise, and frequency-domain masking—to challenge the VAE, then leverage diffusion over the 128-dim latent vectors to further enhance sample quality. Our results indicate that this combination yields superior microstructure synthesis compared to standalone VAEs.

II. Literature Review

In materials science, the correlation between structure and property is a fundamental notion, [1, 2] with microstructures frequently acting as the principal determinants of a material's physical attributes and performance. Nonetheless, direct experimental observation and reconstruction of these microstructures are sometimes obstructed by substantial costs and technical problems [3, 4], complicating the precise assessment and prediction of material properties. A primary goal in this domain is to produce statistically representative collections of microstructural realizations to facilitate computational design and performance evaluation. To tackle this difficulty, we introduce a 2D microstructure creation framework utilizing cutting-edge generative artificial intelligence (AI) methods. This framework permits the tailored fabrication of microstructures according to precise specifications, enabling on-demand microstructure synthesis to assist diverse research and development endeavors. Various statistical methodologies have been established for microstructure synthesis, including [5] Markov Random Fields (MRFs), [6] Gaussian Random Fields (GRFs), [7] and descriptor-based microstructure reconstruction. Bostanabad et al. [8] provide a comprehensive examination of contemporary methods for microstructure reconstruction. Although these strategies have demonstrated utility, they possess certain limitations. [9] Torquato's book offers foundational insights into the characterization of random heterogeneous materials and presents a complete overview of mathematical and computational tools for microstructure modeling and production.

[10] Statistical models are computationally demanding and therefore lack scalability concerning both the size and the quantity of microstructure generations. [11] These models exhibit less flexibility and frequently necessitate particular assumptions regarding the statistical characteristics of the microstructure, including stationarity and isotropy. As a result, they may not generalize proficiently across various materials or structures. Modifying these models to integrate new restrictions or objectives can be arduous and may necessitate significant alterations to the model or the optimization procedure. [12] In recent years, sophisticated generative AI models have exhibited their formidable ability to capture complex characteristics from training data distributions. [13] These AI techniques are multifaceted and applicable

across diverse domains. Notable instances of generative AI tools comprise [14] Variational Autoencoders (VAEs), [15] Generative Adversarial Networks (GANs), [16] and Diffusion Models (DMs). [17] Although VAEs are recognized for their capacity to learn effective data representations, they frequently encounter difficulties in producing high-resolution images. [18] GANs have been effectively utilized in producing high-quality 2D microstructures; nevertheless, they do not permit user control over the created microstructures. [19] Moreover, GANs are recognized for their training instability and high computing demands, particularly in the generation of 2D structures. [20] Diffusion models are purported to generate superior-quality images compared to generative adversarial networks, indicating a potential benefit in output quality.

Nonetheless, akin to GANs, DMs necessitate substantial computer resources, especially during inference, owing to the iterative nature of the generation process. [21] Given the constraints of VAEs, DMs, and GANs, LDMs have arisen as a premier option, integrating the advantages of both VAEs and DMs. Latent Diffusion Models (LDMs) function within a latent space, a lower-dimensional representation attained using a Variational Autoencoder (VAE) in the LDM framework, which is considerably less computationally demanding than processing in pixel space. [22] This architecture allows LDMs to produce high-quality, diversified, and intricate samples, providing computational efficiency that much surpasses that of conventional DMs. Latent Diffusion Models facilitate a regulated generation process by delineating features or conditions, essential for the development of tailored microstructure designs. [23] In contrast to GANs, which may face training difficulties and mode collapse resulting in limited output diversity, LDMs exhibit superior stability during training and provide a broader range of outputs. The reliability and versatility of LDMs render them an appropriate selection for a wide range of applications in generative AI [24].

III. Methodology

A. Dataset Preparation

The dataset utilized in this study was generated from three-dimensional simulations of the Cahn-Hilliard problem, resolved using the Finite Element Method (FEM). It encompasses a diverse array of phase separation situations, meticulously represented by simulations under different conditions determined by two essential parameters: the starting volume percentage (ϕ) and the interaction parameter (χ). The Cahn-Hilliard equation characterizes a microstructure by simulating the spatial distribution of two or three components. In our dataset, ϕ is systematically modified to investigate a broad range of starting mixture compositions, elucidating the impact of early concentration heterogeneities on the dynamics of phase separation. The interaction parameter, χ, constitutes another crucial component inside the dataset. It measures the extent of attraction or repulsion among the components of the combination. A greater χ value indicates a pronounced inclination towards phase separation due to adverse interactions, whereas a lower value implies enhanced miscibility. By modifying χ, we investigate several interaction regimes, ranging from mild to significant phase-separation tendencies. This variation facilitates a comprehensive analysis of the influence of molecular interactions on the macroscopic patterns generated during phase separation. The dataset records more than 400 time-stamped snapshots of a 2D Cahn-Hilliard simulation at a resolution of 101 × 101 for each combination of ϕ and χ, offering a comprehensive temporal sequence of the phase separation process. We implement a PyTorch `Dataset` class to manage clean and noisy 2D microstructure images. Some sampled images from the dataset are shown in Fig. 1.

We also test our model on there different kinds of noise: a) Salt and Pepper noise b) Grid noise c) Freqeuncy noise. The details on how to generate and add noise is discussed below:

1. Salt and Pepper Noise

Let the original binary image be represented by a matrix I, where each element $I(i, j) \in \{0, 1\}$. Salt and pepper noise is introduced by creating a noise mask N_{sp}, of the same dimensions as I. Each element $N_{sp}(i, j)$ is a random variable drawn from a Bernoulli distribution with probability p_{flip}:

$$N_{sp}(i, j) \sim \text{Bernoulli}(p_{flip})$$

where p_{flip} is the `flip_probability`.

The noisy image I_{noisy} is generated by inverting the pixel values of I where $N_{sp}(i, j) = 1$:

$$I_{noisy}(i, j) = \begin{cases} 1 - I(i, j) & \text{if } N_{sp}(i, j) = 1 \\ I(i, j) & \text{if } N_{sp}(i, j) = 0 \end{cases}$$

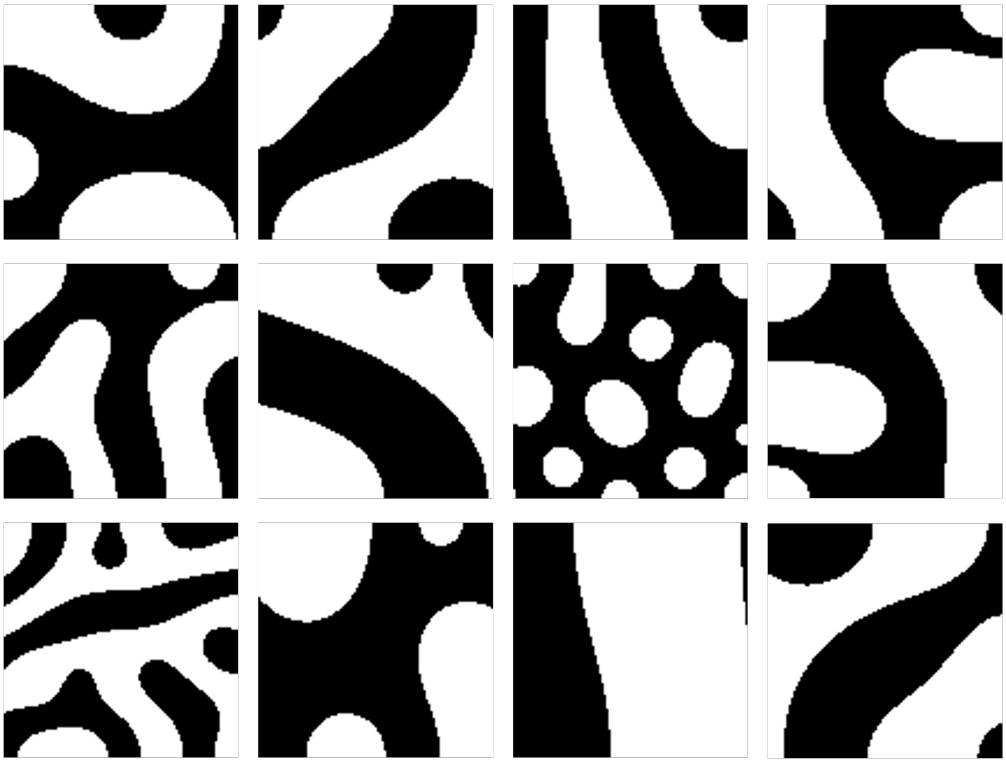

Fig. 1 Sample images from the dataset

This can also be expressed as:

$$I_{noisy}(i,j) = (I(i,j) \cdot (1 - N_{sp}(i,j))) + ((1 - I(i,j)) \cdot N_{sp}(i,j))$$

2. Grid Noise

Let the image have dimensions $H \times W$. Normalized spatial coordinates (x, y) are defined such that $x \in [0, 1]$ and $y \in [0, 1]$. If a rotation by an angle θ_g (given by `grid_rotation`) is applied, the coordinates (x, y) are first centered to $(x_c, y_c) = (x - 0.5, y - 0.5)$. The rotated coordinates (x', y') are then calculated using a standard 2D rotation matrix:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta_g) & -\sin(\theta_g) \\ \sin(\theta_g) & \cos(\theta_g) \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

If no rotation is applied, $(x', y') = (x, y)$.

A grid pattern P_g is generated using these (potentially rotated) coordinates:

$$P_g(i,j) = A_g \cdot \sin(2\pi f_g x'(j)) \cdot \sin(2\pi f_g y'(i))$$

where f_g is the grid frequency and A_g is the grid amplitude.

This pattern P_g is then added to the original image I. Let the intermediate noisy image be $I_{temp} = I + P_g$. The grid noise image then defined as:

$$I_{noisy}(i,j) = \begin{cases} 1 & \text{if } I_{temp}(i,j) > 0.5 \\ 0 & \text{if } I_{temp}(i,j) \leq 0.5 \end{cases}$$

3. Frequency Noise

Let the original image be $I(x, y)$. First, the 2D Discrete Fourier Transform (DFT) of the image is computed:

$$F(u,v) = \mathcal{F}\{I(x,y)\} = \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} I(x,y) e^{-j2\pi(\frac{ux}{H} + \frac{vy}{W})}$$

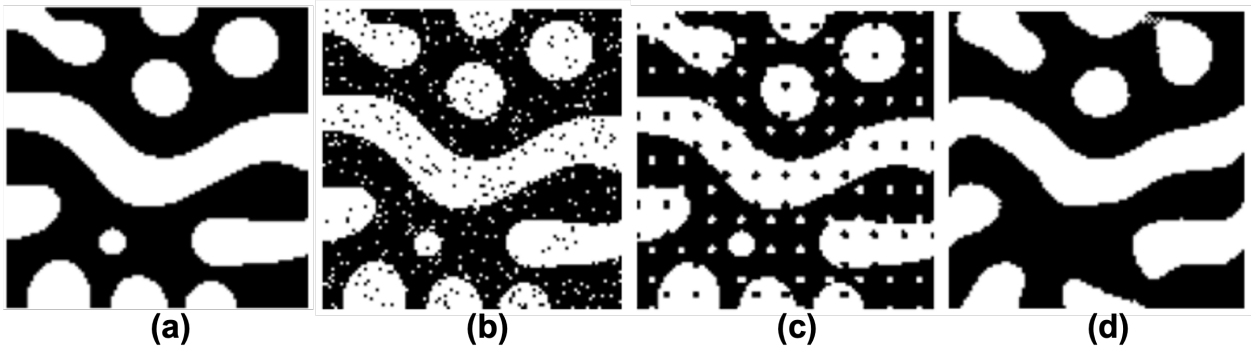

Fig. 2 Sample images of (a) Ground truth, (b) Salt and Pepper noise, (c) Grid noise and (d) Frequency noise. Adding noise in the frequency domain changes the interface between the phases of the microstructure.

The zero-frequency component is then shifted to the center of the spectrum, resulting in $F_{shifted}(u, v)$.

A frequency domain mask $M(u, v)$ is constructed. This mask has values typically between $1 - I_{fm}$ and 1, where I_{fm} is the frequency mask intensity. Let (u_c, v_c) be the coordinates of the center of the frequency domain. The normalized radial distance from the center for any frequency coordinate (u, v) is $d_{norm}(u, v) = \sqrt{(u - u_c)^2 + (v - v_c)^2}/d_{max}$, where d_{max} is the maximum possible distance from the center to the edge of the frequency domain. The mask $M(u, v)$ is designed to suppress frequencies within specific radial bands (rings). For a predefined number of rings, N_{rings}, the center of each ring k (for $k = 0, \ldots, N_{rings} - 1$) can be defined at a normalized distance $r_k = \frac{k+1}{N_{rings}+1}$. The mask value is set to $1 - I_{fm}$ if the normalized distance $d_{norm}(u, v)$ falls within a ring, i.e., if $|d_{norm}(u, v) - r_k| < w_{fm}/2$ for any ring k. Here, w_{fm} is the frequency mask width, representing the normalized width of each ring.

$$M(u, v) = \begin{cases} 1 - I_{fm} & \text{if } \exists k \in [0, N_{rings} - 1] \text{ s.t. } |d_{norm}(u, v) - r_k| < w_{fm}/2 \\ 1 & \text{otherwise} \end{cases}$$

. The mask is then applied in the frequency domain:

$$F_{filtered}(u, v) = F_{shifted}(u, v) \cdot M(u, v)$$

The result $F_{filtered}(u, v)$ is shifted back so the zero-frequency component is at the origin, let this be $F'_{filtered}(u, v)$. The noisy image in the spatial domain is obtained by applying the inverse 2D DFT:

$$I_{spatial\_filtered}(x, y) = \text{Re}(\mathcal{F}^{-1}\{F'_{filtered}(u, v)\})$$

where $\text{Re}(\cdot)$ denotes taking the real part. The resulting image $I_{spatial\_filtered}$ is then normalized and clipped to the range $[0, 1]$. If the image is binary (specified by `noise_type` containing "binary"):

$$I_{noisy}(x, y) = \begin{cases} 1 & \text{if } I_{spatial\_filtered}(x, y) > 0.5 \\ 0 & \text{if } I_{spatial\_filtered}(x, y) \leq 0.5 \end{cases}$$

.

All the three noise variations are shown in Fig. 2 along with the ground truth for comparison.

B. Model Architecture

The model employed is a Variational Autoencoder (VAE) designed for 2D image processing and subsequent denoising tasks. The VAE comprises an encoder, a decoder, and a reparameterization mechanism to sample from the learned latent space. The architecture is detailed below.

The VAE learns a probabilistic mapping from the input data x to a lower-dimensional latent space representation z, and then maps z back to the reconstructed data $\hat{x}$.

1. Encoder

The encoder network $q_\phi(z|x)$ approximates the posterior distribution over the latent variables z given an input image x. For the variational autoencoder implementation detailed in this work, we constructed an encoder architecture comprising a series of convolutional layers followed by fully connected layers to parameterize the latent distribution. The encoder processes input images of size $1 \times 101 \times 101$ through a systematic feature extraction procedure. The first convolutional layer applies 32 filters with kernel size 4×4, stride 2, and padding 1, transforming the input to a $32 \times 50 \times 50$ feature map, followed by batch normalization and ReLU activation. The second convolutional layer employs 64 filters with identical kernel dimensions, stride, and padding parameters, producing a $64 \times 25 \times 25$ feature map, again followed by batch normalization and ReLU activation. A third convolutional layer with 128 filters, kernel size 3×3, stride 2, and padding 1 further processes the representations to yield a $128 \times 13 \times 13$ feature map. The final convolutional layer applies 256 filters with kernel size 3×3, stride 2, and padding 1, resulting in a $256 \times 7 \times 7$ feature map. The resulting feature maps are flattened into a 12,544-dimensional vector, which is subsequently projected to 1024 dimensions via a fully connected layer with batch normalization and ReLU activation. This intermediate representation is then mapped to the latent space parameters through two parallel fully connected layers, one outputting the mean vector μ and the other producing the log-variance vector $\log(\sigma^2)$ of the approximated Gaussian posterior. This parameterization enables the model to learn a probabilistic mapping from the input space to a latent distribution, facilitating both reconstruction and generative capabilities.

2. Reparameterization

To enable backpropagation through the sampling process, we implement the reparameterization trick, which provides a differentiable pathway for gradient flow. A latent vector z is sampled from the learned distribution $\mathcal{N}(\mu, \sigma^2)$ by reformulating the sampling operation as $z = \mu + \epsilon \cdot \sigma$, where $\sigma = \exp(0.5 \cdot \log(\sigma^2))$ and ϵ represents a random sample from a standard normal distribution $\mathcal{N}(0, I)$. This technique effectively separates the stochastic component from the network parameters, allowing gradients to propagate through the sampling procedure during the training process while maintaining the statistical properties of the target distribution.

3. Decoder

The decoder network $p_\theta(x|z)$ reconstructs the input image from the latent representation z. The architecture mirrors the encoder structure using transposed convolutional layers to progressively increase spatial dimensions. The latent vector z is initially projected to 1024 dimensions via a fully connected layer with batch normalization and ReLU activation. A subsequent fully connected layer further maps this representation to 12,544 dimensions, followed by batch normalization and ReLU activation. The resulting vector is reshaped into a $256 \times 7 \times 7$ feature map to serve as input to the transposed convolutional layers.

The first transposed convolutional layer applies 128 filters with kernel size 3×3, stride 2, padding 1, and output padding 0, transforming the feature map to $128 \times 13 \times 13$, followed by batch normalization and ReLU activation. The second transposed convolutional layer employs 64 filters with kernel size 3×3, stride 2, padding 1, and output padding 0, yielding a $64 \times 25 \times 25$ feature map with subsequent batch normalization and ReLU activation. The third transposed convolutional layer uses 32 filters with kernel size 4×4, stride 2, padding 1, and output padding 0, producing a $32 \times 50 \times 50$ feature map. The final transposed convolutional layer utilizes a single filter with kernel size 4×4, stride 2, padding 1, and output padding 1 to reconstruct the image to its original dimensions of $1 \times 101 \times 101$. A sigmoid activation function is applied to the output to constrain pixel values to the range $[0, 1]$.

The complete VAE architecture integrates the encoder, reparameterization step, and decoder in sequence. During training, the model minimizes a composite loss function comprising a reconstruction term that quantifies the fidelity of the generated images and a Kullback-Leibler divergence term that regularizes the latent space by encouraging alignment with a standard normal prior distribution. This dual objective enables the model to simultaneously achieve accurate reconstructions and learn a structured latent space conducive to interpolation and random sampling. The complete architecture is illustrated in Fig. 3.

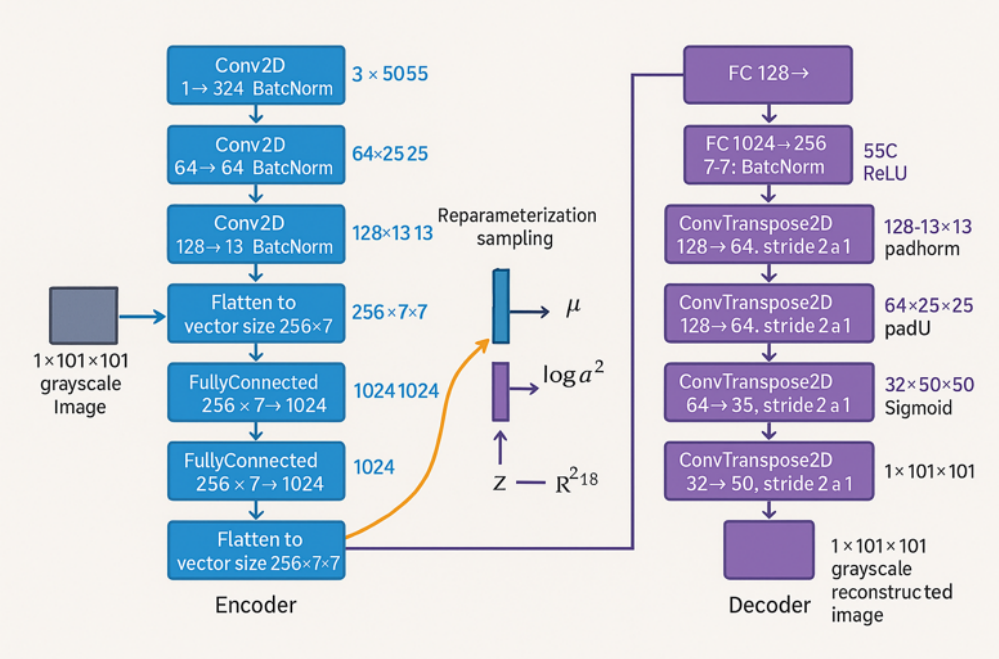

Fig. 3 Model Architecture

IV. Results and Discussion

A. Hyper Parameter Study 1: Varying Batch Size

In this study, we focused on finding the optimize parameter in terms of better validation loss and simultaneously decoding better quality images. In this 3-stage optimization process, 3 parameters have been optimized. This study investigates the effect of varying batch size on model performance. As shown in decoded sampled images in Fig. 4 and Fig. 5, a batch size of 32 yields the best validation loss, indicating better model performance compared to other ones. Fig. 6 and Fig. 7 represent the loss of training and validation during the optimization batch study.

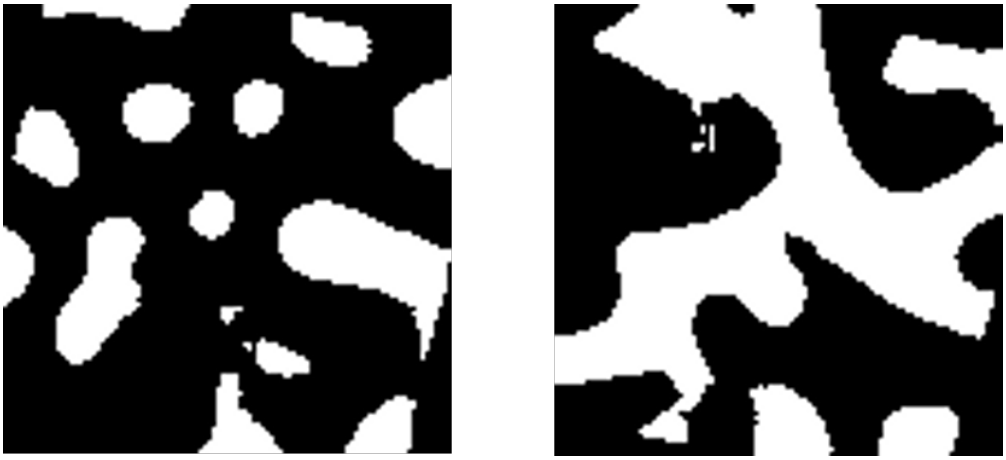

Fig. 4 Decoded Sampled Images at Latent Dimension: 128, beta: 1, Best Validation Loss(Batch Size 32)

Fig. 5 Decoded Sampled Images at Latent Dimension: 128, Worst Validation Loss(Batch Size 16)

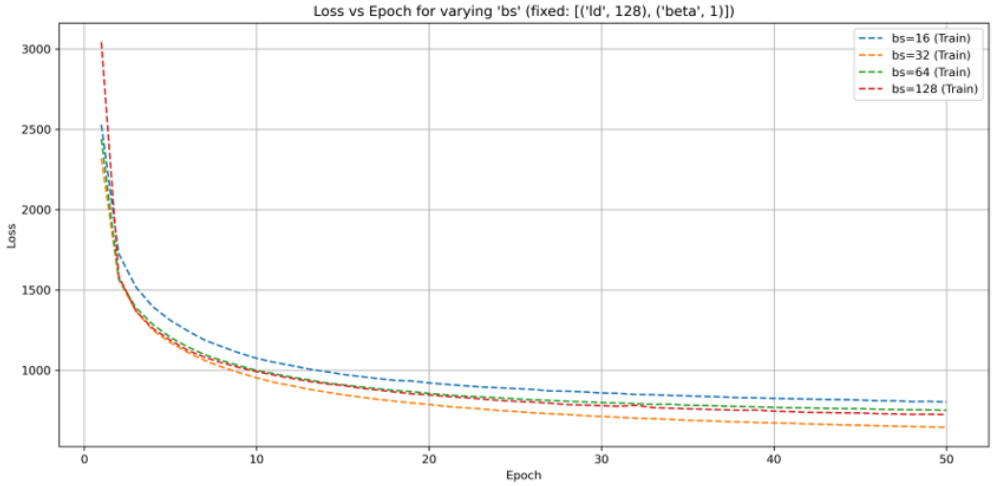

Fig. 6 Training Loss during Batch Size Optimization

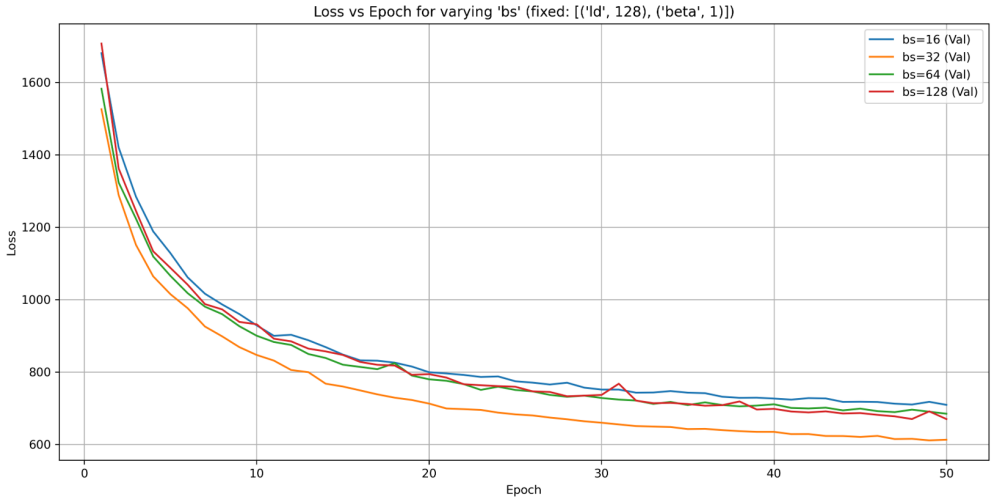

Fig. 7 Validation Loss during Batch Size Optimization

B. Hyper Parameter Study 2: Varying Latent Dimension

Optimized batch size has been used to find the best latent parameter. Latent parameters are the part of the reconstructed loss in the total Loss function. The lower the latent parameter, the better the characteristic reproduced in the decoded images. This makes the model more localized. To make a better generalization of the model, the other parameter in the total loss function, 'beta', represented as an additional weight to the KL divergence, has also been optimized, so that the model can perform better in a generalized sense. The purpose of the addition of KL divergence is to make the latent space more Gaussian distribution so that the model can work with a variety of images. As shown in decoded sampled images in Fig. 8 for LD 128 and Fig. 9 for LD 32, an LD 128 indicates better reconstruction of features. The Training loss and Validation loss are also less in the case of LD 128 as shown in Fig. 10 and Fig. 11 .

Fig. 8 Latent Dimension: 128, beta: 1, Best Validation Loss

Fig. 9 Latent Dimension: 32, beta: 1, Best Validation Loss

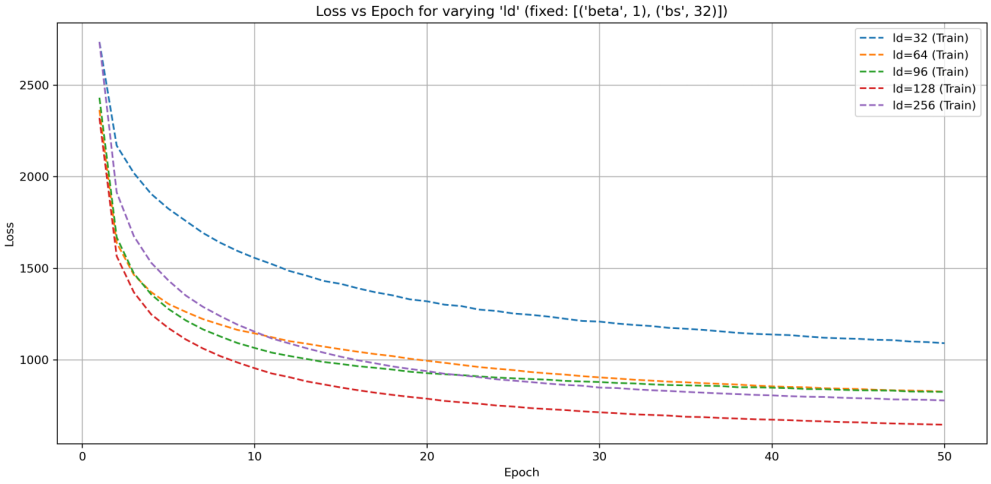

Fig. 10 Training Loss during Latent Dimension Optimization

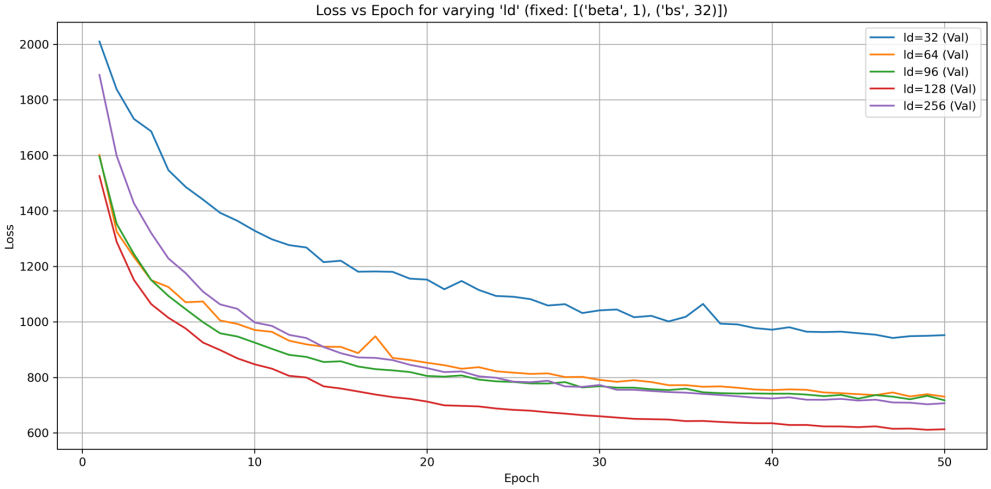

Fig. 11 Validation Loss during Latent Dimension Optimization

C. Hyper Parameter Study 3: Varying beta

Finally, Optimized batch size 32 and Latent parameter 128 has been used to find the optimize beta parameter required for generation of the model. The validation loss in Fig. 13 and the training loss in Fig. 12 show a better performance of the model with beta=1.5. The reconstructed image Fig. 14 also indicates a similar trend as compared to beta 0.25(Fig. 15)

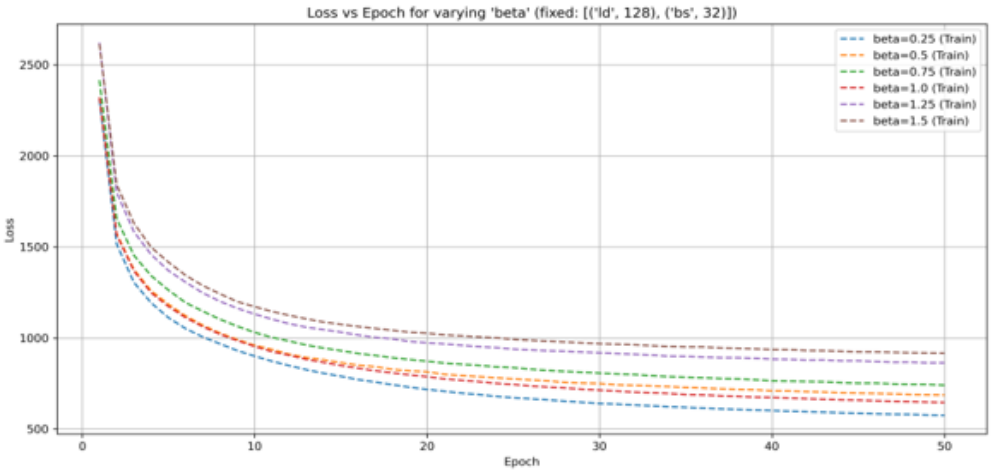

Fig. 12 Training Loss during Beta Optimization

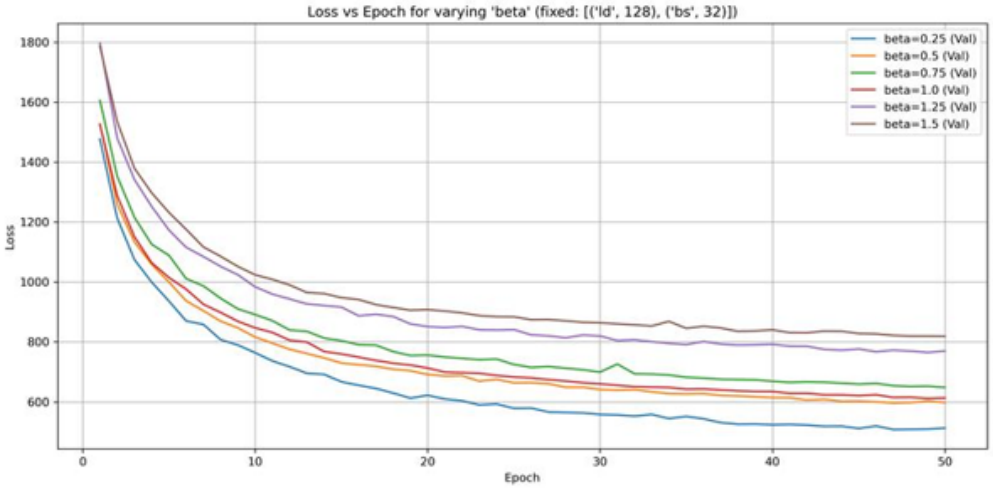

Fig. 13 Validation Loss during Beta Optimization

Fig. 14 Latent Dimension: 128, beta: 1.5, Best Validation Loss

Fig. 15 Latent Dimension: 32, beta: 0.25, Best Validation Loss

D. Salt-Pepper and Grid Noise

To identify model robustness to prior noise, we have done a salt-pepper noise and grid noise study. We added different noises in the figures to check their effectiveness in detecting and eliminating noise from the sample. The result, Fig. 16, with flip probability 0.15, shows the robustness of our model in the presence of salt and pepper noise. The generated image is able to differentiate larger features and noise, but it fails to identify the difference between smaller features and noise because the noise and the features are same scale.

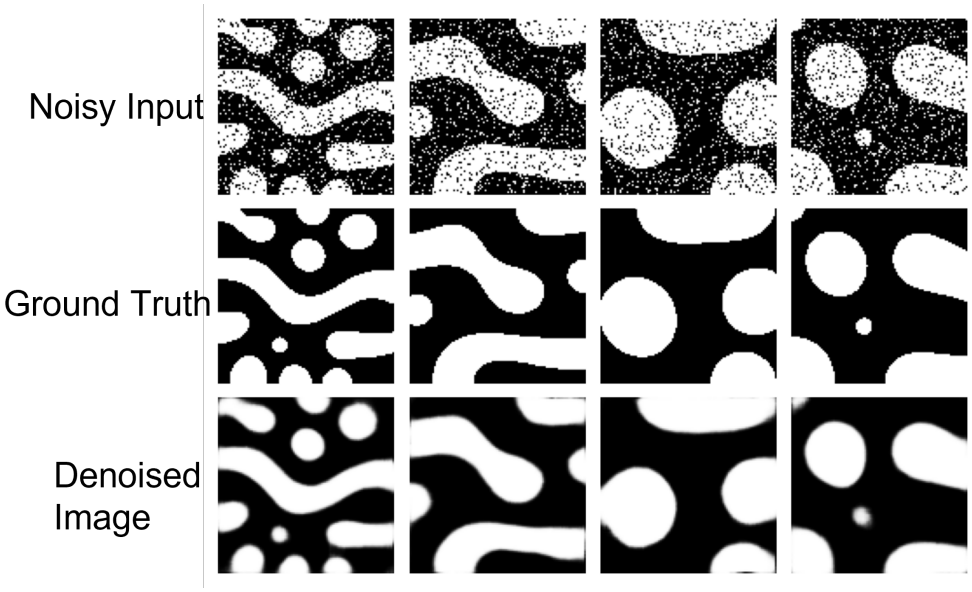

Fig. 16 Salt and Pepper Noise Filtering

Similarly, this model is also able to detect and eliminate grid noise and remove it as shown in Fig. 17. This proves the robustness and generalization of the model. In both noise cases, the ground truth and the Denoised decoded image show a similar nature, signifying the effectiveness of the model.

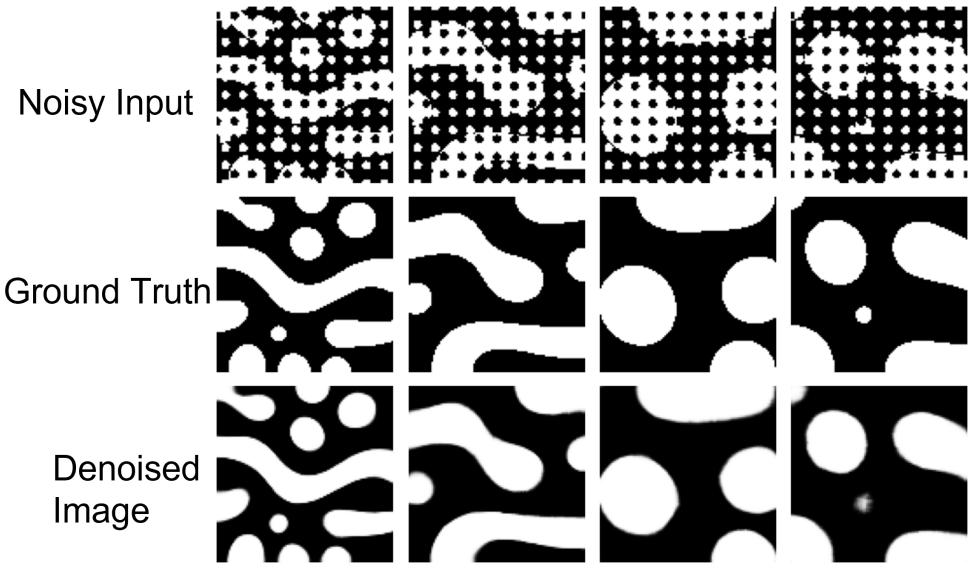

Fig. 17 Grid Noise Filtering

We also added noise in the frequency domain of the microstructure images as described in III.A.3. This resulted

in images having distorted or completely different boundaries between phases compared to the ground truth images. However, our model was able to successfully filter out most of the frequency noise from the microstructure images. This noise study proves that our VAE model can also be used for noise filtering.

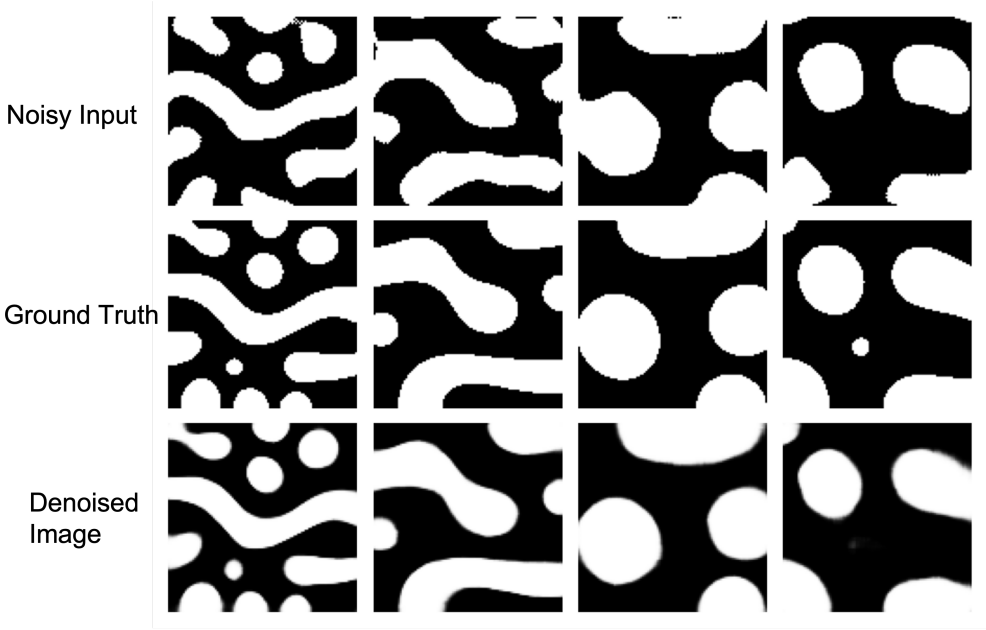

Fig. 18 Frequency Noise Filtering

V. Conclusion

Conditional microstructure production is advantageous in diverse domains such as materials science, energy storage, biomedical engineering, and additive manufacturing, facilitating meticulous control over microstructural characteristics to enhance performance, durability, and usefulness. This study introduced a flexible, scalable framework for the generation of 3D microstructures utilizing LDM. Our methodology effectively generates varied, high-resolution microstructures and offers predictive production parameters to connect computational models with experimental synthesis. This skill permits researchers and engineers to develop materials with precise properties, enhancing our comprehension of processing-structure-property linkages. Nevertheless, many restrictions warrant consideration. The training dataset is produced using physics simulators instead of experimental microstructures. Secondly, the sequential training process—initially training the VAE, subsequently the feature predictor, and ultimately the latent diffusion model—could be optimized for enhanced efficiency. Thirdly, during inference, users must show prudence when choosing conditional parameters. Unrealistic user specifications will provide microstructures that fail to meet the user requirements, a discrepancy that can be readily verified using the feature predictor network. Future research could concentrate on integrating experimental microstructure data to mitigate these constraints. Enhancing the training pipeline to facilitate parallel or integrated training of the VAE and LDM could markedly decrease the total training duration. Moreover, enhancing the interface's intuitiveness, instituting limitations on user inputs to avert unrealistic conditioning parameters, or automating parameter selection could reduce the likelihood of producing impractical microstructures, thereby rendering the framework more robust and accessible to a wider audience.

References

[1] Newnham, R. E., *Structure-property relations*, Vol. 2, Springer Science & Business Media, 2012.

[2] Li, J., Zhang, Q., Huang, R., Li, X., and Gao, H., "Towards understanding the structure–property relationships of heterogeneous-structured materials," *Scripta Materialia*, Vol. 186, 2020, pp. 304–311.

[3] Franken, L. E., Boekema, E. J., and Stuart, M. C., "Transmission electron microscopy as a tool for the characterization of soft materials: application and interpretation," *Advanced Science*, Vol. 4, No. 5, 2017, p. 1600476.

[4] Mohammed, A., and Abdullah, A., "Scanning electron microscopy (SEM): A review," *Proceedings of the 2018 international conference on hydraulics and pneumatics—HERVEX, Băile Govora, Romania*, Vol. 2018, 2018, pp. 7–9.

[5] Bostanabad, R., Bui, A. T., Xie, W., Apley, D. W., and Chen, W., "Stochastic microstructure characterization and reconstruction via supervised learning," *Acta Materialia*, Vol. 103, 2016, pp. 89–102.

[6] Jiang, Z., Chen, W., and Burkhart, C., "Efficient 3D porous microstructure reconstruction via Gaussian random field and hybrid optimization," *Journal of microscopy*, Vol. 252, No. 2, 2013, pp. 135–148.

[7] Xu, H., Dikin, D. A., Burkhart, C., and Chen, W., "Descriptor-based methodology for statistical characterization and 3D reconstruction of microstructural materials," *Computational Materials Science*, Vol. 85, 2014, pp. 206–216.

[8] Bostanabad, R., Zhang, Y., Li, X., Kearney, T., Brinson, L. C., Apley, D. W., Liu, W. K., and Chen, W., "Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques," *Progress in Materials Science*, Vol. 95, 2018, pp. 1–41.

[9] Torquato, S., et al., *Random heterogeneous materials: microstructure and macroscopic properties*, Vol. 16, Springer, 2002.

[10] Le, T., Epa, V. C., Burden, F. R., and Winkler, D. A., "Quantitative structure–property relationship modeling of diverse materials properties," *Chemical reviews*, Vol. 112, No. 5, 2012, pp. 2889–2919.

[11] Carraher Jr, C. E., and Seymour, R., *Structure—property relationships in polymers*, Springer Science & Business Media, 2012.

[12] Bandi, A., Adapa, P. V. S. R., and Kuchi, Y. E. V. P. K., "The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges," *Future Internet*, Vol. 15, No. 8, 2023, p. 260.

[13] Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., and Sun, L., "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[14] Kingma, D. P., Welling, M., et al., "Auto-encoding variational bayes," , 2013.

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial networks," *Communications of the ACM*, Vol. 63, No. 11, 2020, pp. 139–144.

[16] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S., "Deep unsupervised learning using nonequilibrium thermodynamics," *International conference on machine learning*, pmlr, 2015, pp. 2256–2265.

[17] Wang, Z., She, Q., and Ward, T. E., "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, Vol. 54, No. 2, 2021, pp. 1–38.

[18] Henkes, A., and Wessels, H., "Three-dimensional microstructure generation using generative adversarial neural networks in the context of continuum micromechanics," *Computer Methods in Applied Mechanics and Engineering*, Vol. 400, 2022, p. 115497.

[19] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., "Improved techniques for training gans," *Advances in neural information processing systems*, Vol. 29, 2016.

[20] Dhariwal, P., and Nichol, A., "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, Vol. 34, 2021, pp. 8780–8794.

[21] Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M., "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 45, No. 9, 2023, pp. 10850–10869.

[22] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10684–10695.

[23] Du, P., Parikh, M. H., Fan, X., Liu, X.-Y., and Wang, J.-X., "Conditional neural field latent diffusion model for generating spatiotemporal turbulence," *Nature Communications*, Vol. 15, No. 1, 2024, p. 10416.

[24] Herron, E., Lee, X. Y., Balu, A., Pokuri, B. S. S., Ganapathysubramanian, B., Sarkar, S., and Krishnamurthy, A., "Generative design of material microstructures for organic solar cells using diffusion models," *AI for Accelerated Materials Design NeurIPS 2022 Workshop*, 2022.