Conditional Weight-Space Diffusion for Neural Cellular Automata in Texture Synthesis

Ludek Cizinsky (377297), Andreas Christopher Theilgaard (396796)

CS-503 Final Report

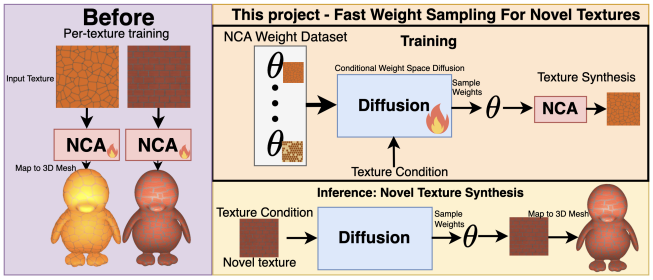

Figure 1: Current per-instance training for NCA texture synthesis (left) is costly. We explore conditional weight-space diffusion (right) to enable fast, sample-based synthesis of novel textures.

***Abstract*—Realistic texture synthesis plays a vital role in numerous downstream applications, ranging from gaming to virtual reality. Among the various approaches, Neural Cellular Automata (NCA)-based methods represent the current state-of-the-art. However, despite their many advantages, these methods require per-instance training, which hinders their scalability and limits their use in real-time settings. To overcome this limitation, we propose a conditional weight initializer that accelerates the adaptation of NCA models to novel textures at test time. Specifically, we leverage a conditional diffusion model that maps a texture image directly to the corresponding NCA weights, which can then be optionally fine-tuned to further improve synthesis quality (see Figure 1). Our results demonstrate that this approach can significantly reduce convergence time while maintaining high-quality synthesis performance, thus enabling faster and more scalable texture generation.**

I. Introduction

Textures are ubiquitous, enriching our visual interactions with the physical and virtual worlds. Combined with the recent advancements in augmented and virtual reality, *realistic* texture synthesis has currently become one of the key research areas in computer vision [1], [2].

Texture synthesis broadly involves two major tasks: 3D texture synthesis and exemplar-based synthesis. In 3D synthesis, realistic textures must be generated while respecting the geometry of objects. Although recent methods synthesizing textures directly on 3D meshes have achieved impressive results [2]–[4], their applicability remains limited to the textures and geometries seen during training. [1] addressed this by introducing MeshNCA, capable of generalizing to unseen meshes and enabling interactive control. However, MeshNCA still requires per-texture training, restricting its scalability. Exemplar-based synthesis, on the other hand, aims to generate textures that capture the visual characteristics of an input example. Building on the seminal idea by [5], [6] introduced Neural Cellular Automata (NCA) as a powerful and adaptive texture synthesis technique capable of real-time adaptation to input distortions. While promising, NCAs, including the dynamic variant (DyNCA) proposed by [7], similarly require per-texture training, significantly limiting their scalability and application to real time synthesis. To address this challenge, we draw inspiration from the recent success of hyper-network-based optimization methods [8] and propose **HyperNCA**, a diffusion model trained (conditionally) in the weight space of pre-trained NCAs. The learned priors from this model can then be leveraged for effective weight initialization (see Figure 1).

To arrive at our final approach, we investigate the following research questions (**RQ**) and hypotheses (**H**):

- **H1:** *Starting from an unconditional diffusion model that generates NCA weights, we can achieve comparable synthesis results to the original NCA model while requiring less training time.*
- **RQ1:** *Can a conditional diffusion model generate NCA weights from a given conditional signal (texture image) and achieve synthesis results comparable to the original NCA model?*
- **RQ2:** *Building on **RQ1**, what are the limitations of the conditional diffusion model? Does it generalize to out-of-distribution textures? How does it perform on ambiguous or low-information input images?*
- **RQ3:** *Can a hybrid approach, incorporating guidance during instance training, lead to faster convergence?*

To the best of our knowledge, these questions have not yet been systematically explored in the context of weight-space diffusion for neural texture synthesis.

First, our results show that using an *unconditional* diffusion model for weight initialization is no better than random weight initialization (**H1**). Second, a *conditional* diffusion model does not directly yield NCA network whose synthesis quality matches that of an NCA network trained from scratch (**RQ1**). In many cases, the model struggles to reproduce fine-grained texture details or fails to match the original color distribution (**RQ2**). Finally, we find that using the conditional diffusion model for weight initialization can significantly speed up convergence (up to $10\times$ faster in some cases) depending on the desired synthesis quality (**RQ3**). Specifically, if the goal is to reach the highest possible synthesis quality (matching the upper limit of a NCA network trained from scratch), initializing from HyperNCA-generated weights offers no clear advantage over random initialization.

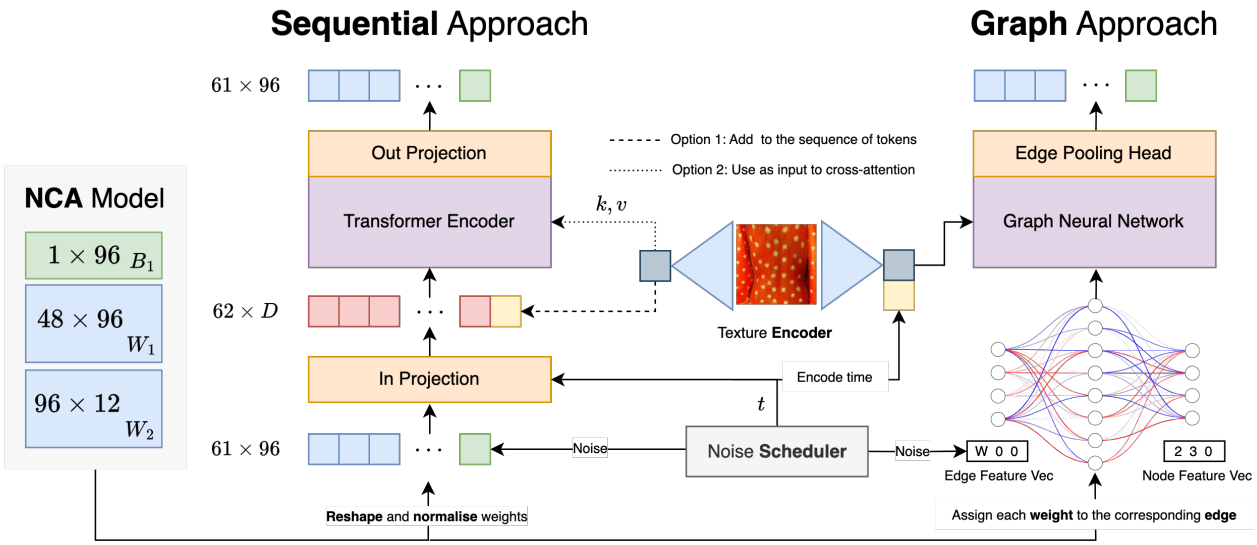

Figure 2: We propose two denoising architectures: sequential and graph-based. The sequential model tokenizes NCA weights by splitting the input layer column-wise and the output layer row-wise, yielding 61 tokens of 96 dimensions. These tokens, combined with noise from the scheduler, are denoised using a transformer encoder. Conditional information is integrated either by appending a condition token or via cross-attention. In the graph-based model, each weight is encoded as an edge feature that includes the weight value, edge type (linear), and layer index. Nodes are encoded with layer index, position within the layer, and node type (neuron). The conditional signal is added as a global feature node.

II. Related Work

Texture synthesis is a long-standing problem in computer vision and graphics. While existing models can generate high-quality dynamic textures [9], [10], they often require slow optimization and offer limited post-training control [7]. DyNCA [7] addresses these limitations by enabling faster training, higher-quality synthesis, and user-controllable properties such as motion speed and direction. In 3D settings, most methods synthesize textures in 2D and map them onto 3D surfaces via UV mapping [11], [12], which often introduces unwanted artifacts such as distortions or overlaps [1]. MeshNCA overcomes these issues by proposing UV map free NCA based method that directly operates on the vertices of a 3D mesh. Both DyNCA and MeshNCA achieve state-of-the-art results, but share a key limitation: they require per-instance training.

Hyper-networks [13], [14] generate the weights of a target network conditioned on input and have been used in diverse applications such as segmentation [15], style transfer [16], and implicit scene representations [17]. Of particular relevance are methods that learn to generate high-performing weights for downstream tasks. Several recent works combine hyper-networks with diffusion models. [18] proposed an *unconditional* diffusion model to sample neural network weights. The lacking conditioning mechanism was later addressed by conditioning on tasks or datasets, either via LoRA weight generation [19] or full model prediction [20]. Driven by the rise of Neural Implicit Representations (NIRs) [21], recent work has applied hyper-networks to generate NIR weights. Early methods directly mapped inputs to representations [22], while later approaches trained diffusion-based hyper-networks to generate NIRs for 3D objects [23], NeRFs [24], or SDFs [25]. While these works demonstrate the potential of diffusion-powered hyper-networks for generating implicit functions, none focus on textures or NCA-based synthesis. To the best of our knowledge, we are the first to explore this intersection, introducing a conditional diffusion-based hyper-network tailored to accelerate NCA texture synthesis.

III. Method

We propose a neural weight-space diffusion model to learn the conditional distribution over NCA weights. During training, Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ is added to clean NCA weights $\boldsymbol{x}$ to obtain noisy versions $\boldsymbol{x}_t$, and the model is trained to denoise $\boldsymbol{x}_t$ back to $\boldsymbol{x}$. At inference time, the model samples weights conditioned on a target texture. This sampled weights can then be used for initializing the NCA model, which then synthesizes textures in an iterative fashion. We explore two parameterizations of the denoising model: (1) a sequential transformer that tokenizes the weight matrices and processes them as a sequence, and (2) a graph-based model that treats the NCA as a graph and operates directly on its weight structure without tokenization. A high-level overview of both architectures is shown in Figure 2.

A. Sequential Approach

Given a trained NCA model with parameters $\boldsymbol{x} = \{W_1 \in \mathbb{R}^{96\times48}, b_1 \in \mathbb{R}^{96}, W_2 \in \mathbb{R}^{12\times96}\}$, we tokenize the weights by splitting W_1 column-wise (48 tokens), W_2 row-wise (12 tokens), and treating b_1 as a single token. This results in 61 tokens in total, $\boldsymbol{x} \in \mathbb{R}^{61\times96}$. Noise is added to each tokenized sample using the EDM scheduler [26], and denoising is performed with an encoder-only transformer. Following [8], each token is projected into the transformer's hidden space via a trainable MLP. The model is trained using a noise-weighted MSE loss between the denoised and original weights. To condition the model on a texture input, we use an encoder $\mathcal{E} : \mathbb{R}^{3\times128\times128} \to \mathbb{R}^c$. The resulting feature vector is either appended as an additional token or fused via cross-attention within the transformer (Figure 2).

B. Graph-Based Approach

To apply a message-passing mechanism to NCA weights, we represent the model as a graph following the method of [27] (see Figure 2, right). Each neuron is encoded as a node with a 3D feature vector capturing its layer index, position within the layer, and type (constant in our case). Edges represent weights and are annotated with a 3D vector encoding the weight value, layer index, and a fixed edge type. Bias terms are treated as separate nodes connected to all neurons in their respective layer. We concatenate the diffusion timestep embedding to each edge's attributes and incorporate the encoded target texture as a global graph feature, $\mathbf{u}$, connected to all nodes. Using this graph representation, we apply message passing as defined in [27]:

$$\mathbf{v}_i \leftarrow \text{MLP}_2^v\left(\mathbf{v}_i, \sum_{j,\, e_{(i,j)}\in E_{(i,j)}} \text{MLP}_1^v(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{(i,j)}, \mathbf{u}), \mathbf{u}\right)$$

$$\mathbf{e}_{(i,j)} \leftarrow \text{MLP}^e(\mathbf{v}_i, \mathbf{v}_j, \mathbf{e}_{(i,j)}, \mathbf{u})$$

$$\mathbf{u} \leftarrow \text{MLP}^u\left(\sum_i \mathbf{v}_i, \sum_{e\in E} \mathbf{e}, \mathbf{u}\right)$$

Finally, we attach a classification head to the GNN's edge outputs to predict the denoised weights.

IV. Experimental Setup

Dataset & Metrics. We use a dataset provided by Ehsan Pajouheshgar containing over 30,000 pairs of texture images and corresponding NCA weights. The dataset is a combination of DTD textures [28] and textures gathered from Flickr. Following related work [4], [29] we use Fréchet Inception Distance (FID) [30] as well as LPIPS [31] and PSNR to assess the fidelity of the generated images.

Implementation Details. Unless otherwise stated, we use the Gram texture encoder, Adam optimizer [32] with a learning rate of 2×10^{-4}, batch size 256, hidden dimension 256, and the EDM scheduler [26]. All models are trained for 1000 epochs on an NVIDIA V100 GPU.

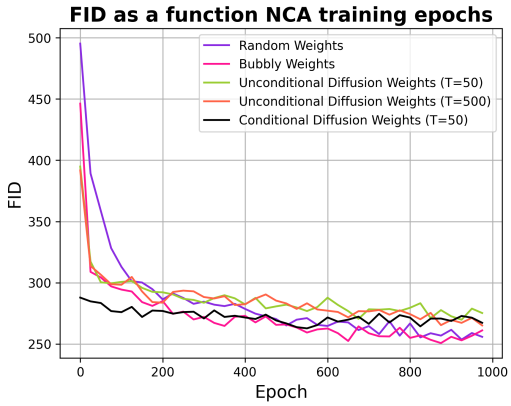

Figure 3: Average FID per epoch (lower is better). Training is conducted on 100 randomly sampled textures, each NCA model is trained for 1000 epochs.

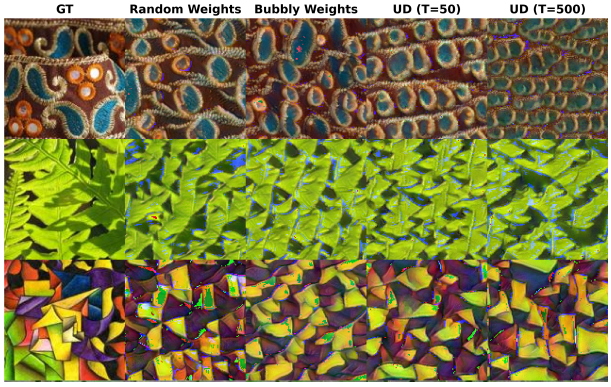

Figure 4: Synthesis results at epoch 975 for different weight initializations (columns) and input patterns (rows). UD denotes unconditional diffusion weights.

V. Results

In this section we present the results of our experiments. We show the results in chronological order based on the research questions and hypotheses outlined in the introduction. In section V-A we answer and show the results related to **H1**. In section V-B we present the results for **RQ1** and **RQ2**. Finally, in section V-C we show the results related to **RQ3**.

A. *By initializing an NCA model with weights sampled from an unconditional diffusion model, can we achieve similar synthesis performance with reduced training time?*

We train an unconditional diffusion baseline using the EDM sampler for 1000 epochs (256,000 samples). Each NCA model is initialized from pretrained weights specifically trained to synthesize a bubble pattern, hypothesized to accelerate convergence toward new target textures. We compare this approach against NCA models initialized either randomly or from weights sampled from the unconditional

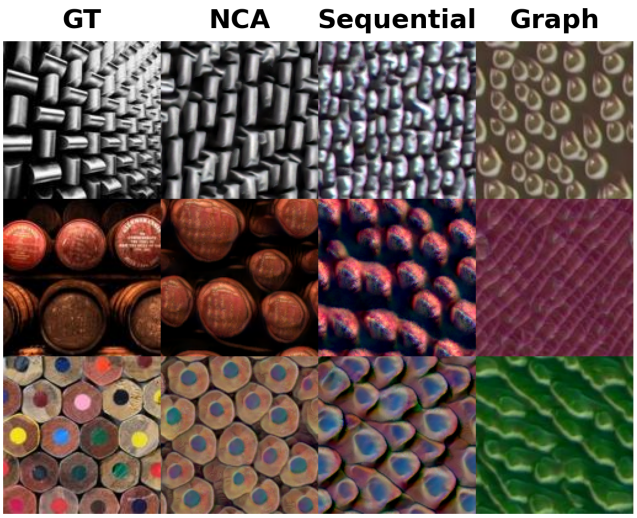

Figure 5: Synthesis results for the baseline settings of sequential and graph based approaches compared to the ground truth texture and NCA synthesised images.

Type	FID ↓	PSNR ↑	LPIPS ↓
Sequential	**5.79**	0.95	**0.029**
Graph	197.08	**10.86**	0.60

(a) Conditional diffusion results

NCA Weights	FID< 285		FID< 270	
	Epoch	Time (s)	Epoch	Time (s)
Random Weights	275	2062.5	500	3750.0
Bubbly Weights	150	1125.0	325	2437.5
Uncond. Diff (T=50)	300	2260.0	925	6947.5
Cond. Diff (T=50)	25	197.5	475	3572.5

(b) Training time and epochs for FID thresholds

Table I: (a) Conditional diffusion results for NCA model predictions comparing sequential and graph based approach. (b) Training time and epochs required for different NCA weight initializations to reach specific FID thresholds.

diffusion model, testing different numbers of diffusion inference steps (50, 500).
Figure 3 indicates that initialization from either random or bubble-pattern trained weights yields superior convergence (lower FID) compared to weights sampled from the unconditional diffusion model. We also note the same behaviour and pattern for other perceptual metrics like PSNR and LPIPS (see appendix VIII). Diffusion-based initialization might position the NCA models in suboptimal local minima, hindering effective convergence and introducing additional computational overhead. The relatively high FID scores can be partially attributed to the presence of non-texture images from Flickr in the dataset and the limited representational capability of the selected NCA architecture. This can be seen qualitatively in Figure 3. Lastly, we note the small sample size of 100, due to the high training cost (7.5 minutes per model), may introduce variance and limit the representativeness of the reported metrics.

B. Is it possible to learn a conditional weight diffusion model achieving comparable synthesis results to the original NCA model?

Both quantitative (Table Ia) and qualitative results (Figure 5) show that our best-performing model (sequential) does not fully match the synthesis quality of a corresponding NCA model trained from scratch. As illustrated in Figure 5, HyperNCA captures overall color distributions reasonably well but struggles to reproduce fine-grained texture details and shapes. We hypothesize that this limitation stems from aggressive compression of the conditioning signal from a 128×128 RGB input to a lower-dimensional embedding. This motivates further investigation into alternative conditioning strategies, as discussed in our ablation study. The graph-based model performs worse (except PSRN), often failing entirely or only approximating color. We attribute this to the model's significantly smaller capacity: 500K parameters compared to 29M in the sequential model, a difference of several orders of magnitude. We attribute the higher PSNR score of the graph-based model to its tendency to produce overly smooth textures. As shown in Figure 5 (middle row), this results in outputs that are closer to the ground truth at a *pixel* level, despite lacking perceptual and structural fidelity.

C. Can a hybrid approach utilizing guidance during instance training lead to faster convergence?

In Figure 3, we present results from a hybrid approach where NCA training is initialized with weights sampled from the baseline conditional diffusion model. We observe that initializing the NCA model with conditionally sampled weights produces reasonably good textures early in training, as reflected in the FID scores. This is further supported by Table Ib, which shows that if we accept any model achieving FID < 285, using conditionally sampled weights can reduce compute cost by up to 83% compared to the next best option, which is initializing with "bubbly" weights.

However, it is important to note that conditionally sampled weights remain suboptimal for achieving the highest synthesis quality. Similar to the unconditional case, they yield slightly higher FID scores compared to both random and bubbly initializations. For instance, as shown in Table Ib, if the target is FID < 270, bubbly initialization is preferred.

D. Ablation Studies

We conduct ablation studies for both proposed methods, starting with the sequential model followed by the graph-based approach. All results are averaged over 3328 samples using 100 diffusion and 50 NCA steps.

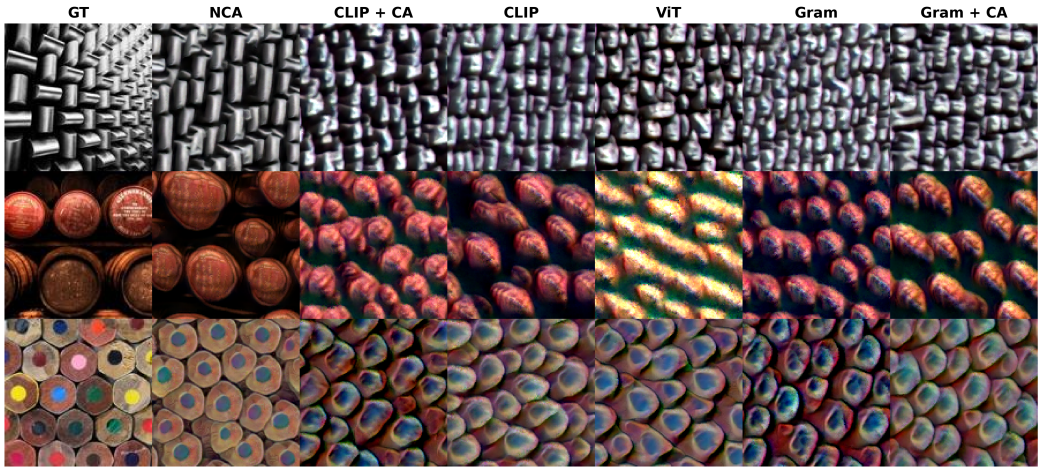

Figure 6: Synthesis results for various ablations of the sequential encoder (columns) across different textures (rows).

Scheduler	Encoder	Cross Attn	FID ↓	PSNR ↑	LPIPS ↓
EDM	Gram	✗	5.79	**0.95**	**0.029**
EDM	CLIP	✗	5.81	0.94	**0.029**
EDM	Gram	✓	6.01	0.94	**0.029**
EDM	CLIP	✓	6.06	0.94	0.030
EDM	ViT	✗	5.85	0.93	0.030
EDM	ViT	✓	5.93	0.94	0.030
DDIM	Gram	✗	**5.55**	0.94	**0.029**

Table II: Sequential model ablation. Best performance per metric is in **bold**.

Encoder	CDIM	TDIM	FID ↓	PSNR ↑	LPIPS ↓
Gram	512	6	197.08	**10.86**	0.60
Gram	512	32	**195.98**	10.75	0.60
CLIP	512	6	199.54	10.80	0.60
CLIP	128	6	204.75	10.28	**0.59**

Table III: Graph model ablation. Best performance per metric is in **bold**.

Sequential Model. In Table II we show the results of using different texture encoders either Gram, ViT or CLIP as well as conditioning on the texture using either the cross-attention mechanism in the transformer block or concatenating the condition to the other tokens. We initially expected stronger encoders to improve performance. Table II shows, however, that simply inserting the Gram embedding yields the best scores, although by a narrow margin. We also expected cross-attention would lead to better results as it enables more precise alignment between condition and latent features and is common practice adopted in models like Stable Diffusion [33]. Qualitative examples (Figure 6) confirm that NCA trained from scratch still most closely reproduces the target image. All tested conditional diffusion variants struggle with color fidelity and fine-grained patterns. Finally, we observe a minor drop in FID score using DDIM scheduler compared to EDM. Yet overall, the metrics remain largely unchanged, indicating scheduler's low impact on the results.

Graph Model. Table III summarizes our ablation study on the impact of time embedding size, texture encoder type, and conditional embedding dimension. For the sequential model, replacing the Gram encoder with CLIP shows no significant performance gain. Likewise, increasing the time embedding dimension does not improve results. As expected, reducing the conditional embedding size leads to the worst results.

VI. Conclusion & Limitations

Our results show that conditional weight-space diffusion can accelerate NCA model training for unseen textures, but only up to a certain synthesis quality. For fine-grained texture details, it offers no advantage over random initialization. Among the two approaches, the sequential model outperforms the graph-based variant, a somewhat surprising outcome given the naive tokenization of NCA weights in the sequential setup. However, this performance gap is likely due to model capacity: the graph-based model has only 0.5M parameters compared to 29M in the sequential model. Moreover, our graph model uses a simple GNN architecture with MLPs for combining node, edge, and global features. Thus, we cannot conclusively determine that the graph-based approach is inferior. Future work could explore more expressive GNN architectures and improved conditioning mechanisms, as we did in the sequential ablations. Another direction would be to focus more on hybrid methods using learned diffusion priors to guide or accelerate NCA model training. Finally, our study is limited to a simple MLP-based NCA. For more complex architectures with more layers, the sequential model's tokenization strategy breaks down due the number of parameters, which could make the graph-based approach a more promising candidate. Yet, this is a question we leave for future investigation.

VII. Author Contribution Statement

Ludek set up the initial training pipeline for the unconditional model, including data loading, checkpointing, logging, the diffusion scheduler, and the transformer-based denoiser. His main focus was on implementing and evaluating the graph-based method. Andreas implemented the NCA training and evaluation pipeline, integrated the DDIM scheduler, and added the conditional diffusion model by incorporating cross-attention and adding support for Gram, CLIP, and ViT texture encoders. Andreas conducted all experiments for the sequential model and evaluated the hybrid approach. For the final report, Ludek wrote the introduction, related work, paragraph about graph based approach in the methods and conclusion & limitations sections. Andreas completed the result section, while both authors reviewed and finalized all sections of the report. We both contributed equally to designing Figure 2, which illustrates our two proposed approaches.

References

[1] E. Pajouheshgar, Y. Xu, A. Mordvintsev, E. Niklasson, T. Zhang, and S. Süsstrunk, "Mesh neural cellular automata," *ACM Trans. Graph.*, vol. 43, no. 4, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3658127

[2] A. Bokhovkin, S. Tulsiani, and A. Dai, "Mesh2tex: Generating mesh textures from image queries," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[3] K. C. Dharma, C. T. Morrison, and B. Walls, *Texture Generation Using a Graph Generative Adversarial Network and Differentiable Rendering*. Springer Nature Switzerland, 2023, p. 388–401.

[4] D. Z. Chen, Y. Siddiqui, H.-Y. Lee, S. Tulyakov, and M. Nießner, "Text2tex: Text-driven texture synthesis via diffusion models," 2023. [Online]. Available: https://arxiv.org/abs/2303.11396

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 262–270.

[6] E. Niklasson, A. Mordvintsev, E. Randazzo, and M. Levin, "Self-organising textures," *Distill*, 2021, https://distill.pub/selforg/2021/textures.

[7] E. Pajouheshgar, Y. Xu, T. Zhang, and S. Süsstrunk, "Dynca: Real-time dynamic texture synthesis using neural cellular automata," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[8] W. Peebles, I. Radosavovic, T. Brooks, A. Efros, and J. Malik, "Learning to learn with generative models of neural network checkpoints," *arXiv preprint arXiv:2209.12892*, 2022.

[9] C. M. Funke, L. A. Gatys, A. S. Ecker, and M. Bethge, "Synthesising dynamic textures using convolutional neural networks," *CoRR*, vol. abs/1702.07006, 2017. [Online]. Available: http://arxiv.org/abs/1702.07006

[10] J. Xie, S.-C. Zhu, and Y. N. Wu, "Synthesizing dynamic patterns by spatial-temporal generative convnet," 2017. [Online]. Available: https://arxiv.org/abs/1606.00972

[11] P. Guerrero, M. Hašan, K. Sunkavalli, R. Měch, T. Boubekeur, and N. J. Mitra, "Matformer: a generative model for procedural materials," *ACM Transactions on Graphics*, vol. 41, no. 4, p. 1–12, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1145/3528223.3530173

[12] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or, "Texture: Text-guided texturing of 3d shapes," 2023. [Online]. Available: https://arxiv.org/abs/2302.01721

[13] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," 2016. [Online]. Available: https://arxiv.org/abs/1609.09106

[14] V. K. Chauhan, J. Zhou, P. Lu, S. Molaei, and D. A. Clifton, "A brief review of hypernetworks in deep learning," *Artificial Intelligence Review*, vol. 57, no. 9, Aug. 2024. [Online]. Available: http://dx.doi.org/10.1007/s10462-024-10862-8

[15] Y. Nirkin, L. Wolf, and T. Hassner, "Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation," 2021. [Online]. Available: https://arxiv.org/abs/2012.11582

[16] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. H. Bermano, "Hyperstyle: Stylegan inversion with hypernetworks for real image editing," 2022. [Online]. Available: https://arxiv.org/abs/2111.15666

[17] S. R. Maiya, A. Gupta, M. Gwilliam, M. Ehrlich, and A. Shrivastava, "Latent-inr: A flexible framework for implicit representations of videos with discriminative semantics," 2024. [Online]. Available: https://arxiv.org/abs/2408.02672

[18] K. Wang, D. Tang, B. Zeng, Y. Yin, Z. Xu, Y. Zhou, Z. Zang, T. Darrell, Z. Liu, and Y. You, "Neural network diffusion," 2024. [Online]. Available: https://arxiv.org/abs/2402.13144

[19] X. Jin, K. Wang, D. Tang, W. Zhao, Y. Zhou, J. Tang, and Y. You, "Conditional lora parameter generation," 2024. [Online]. Available: https://arxiv.org/abs/2408.01415

[20] B. Soro, B. Andreis, H. Lee, W. Jeong, S. Chong, F. Hutter, and S. J. Hwang, "Diffusion-based neural network weights generation," 2024. [Online]. Available: https://arxiv.org/abs/2402.18153

[21] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2020. [Online]. Available: https://arxiv.org/abs/2003.08934

[22] Y. Chen and X. Wang, "Transformers as meta-learners for implicit neural representations," 2022. [Online]. Available: https://arxiv.org/abs/2208.02801

[23] Z. Erkoç, F. Ma, Q. Shan, M. Nießner, and A. Dai, "Hyperdiffusion: Generating implicit neural fields with weight-space diffusion," 2023. [Online]. Available: https://arxiv.org/abs/2303.17015

[24] B. Sen, G. Singh, A. Agarwal, R. Agaram, K. M. Krishna, and S. Sridhar, "Hyp-nerf: Learning improved nerf priors using a hypernetwork," 2023. [Online]. Available: https://arxiv.org/abs/2306.06093

[25] G. Chou, Y. Bahat, and F. Heide, "Diffusion-sdf: Conditional generative modeling of signed distance functions," 2023. [Online]. Available: https://arxiv.org/abs/2211.13757

[26] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," 2022. [Online]. Available: https://arxiv.org/abs/2206.00364

[27] D. Lim, H. Maron, M. T. Law, J. Lorraine, and J. Lucas, "Graph metanetworks for processing diverse neural architectures," 2023. [Online]. Available: https://arxiv.org/abs/2312.04501

[28] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[29] X. Yu, Z. Yuan, Y.-C. Guo, Y.-T. Liu, J. Liu, Y. Li, Y.-P. Cao, D. Liang, and X. Qi, "Texgen: a generative diffusion model for mesh textures," *ACM Transactions on Graphics*, vol. 43, no. 6, p. 1–14, Nov. 2024. [Online]. Available: http://dx.doi.org/10.1145/3687909

[30] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018. [Online]. Available: https://arxiv.org/abs/1706.08500

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018. [Online]. Available: https://arxiv.org/abs/1801.03924

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022. [Online]. Available: https://arxiv.org/abs/2112.10752

VIII. Appendix

A. PSNR and LPIPS for weight initialisation experiments

Figure 7 shows the evolution of average PSNR and LPIPS metrics over training epochs for the weight initialization experiments discussed in Section V-B.

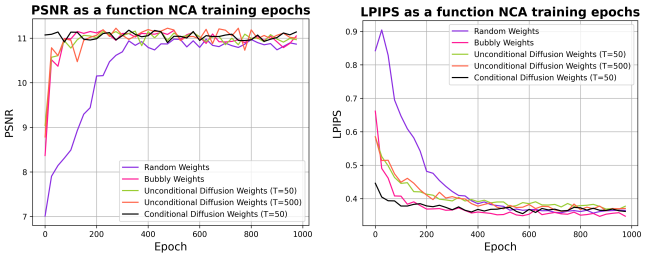

Figure 7: Average PSNR ↑ (left) and LPIPS ↓ (right) per epoch. Training is conducted on 100 randomly sampled textures; each NCA model is trained for 1000 epochs.