NeF-GPT: Autoregressive Generation of Neural Field Weights with Decoder-Only Transformer

Luca Fanselau
TUM Munich
go68vog@mytum.de

Luis Muschal
TUM Munich
go98zoh@mytum.de

https://github.com/nef-gpt/adl4cv.git

Abstract

The absence of a clear structure of implicit neural fields (NeF) makes it difficult to apply generative modeling directly to synthesize new data. On the other hand they have shown a remarkable success in compressed scene representation. To this end, we propose a novel approach for generating Multi Layer Perceptron (MLP)-weights of neural fields in an autoregressive transformer-based fashion. We demonstrate the effectiveness of our approach on the MNIST dataset as well as 3D meshes from ShapeNet and evaluate it against results of state-of-the-art diffusion-based methods.

1. Introduction

Recent successes in neural fields for compressed scene representation and autoregressive transformer models have been remarkable. Additionally, first approaches towards generating novel neural fields using diffusion models [2] and unconditional triangle-mesh generation using transformers [9] have been proposed. Challenges persist, however, due to the unstructured nature of implicit neural fields and the mismatch between the continuous MLP-weights and the discrete vocabulary typically used by transformers. Therefore, the main problem lies in finding a representation of the MLP-weights that can be used to train sequence-to-sequence architectures. The challenge therefore is two-fold: (1) finding a strategy to reduce the structural difference between neural fields to make them interpretable as sequences, and (2) find a discretization method to transform the generation task into a token prediction task.

To address these challenges, we propose a novel pipeline to handle the training of neural fields, the tokenization of these weights and the generation of novel neural fields from the same distribution using an autoregressive transformer-based model.

2. Related Works

Implicit neural fields and Diffusion Models

Recent advancements have demonstrated the effectiveness of implicit neural fields in representing high-fidelity 3D geometries and radiance fields. For instance, DeepSDF [7] encodes the shapes of objects as signed distance functions using a multi-layer neural network, and NeRF uses MLPs to encode radiance fields for photorealistic rendering from novel views [6]. Additionally, methods like Fourier features and periodic activation functions have been proposed to improve the representation of complex signals by addressing the bias towards learning low-frequency details in standard MLP [10, 11]. Our method aims to generate novel implicit neural fields that represent the implicit signal of both image data and the surface of 3D structures.

Transformer-Based 3D Structure Generation

Transformer architectures have shown promise in generating 3D structures. MeshGPT [9], for example, uses a decoder-only transformer to autoregressively generate triangle meshes, representing them as sequences of geometric embeddings. This approach has demonstrated improvements in mesh generation quality, emphasizing the capability of transformers to handle complex geometric data efficiently. Adapting this technique, our pipeline uses a GPT like transformer but changes the domain from triangle meshes to MLP-weights.

Diffusion Models in Generative Modeling

Diffusion models have been emerging in the recent past and shown promising results for novel generation tasks. Specifically, HyperDiffusion [2] operates directly on the MLP-weights of neural fields, enabling high-fidelity synthesis of 3D and 4D shapes. This method leverages a transformer-based architecture to model the diffusion process, achieving state-of-the-art performance in generating compact and coherent implicit neural fields. Influenced by this work we show the possibility of generating novel implicit neural field weights using an autoregressive process.

3. Methods

NeF-GPT deploys a transformer-based architecture to autoregressively and unconditionally generate novel implicit neural fields encoded by MLPs. It operates on MLP-weights directly. The training process includes two steps, as illustrated below.

In the first step, neural field overfitting, detailed in Figure 4 a collection of MLPs are optimized so that each MLP represents an implicit neural field of a data sample. This step involves training on either the MNIST or ShapeNet datasets. The optimized MLP-weights are then processed into sequences and tokenized to work with a transformer-based architecture for generation.

In the second step, the tokenized MLP-weights are used to train a decoder-only transformer architecture.

After training, the transformer can generate new MLP-weights by sampling from the trained model. Generated 3D shapes can be visualized and further processed using techniques like Marching Cubes.

4. Experiments

To evaluate the proposed methods, experiments were structured to progress from simpler 2D problems to more complex 3D scenarios. Initially, training was conducted on the MNIST dataset to establish a functional pipeline. Upon achieving satisfactory results, the approach was then extended to 3D meshes.

MNIST

Training Neural Fields
The Neural fields (NeF) were first trained to resemble the pixel-brightness from the individual MNIST images. The x-, y-input is first encoded using a sinusoidal positional encoding to 18 inputs. The neural network that is used to learn each image consisted of a hidden layer with 16 neurons, and a single output neuron. ReLU activation and sigmoid activation is deployed for the hidden neurons and output neuron, respectively.

Regression Transformer
Due to the continuous nature of the NeF-weights as a first experiment a transformer architecture was adapted to work with continuous weights of the trained NeF. The process involved flattening MLP-weights which resulted in a sequence length of 561, adjusting the Transformer output by changing the head to output a one-dimensional weight prediction and using MSE-loss instead of cross-entropy loss to optimize the architecture.

Permutation Invariance
One of the first issues that were encountered is the unstructured nature on neural networks which can result in different MLP-functions that resemble the same underlying function [3] making it difficult to transform them to consistently ordered sequences. To enforce a similar structure the neural fields are conditioned using weight initialization. Hereby an already fitted NeF is used to initialise all weights that are then fitted and later used to train the transformer.

Tokenization
Additionally due to the deterministic nature of the deployed regression transformer and the absence of special tokens we were not able to generate novel NeFs. To introduce the possibility of generating novel sequences the weights are discretized, and special tokens are introduced. The flattened weights are discretized using vector quantize and a vocabulary size of 245. To accelerate and improve the training the codebook are initialized using kMeans, and optimized with an MSE-loss. Furthermore, eleven special tokens are introduced. An SOS-token to indicate the start of sequence and ten different conditioning token to indicate the MNIST-NeF label of the inferred NeF-weights.

Traditional Transformer
The final transformer was based on a decoder-only GPT implementation [5] and trained using an embedding size of 240, 20 heads, 12 attention block layers and a context length of 562. The flattened, tokenized sequence is fed in as a whole. The start of sequence token is followed by the conditioning token and then the MLP-weight sequence. The transformer was trained for 40000 iterations with a batch size of 64 and a learning rate of 3e-3. To improve training cosine learning rate scheduler was used.

Preliminary Results on MNIST
In Figure 1 the final results of the MNIST transformer are shown. For the visualization the images were generated using the ten different conditioning tokens. They are shown next to their three nearest neighbors from the MNIST dataset. These results demonstrated the feasibility of proposed pipeline to approach the 3D generation task.

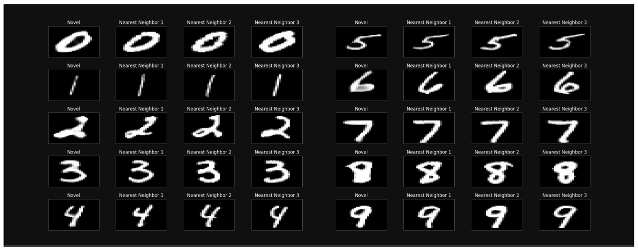

Figure 1. Autoregressive Generation of Neural Field Weights for MNIST

ShapeNet

The approach was extended to 3D meshes from the ShapeNet dataset. A subset of around 4k meshes from a single class, namely airplanes, was selected. The meshes were first converted to point clouds representing signed distance values for every point. The point clouds were then used to train the implicit neural fields. As with the MNIST Data a single sample from the dataset was overfitted and then used to initialize the other samples.

Training Neural Fields

To handle the higher complexity of the 3D shapes in comparison to the 2D MNIST images, the neural network architecture of the implicit neural fields was extended. The hidden layers were increased to 32 neurons and two additional hidden layers were added. The input of x, y, z coordinates is again encoded using a sinusoidal positional encoding to 15 inputs. The output of the neural network is a single neuron with a tanh activation function.

Tokenization:

To enable the generation of novel 3D shapes, a learned vocabulary was implemented. The MLP-weights were flattened to a sequence length of 3712 and a 1D Vector Quantization was applied. The codebook was initialized using kMeans and optimized with L2 loss. As with MNIST a SOS-token was introduced to enable the generation of novel shapes.

Mode Collapse:

During early training of the Transformer Model we experienced mode collapse where a local minimum to the Cross Entropy Loss formulation of the Transformer training was found in predicting only a single mode of the true distribution. Mode collapse is a phenomenon that is known to happen in GANs [1]. The single mode that was generated by the model corresponded with the initializing weights of the neural field, resulting in stagnating cross-entropy. Additionally, the initialization neural field happened to be a data sample that is very similar to a larger portion of the airplanes in the dataset. Therefore, the initialization was changed to represent a less frequent type in the dataset and additionally trained using L2-Regularization, to enforce that the neural field learns more diverse features.

Transformer

Using the newly created dataset of neural fields multiple transformer models were trained. Three different architectures of a decoder only GPT model were tested, whose implementation was also inspired by NanoGPT [5]. The model hyperparameters are shown in Table 1.

All models were trained for 1750 iterations, with an effective batch size of 32, and a learning rate of 3e-3. The models were trained using a cosine learning rate scheduler. The training and validation cross-entropy losses are shown in Figure 2.

Parameter	Small	Medium	Big
Embedding Size	256	384	512
Number of Heads	16	16	16
Number of Layers	6	7	8
Vocabulary Size	128	128	128
Context Length	3712	3712	3712

Table 1. Transformer Architectures

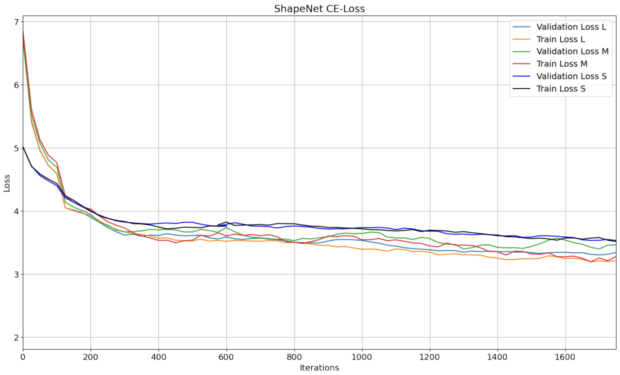

Figure 2. Cross Entropy Loss during training

5. Results

Metrics:

The ShapeNet medium model was evaluated using the Minimum Matching Distance(MMD), Coverage (COV), and 1-Nearest-Neighbor Accuracy (1-NNA) metrics, and the Frechet PointNet++ Distance as proposed by HyperDiffusion to enable qualitative comparison [2]. While for COV higher is better, indicating a full coverage of the reference space, for 1-NNA 50% is the best possible score.

$$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X, Y)$$

$$\text{COV}(S_g, S_r) = \frac{|\{\arg\min_{Y \in S_r} D(X, Y) | X \in S_g\}|}{|S_r|}$$

$$\text{1-NNA}(S_g, S_r) =$$

$$\frac{\sum_{X \in S_g} \mathbb{1}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{1}[N_Y \in S_r]}{|S_g| + |S_r|}$$

$$N_X = \underset{Y \in S_r \cup S_g}{\text{argmin}} \; D(X, Y)$$

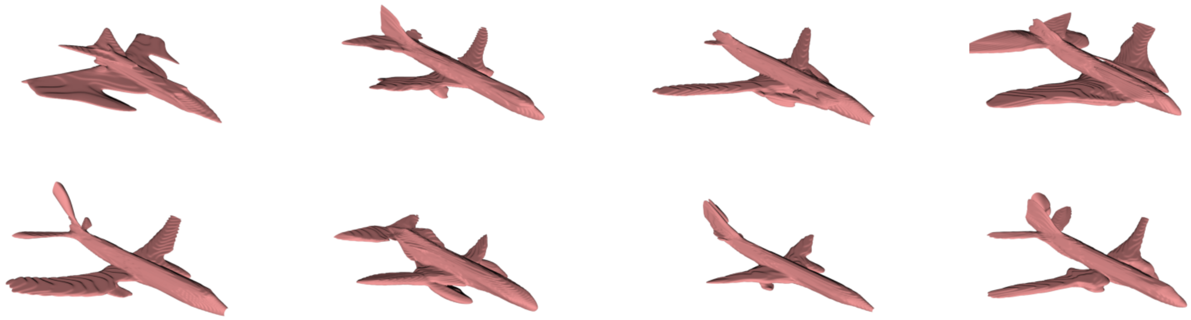

Figure 3. Autoregressive Generation of Neural Field Weights for ShapeNet

Similar to the reference work, the Chamfer Distance (CD) for 3D point clouds was used as a distance measure $D(X, Y)$ and also multiplied by a constant 10^2 to enable a comparison with the metrics present by the HyperDiffusion paper.

Additionally, a perceptual metric was also adapted, the Frechet PointNet++ Distance (FPD) [8], which is an extension to the domain of Point Clouds to the Frechet Inception Distance (FID) metric [4] typically used to evaluate fidelity. For the FPD lower values are better.

Model Evaluation

In Figure 3 novel generated airplanes are shown. For this the SOS-token is fed into the transformer and the tokens are generated in an autoregressive fashion. The tokens are then back transformed to MLP-weights using the learned codebooks, the reconstructed NeF is used to calculate a point cloud containing SDF-values and the marching cube algorithm is deployed to transform the point cloud to a valid mesh. A comparison regarding the performance of the model against different diffusion models was conducted as there is no direct comparison model available for other autoregressive techniques in this context. The results are shown in Table 2.

Method	MMD ↓	COV % ↑	1-NNA % ↓	FPD ↓
Voxel Baseline	6.0	28	94.1	38.9
PVD	3.4	39	76.3	5.8
DPC	3.1	46	74.7	18.7
HyperDiffusion	3.4	49	69.3	3.5
Ours [3, 0.8]	10.0	**38.77**	90.77	**31.34**
Ours [3, 1]	**9.7**	34.57	91.31	36.78
Ours [5, 0.8]	10.5	37.50	**90.43**	35.21
Ours [5, 1]	11.1	35.64	91.36	31.87

Table 2. Results on the Shapenet dataset. Our best results are highlighted, the numbers in the brackets indicate the inference parameters used for the generation. The first number is the Top-K value and the second number is the temperature.

The results show that in principle the generation of neural field weights using this novel approach is feasible, as we outperform the diffusion based voxel baseline in Coverage, 1-NNA and output fidelity. We suspect that the performance of the method could be further improved by increasing the model size and training time, as shown in Figure 2 the training loss is still decreasing after the full training, but this investigation was not feasible due to time and compute limitations.

6. Limitations and further research

While NeF-GPT achieves, to our knowledge, the first autoregressive generation of neural field weights directly, some limitations remain. Firstly, the transformer is trained solely on the MLP-weight sequence, without any indication of how well the reconstruction resembles the original surface area. This could be improved by incorporating a secondary loss function that provides additional information in the image space. Additionally, conditioning the initial weights may hinder the learning of the neural fields by enforcing a similar structure. This issue could be addressed by developing an autoencoder structure which permits a permutation invariant input.

7. Conclusion

During this report we proposed NeF-GPT, a new autoregressive generative process for implicit neural fields. This method leverages the compact representation power of neural fields to model high-dimensional surface data by directly autoregressively generating tokens that represent the weight space of the neural fields using a decoder only transformer-based approach.

References

[1] Bhagyashree, Vandana Kushwaha, and G. C. Nandi. Study of prevention of mode collapse in generative adversarial network (gan). In *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, pages 1–6, 2020. 3

[2] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion, 2023. 1, 3

[3] Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. 1990. 2

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 4

[5] Andrej Karpathy. NanoGPT. `https://github.com/karpathy/nanoGPT`, 2022. 2, 3

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1

[7] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 1

[8] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. 4

[9] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers, 2023. 1

[10] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020. 1

[11] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 1