Adversarial Generation of Continuous Images

Ivan Skorokhodov
KAUST
Thuwal, Saudi Arabia
iskorokhodov@gmail.com

Savva Ignatyev
Skoltech
Moscow, Russia
savvaignatiev@gmail.com

Mohamed Elhoseiny
KAUST
Thuwal, Saudi Arabia
mohamed.elhoseiny@kaust.edu.sa

Abstract

*In most existing learning systems, images are typically viewed as 2D pixel arrays. However, in another paradigm gaining popularity, a 2D image is represented as an implicit neural representation (INR) — an MLP that predicts an RGB pixel value given its (x, y) coordinate. In this paper, we propose two novel architectural techniques for building INR-based image decoders: factorized multiplicative modulation and multi-scale INRs, and use them to build a state-of-the-art continuous image GAN. Previous attempts to adapt INRs for image generation were limited to MNIST-like datasets and do not scale to complex real-world data. Our proposed architectural design improves the performance of continuous image generators by $\times$6-40 times and reaches FID scores of 6.27 on LSUN bedroom 256×256 and 16.32 on FFHQ 1024×1024, greatly reducing the gap between continuous image GANs and pixel-based ones. To the best of our knowledge, these are the highest reported scores for an image generator, that consists **entirely** of fully-connected layers. Apart from that, we explore several exciting properties of INR-based decoders, like out-of-the-box superresolution, meaningful image-space interpolation, accelerated inference of low-resolution images, an ability to extrapolate outside of image boundaries and strong geometric prior. The source code is available at https://github.com/universome/inr-gan.*

1. Introduction

In deep learning, images are typically represented as 2D arrays of pixels. However, there is another paradigm which views an image as a continuous function $F(\boldsymbol{p}) = \boldsymbol{v}$ that maps pixel 2D coordinate $\boldsymbol{p} = (x, y) \in \mathbb{R}^2$ into RGB value $\boldsymbol{v} = (r, g, b) \in \mathbb{R}^3$. The advantage of such a representation is that it gives a true continuous version of the underlying 2D signal instead of its cropped quantized counterpart, like pixel-based representations do. In practice, we almost never know the underlying function $F(\boldsymbol{p})$ and thus have to work with its approximations. The most popular and expressive way to approximate $F(\boldsymbol{p})$ is through a neural network $\mathsf{F}_{\boldsymbol{\theta}}$ [76, 71], illustrated on figure 1. It is called an *implicit neural representation (INR)* and is especially popular in 3D deep learning where working with voxels directly (i.e. pixels defined on a 3D grid) is too expensive [47, 9, 65, 53].

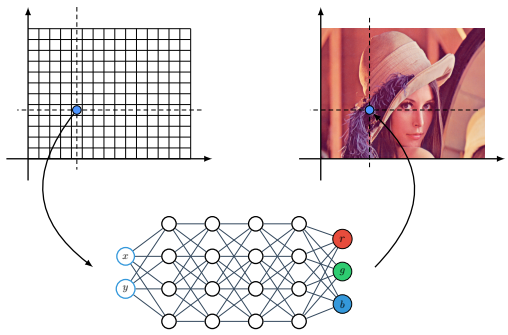


Figure 1: An illustration of an image represented in the implicit neural representation (INR) form, i.e. as an MLP that predicts an RGB pixel value given its (x, y) coordinates. In contrast to a pixel-based representation, it stores the image in its true continuous form. In our paper, we explore adversarial generation of images in such a representation.

Using INR-based decoders Building such a decoder has two severe difficulties: 1) since it is a hypernetwork, i.e. a network that produces parameters for another network [23], it is unstable to train and requires too many parameters in general [7]; and 2) it is too costly to evaluate INRs for a dense high-resolution coordinates grid limiting their application to low-resolution images only. To alleviate these issues, we design two principled architectural techniques: *factorized multiplicative modulation (FMM)* layer for hypernetworks and *multi-scale* INRs. We use these techniques to build a state-of-the-art continuous image generator that generates pictures in their INR representations. Previous attempts to build such a model were only conducted on small MNIST-like datasets [9, 2, 47] and do not scale to complex real-world data. In our case, we managed to improve their performance by $\times$6-40 times and obtain competitive FID

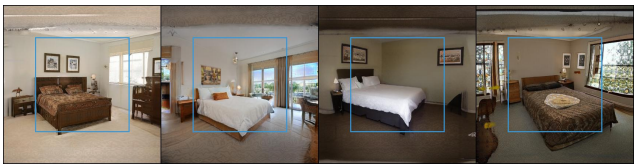


Figure 2: **Extrapolating outside of image boundaries**. After training our INR-based GAN on LSUN 256×256 we tried to evaluate it on a wider grid. During training, we used coordinates from $[0, 1]^2$ square, and here we evaluate it on coordinates from $[-0.3, 1.3]^2$ square. To our surprise, the model is capable of producing meaningful extrapolation and generating the picture beyond the coordinates area it was trained on. Blue bounding box denotes $[0, 1]^2$ coordinates area — an image area the model was trained on.

[25] scores of 6.27 and 16.32 on LSUN bedroom 256×256 and FFHQ 1024×1024 respectively, highly reducing the gap between continuous image GANs and pixel-based ones. To the best of our knowledge, these are the best reported scores for an image generator that consists *entirely* of fully-connected layers.

In our paper, we also shed light on many interesting properties of INR-based decoders:

- *Extrapolating outside of image boundaries* (see Fig. 2): an ability to generate a "zoomed-out" version of an image without being trained explicitly to do this.
- *Geometric prior* (see Fig. 4): better encoding of geometric properties of a dataset in the latent space.
- *Accelerated low-resolution inference* (see Fig. 9): an ability to generate a low-resolution version of an image in shorter time than an image of the corresponding training-time resolution.
- Meaningful image space interpolation (see Fig. 5).
- *Out-of-the-box superresolution* (see Fig. 3): an ability to produce a higher-resolution version of an image without being trained for this task at all.

We emphasize that these features come naturally to INR-based decoders and do not require any additional training.

To summarize, our contributions are the following:

1. We propose a novel factorized multiplicative modulation (FMM) layer for hypernetworks. It makes it possible to generate INRs with a large number of parameters and stabilizes hypernetwork training.
2. We propose a novel *multi-scale* INR architecture. It makes it possible to represent high-resolution images in the INR-based form in a very efficient way.

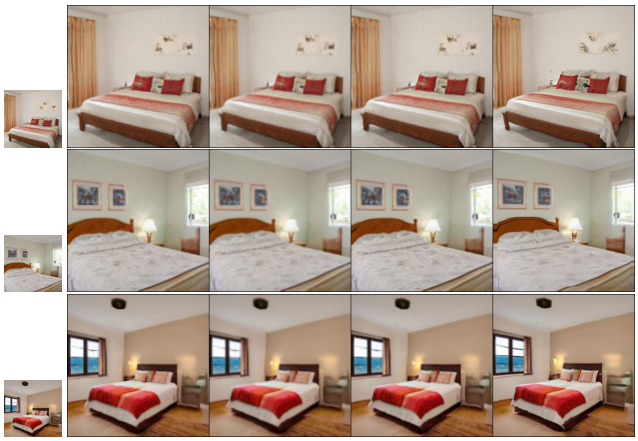
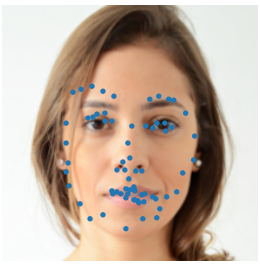


Figure 3: **Out-of-the-box superresolution properties of our INR-based decoder**. We train our INR-based GAN on LSUN 128×128 and perform upsampling at test time to 256×256 resolution with either classical interpolation techniques (first 3 columns) or with "natural" INR-based upsampling by evaluating $\mathsf{F}_{\boldsymbol{\theta}}$ on a denser coordinates grid. INR-based superresolution is much more crisp and is obtained without any additional training loss. Numbers in parentheses denote corresponding UpsampledFID scores.

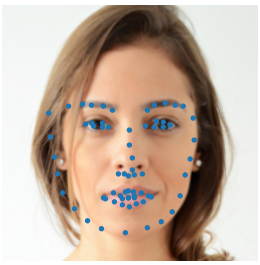
3. Using the above two techniques we build an INR-based GAN model that outperforms the existing continuous image generators by $\times$6-40 times on large real-world datasets.
4. We explore several exciting properties of INR-based decoders, like out-of-the-box superresolution, meaningful image-space interpolation, accelerated inference of low-resolution images, an ability to extrapolate outside of image boundaries and geometric prior.

2. Related work

Implicit Neural Representations. The original idea of augmenting neural networks with coordinates information was proposed in CPPN [74] which is a neuroevolution-based model that is trained to represent a 2D image. After that, there were several works that use coordinates as an additional source of information to neural networks [48, 82, 81, 66, 73], but the largest popularity of implicit neural representations (INRs) is observed in 3D deep learning, where it provides a cheap and continuous way to represent a 3D shape compared to mesh/voxel/pointcloud-based ones [53, 65, 47, 18, 17]. They have also been used for other tasks, like representing textures [61], 3D shapes flow [58], scenes [55, 72], audios and differential equations [71], hu-



(a) StyleGAN2 generator



(b) Our INR-based generator

Figure 4: **Predicting keypoints from latent codes.** We train a linear model to predict face keypoints directly from the corresponding latent codes. Its performance shows how much of geometric information is contained in a latent code and how accessible it is. On this metric, our model easily outperforms StyleGAN2 despite the fact that it was trained with the additional PPL loss [36] which forces better latent codes conditioning for the decoder. That demonstrates better *geometric prior* of our model. The corresponding scores are presented in Table 2.

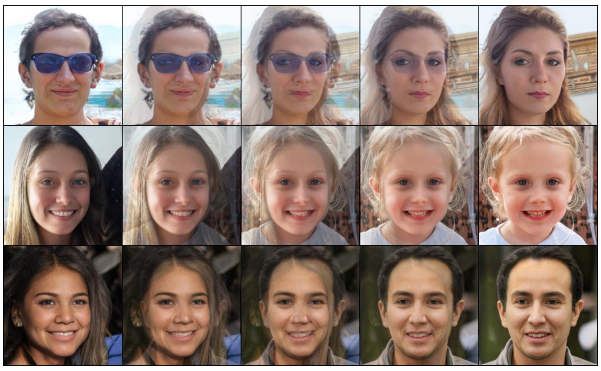
man grasps [37] and other information [54, 11]. Occupancy Networks [53, 67, 11, 33] model a probability function of a voxel being occupied by a 3D shape and typically employ a coordinate-based decoder which operates on top of single-view images. They use multi-resolution surface extraction method, which is similar in nature to our proposed multi-scale INR. However, in our case we share computation between neighbouring pixels while they use surface extraction to find regions to refine the predictions on. In this way, they conduct the full inference for each low-resolution coordinate which is the opposite of what we try to achieve with multi-scale INRs. DeepSDF [65] models a signed distance function instead of the occupancy function and they additionally have an encoder, which transforms an image into a latent code. IM-NET [9] proposes to train a generative model on top of these latent codes and conduct experiments not only on ShapeNet objects [6], but on MNIST images as well. DeepMeta [47] models the occupancy function and they predict decoder parameters instead of the latent codes. A particularly important branch of research on INRs is concerned about the most efficient way to encode coordinates positions [79, 16]. Recent works show that using Fourier features greatly improves INR expressiveness [76, 71], which we observe in our experiments as well.

GANs. State-of-the-art results in image generation are held by generative adversarial networks (GANs) [20]. Two key challenges in GAN training are instability and mode collapse, that's why a big part of research in the recent years was devoted to finding stronger objective formulation [1, 51, 44] and regularizers [22, 52, 56] that would encourage stability and diversity.

Generative models + coordinates. There were attempts to combine generative modelling and INR-based representations prior to our work. IM-NET [9] trains a generative model on top of latent codes of an occupancy autoencoder. In our case, instead of feeding a latent code to a coordinate-based decoder, we produce its parameters with a hypernetwork-based generator. Besides, their image generation experiments were limited to small-scale MNIST-like datasets [42] only. CoordConv GAN [48] concatenates coordinates to each representation in DCGAN model which endows it with geometric translating behavior during latent space interpolation. [86] use CoordConv layers to perform superresolution. SBD [83] augments a VAE [39] decoder with coordinates information. SpatialVAE [2] additionally models rotation and translation separately from the latent codes COCO-GAN [45] generates images by patches given their spatial information and then assembles them into a single image. [88] uses coordinates-based convolutions for sky replacement and video harmonization.

Hypernetworks. Hypernetworks or meta-models are models that generate parameters for other models [23, 46]. Such parametrization provides higher expressivity [14, 15] and compression due to weight sharing through a meta-model [23]. Our factorized multiplicative modulation (FMM) is closely related to squeeze-and-excitation mechanism [27]. But in contrast to it, we modulate weights instead of hidden representations, similar to [8], where authors condition kernel weights on an input via attention. Hypernetworks are known to be unstable to train [78] and [7] proposed a principled initialization scheme to remedy the issue. In our case, we found it to be unnecessary since our FMM module successfully reduces the influence of hypernetwork initialization on signal propagation inside an INR. Hypernetworks found a lot of applications in other areas, like few-shot learning [3], continual learning [80], architecture search [87] and others. In case of generative modelling, [75] built a character-level language model and [68] proposed a generative hypernetwork to produce parameters of neural classifiers. [41] encodes an image dataset into a hypernetwork by training it to produce INR parameters for image reconstruction. They show that one can obtain decent superresolution performance at test time and we confirm this finding in our experiments as well.

Hypernetworks + generative modeling. Combining hypernetworks and generative models is not new. In [68] and [24], authors built a GAN model to generate parameters of a neural network that solves regression or classification task and demonstrate its favourable performance for uncertainty estimation. HyperVAE [57] is designated to encode any target distribution by producing generative model parameters given distribution samples. HCNAF [62] is a hypernetwork that produces parameters for a conditional autoregressive flow model [38, 64, 28].



(a) Image interpolation in pixel-based form.

(b) Image interpolation in INR-based form.

Figure 5: Images have meaningful interpolation when represented in the INR-based form. To interpolate between $\mathsf{F}_{\boldsymbol{\theta}_1}$ and $\mathsf{F}_{\boldsymbol{\theta}_2}$, we compute interpolation parameters $\boldsymbol{\theta} = \alpha\boldsymbol{\theta}_1 + (1-\alpha)\boldsymbol{\theta}_2$ and then evaluate $\mathsf{F}_{\boldsymbol{\theta}}$ for the provided coordinates grid.

Hypernetworks + INRs. There are also works that combine hypernetworks and INRs. [41] proposes to represent image dataset using a hypernetwork and perform super-resolution by passing denser coordinate grid into it. DeepMeta [47] builds an encoder that takes a single-view 3D shape image as input and outputs parameters of an INR. Authors also trained a model to encode a MNIST image into a temporal sequence of digits.

Computationally efficient models. Among other things, we demonstrate that our INR-based decoder enjoys faster inference speed compared to classical convolutional ones. Each INR layer can be seen as a 1×1 convolution [26, 70], i.e. a convolution with the kernel size of 1. The core difference is that layer's weights are produced with a hypernetwork G. Since INR may have a lot of parameters, we factorize them with low-rank factorization [85]. There is a vast literature of using low-rank matrix approximations to compress deep models or accelerate them [59, 10, 31, 12]. In our case, we found it sufficient just to output a weight matrix as a product of two low-rank matrices $\boldsymbol{W} = \boldsymbol{A} \times \boldsymbol{B}$ without using any specialized techniques. SENet [27] proposed to apply squeeze-and-excitation mechanism on the hidden representations, and we apply them on the INR weights to make the model be more stable to train and faster to converge. This is similar in nature to [8], but our modulation weights are dependent on latent code $\boldsymbol{z}$ instead of the input.

3. Image Meta Generation

3.1. Model overview

We adopt a traditional GAN training setup and replace convolution-based generator with our INR-based generator $\mathsf{G}(\boldsymbol{z})$: it becomes a hypernetwork that takes latent code $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ as input and generates parameters $\boldsymbol{\theta}$ for an INR model $\mathsf{F}_{\boldsymbol{\theta}}$. Then, we evaluate $\mathsf{F}_{\boldsymbol{\theta}}$ at all the locations of a predefined coordinates grid, which size is determined by the dataset resolution. For example, for LSUN bedroom 256×256 we compute pixel values at $256^2 = 65536$ grid locations of $[0, 1]^2$ square, that are positioned uniformly. The described process is illustrated on figure Fig. 6. We use recently proposed Fourier features to embed pixel coordinates [76, 71]. From the implementation perspective, these features is just a simple linear layer that maps raw (x, y) coordinates into a d-dimensional vector with the increased by 10-20 times initialization scale and sine or cosine functions as non-linearities. However, in contrast to [76], our Fourier embedding matrix is not kept fixed, but predicted by a generator model from a given latent code $\boldsymbol{z}$. This gives the model flexibility to select feature frequencies that are the most appropriate for a given image. Apart from a different image generation process, our training is identical to traditional GAN training.

3.2. Factorized Multiplicative Modulation

Imagine, that we need to produce parameters $\boldsymbol{W}^\ell \in \mathbb{R}^{n_{\text{in}} \times n_{\text{out}}}, \boldsymbol{b}^\ell \in \mathbb{R}^{n_{\text{out}}}$ of the ℓ-th dense layer of $\mathsf{F}_{\boldsymbol{\theta}}$. A naive hypernetwork implementation would produce them directly and if its penultimate layer dimension equals to h, that would result in $h \times (n_{\text{in}} \cdot n_{\text{out}} + n_{\text{out}})$ parameters for the its final output matrix. Even for small $n_{\text{in}}, n_{\text{out}}$ this is prohibitively expensive. The main problem lies in generating $\boldsymbol{W}^\ell$, thus factorizing it via low-rank matrix decomposition $\boldsymbol{W}^\ell = \boldsymbol{A}^\ell \times \boldsymbol{B}^\ell$ may seem like a reasonable idea. But our preliminary experiments showed that it severely decreases the performance. The reason is that having low-rank weight matrices is equivalent to having a lot of zero singular values which produces instabilities in GAN training [4, 56]. That's why we change the parametrization in such a way that the full rank is preserved while the hypernetwork output is factorized. Namely, we first define a shared parameters matrix $\boldsymbol{W}_s^\ell$ that is multiplied by a low-rank matrix $\boldsymbol{W}_h^\ell = \boldsymbol{A}^\ell \times \boldsymbol{B}^\ell$, produced by a hypernetwork. We compute the final weight

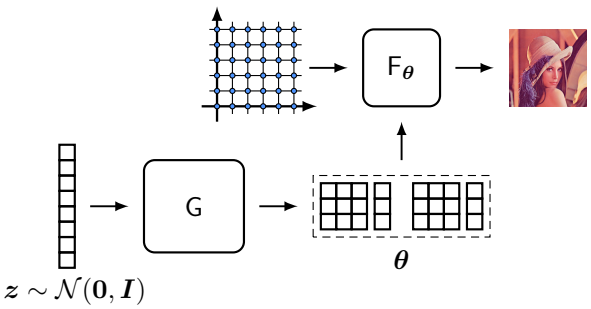


Figure 6: Image generation process with our generator. First, $\mathsf{G}(z)$ produces parameters $\boldsymbol{\theta}$ which are then plugged into INR model $\mathsf{F}_{\boldsymbol{\theta}}$. After that, 2D pixel coordinates (x, y) are passed as input to $\mathsf{F}_{\boldsymbol{\theta}}$ and it predicts the corresponding RGB values. To generate the whole image, we compute pixel values at each (x, y) location of $H \times W$ grid. Using batch matrix-matrix multiplications, this process generates a batch of images faster than traditional convolutional decoders do, since it is does not spend computation on gathering context information from neighbouring pixels. Our INR-based GAN differs from classical GAN training only in this new generator architecture.

matrix $\boldsymbol{W}^{\ell}$ as $\boldsymbol{W}^{\ell} = \boldsymbol{W}_s^{\ell} \odot \sigma(\boldsymbol{W}_h^{\ell})$ where σ is sigmoid function. Bias vector $\boldsymbol{b}^{\ell}$ is produced directly since it is small. In all the experiments, rank of 10 is used for all $\mathsf{F}_{\boldsymbol{\theta}}$ layers except the first one which takes 2D coordinates and the last one which outputs an RGB value: the corresponding matrices are small enough to produce them directly.

3.3. Multi-scale INRs

Scaling a traditional INR-based decoder to large image resolutions is too expensive: to generate a 1024×1024 image we have to input $1024 \times 1024 = 1048576$ coordinates into an INR. Such a huge batch size makes it impossible to use large layer sizes since it would result in too much computation and memory consumption. To solve this problem we propose a *multi-scale INR* architecture: we split $\mathsf{F}_{\boldsymbol{\theta}}$ into K blocks, where each block operates on its own resolution. Earlier blocks compute low-resolution features that are then replicated and passed to the next level. This process is illustrated on Fig. 7 and is equivalent to quantizing the high-resolution coordinates grid for earlier layers. For the multi-scale INRs, we use more neurons for lower resolutions and fewer ones for the more expensive high-resolution blocks. This is similar in nature to how convolutional decoders are built [60] and allows us to both save the computation via sharing and ground the pixels on a common context.

We start with the resolution of 64×64 for the first block and increase it by 2 for each next block until we reach the target resolution. Each block contains 2-4 layers and Fourier coordinates features are concatenated to a hidden

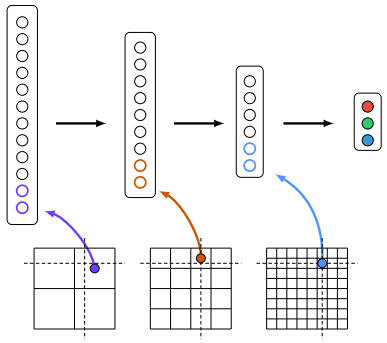


Figure 7: **Multi-Scale INR**. We split $\mathsf{F}_{\boldsymbol{\theta}}$ into K blocks of increasing resolution, where each block operates on its own scale. This allows to share computation between neighbouring pixels *and* condition them on a common context.

representation in the beginning of each block. Replicating low-resolution features for each high-dimensional pixel is equivalent to upsampling the inner grid with nearest neighbours interpolation and this is the way we implement it in practice. Further implementation details can be found in the supplementary material or the attached source code.

4. Experiments

4.1. Standard GAN training

Datasets. We conduct experiments on 3 datasets: LSUN bedroom 128×128, LSUN bedroom 256×256 and FFHQ 1024×1024. LSUN Bedroom consists on 3M images of in-the-wild bedrooms [84] and FFHQ is a high-resolution dataset of 70k human faces [35]. For all the datasets, we apply random horizontal flip for data augmentation.

Evaluation metrics. We evaluate the model using Frechet Inception Distance (FID) [25] metric using 50k images to compute the statistics. We also compute amount of parameters for a given model and amount of multiply-accumulate operations (MACs) — a standard measure for accessing model's computational efficiency [26, 13].

Models and training details. All our models have equivalent training settings and differ only in the generator architecture. We use two existing architectures as our baseline:

1. A non-hypernetwork-based generator, which has shared parameters for all INRs and each INR is conditioned on latent code $\boldsymbol{w} = \mathsf{G}(\boldsymbol{z})$ [9, 2, 65]. I.e., instead of producing parameters for $\mathsf{F}_{\boldsymbol{\theta}}$, we pass $\boldsymbol{w}$ into it as an additional input $\boldsymbol{v} = \mathsf{F}_{\boldsymbol{\theta}}(x, y, \boldsymbol{w})$.

2. A hypernetwork-based generator, which produces parameters $\boldsymbol{\theta}$ for $\mathsf{F}_{\boldsymbol{\theta}}$ but does not use any factorization techniques to reduce the output matrix size [47].

We build upon the above baselines by incorporating Fourier positional embeddings of the coordinates [76, 71], incor-

Table 1: We start with the INR-based decoder conditioned on the latent code [9, 2] and progressively improve it. O/M denotes "out-of-memory" error: we couldn't train the model even for a batch size of 1 on a 32GB NVidia V100 GPU.

Decoder type	LSUN 128×128			LSUN 256 × 256			FFHQ 1024 × 1024		
	GMACs	#params	FID	GMACs	#params	FID	GMACs	#params	FID
Latent-code conditioned INR decoder [40, 2]	30.09	7.1M	229.9	120.33	7.1M	253.3	1925.22	7.1M	O/M
+ Hypernetwork-based decoder [47]	23.54	2055.5M	28.83	88.01	2055.5M	O/M	1377.52	2055.5M	O/M
+ Fourier embeddings from [76] (ours)	28.02	2312.02M	23.07	105.34	2312.02M	O/M	1651.52	2312.02M	O/M
+ Factorized Multiplicative Modulation (ours)	25.87	108.2M	11.51	103.19	108.2M	15.68	1649.37	108.2M	O/M
+ Multi-Scale INR (ours)	21.58	107.03M	5.69	38.76	107.03M	6.27	47.35	117.3M	16.32
StyleGAN2 generator [35]	-	-	-	84.36	30.03M	2.65	143.18	30.37M	4.41

porating our FMM layer and incorporating our multi-scale INR architecture. In all the experiments, G is a 4-layer MLP with residual connections that takes $\boldsymbol{z} \in \mathbb{R}^{512}$ as input and produces INR parameters $\boldsymbol{\theta}$ as output. For the first baseline, it produces the transformed latent codes instead. For hypernetwork-based models, we additionally apply a learned linear layer to $\boldsymbol{w}$ to obtained INR parameters $\boldsymbol{\theta}$. In all the experiments, a ResNet-based discriminator from StyleGAN2 [36] is used. However, since beating the scores is not the goal of the paper, we used its "small" version (corresponding to config-e). R_1-regularization [52] with the penalty weight of 10 is used. For G, the learning rate equals to 0.0005 and for D it equals to 0.002 since StyleGAN2 discriminator uses equalized learning rate parametrization that requires higher learning rates. To keep our setup simple, we do not employ any training tricks for G training, like progressive growing, latent mixing, equalized learning rates, pixel normalization, noise injection, etc that the recent literature showed to be crucial for obtaining SotA results [34, 35, 36]. All the models are trained for 150k iterations on 4 NVidia V100 32GB GPUs.

Results. The results are reported in Table 1. As we can see, the latent code conditioned baseline cannot fit training data at all since it lacks expressivity and cannot capture the whole image representation in the latent code. Its hypernetworks-based counterpart achieves much higher performance but is still not competitive. An attempt to fit the baselines on FFHQ 1024^2 resulted in out-of-memory errors even for a batch size of 1 for a 32GB GPU. The non-factorized hypernetwork-based decoder [47] was too expensive even for 256^2 resolution since its hypernetwork output matrix is too large.

Our INR-based decoder with FMM layer and multi-scale INR architecture achieves competitive FID scores and greatly reduces the gap between continuous and pixel-based image generation. It is still inferior to StyleGAN2 [36] in terms of the performance, but is 3 times more efficient. One should also note that we didn't use any of the numerous training tricks employed by StyleGAN2 and that our discriminator architecture is smaller (it corresponds to config-e [36]). Besides, as we show in Section 4.2, our model naturally gives rise to a lot of additional properties that convolutional decoders lack. Our model also has 3 times more parameters than its convolution-based counterpart. The reason of it is in the huge size of the output projection matrix of G which has the dimensionality of $\approx 10^3 \times 10^5$ occupying 90-99% of parameters of the entire model. Compressing this matrix is an ongoing hypernetworks research topic [23, 12] and we leave this for future work. We emphasize that despite it enormous size, the inference time for this output layer is almost negligible due to highly optimized matrix-matrix multiplications on modern GPUs.

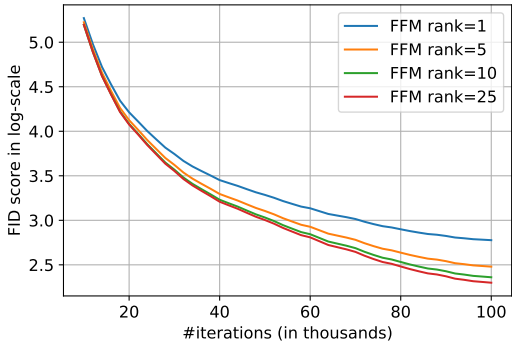


Figure 8: FID scores (in log scale) for different rank values of the proposed FMM modulation. Increasing the rank beyond 1 gives clear advantage, but the returns are diminishing for the subsequent rank increase.

Ablating FFM rank. In all the experiments, we use FMM rank of 10. We ablate over its importance for LSUN bedroom 256×256 and report the resulting convergence plots on Fig. 8. As one can see, the model benefits from the increased rank, but the advantage is diminishing for further rank increase. These results show that vanilla or "full-rank" hypernetworks are heavily overparameterized and there is no need for the predicting each parameter separately. On the other that is the evidence that there is a lot of potential for "going further" than squeeze-and-excitation [27] and AdaIN [30, 35] approaches, i.e. predicting more than a single modulating value for the each neuron.

Table 2: Keypoints Prediction Loss (KPL) metric for our INR-based decoder and StyleGAN2 generator computed on FFHQ 256×256 dataset for different latent spaces. The qualitative difference is illustrated on Fig. 4.

Generator	KPL ($\mathcal{Z}$)	KPL ($\mathcal{W}$)	KPL (random)
StyleGAN2	$2.1 \cdot 10^{-4}$	$7.6 \cdot 10^{-5}$	$4.6 \cdot 10^{-4}$
Ours	$6.0 \cdot 10^{-5}$	$5.2 \cdot 10^{-5}$	$4.7 \cdot 10^{-4}$

4.2. Exploring the properties

Extrapolating outside of image boundaries. At test time, we sample pixels beyond the coordinates grid that the model was trained on and present the results on Fig. 2. It shows that an INR-based decoder is capable of generating meaningful content outside of the grid boundaries, which is equivalent to zooming-out operation. It is a surprising quality indicating that the coordinates features produced by the generator are exploited by an INR in a generalizable manner instead of just being memorized.

Meaningful interpolation. Image space interpolation is known for its poor behaviour [19]. However, when images are represented in the INR-based form and not the pixel-based one, the interpolation becomes reasonable. We illustrate the difference on Fig. 5.

Keypoints prediction. Direct access to coordinates provides more fine-grained control over geometrical properties of images during the generation process. Thus one can expect the INR-based decoder to better embed the geometric structure of an image into the latent space [48]. To test this hypothesis we perform the following experiment. We take 10k samples from our model trained on FFHQ 256×256 and extract face keypoints with a pre-trained model [5]. After that we fit a simple linear regression model to predict these keypoints coordinates given the corresponding latent code. We compute the test loss of this linear regression model on the real-world images from FFHQ 256×256 and coin this metric Keypoints Prediction Loss (KPL). The corresponding latent codes for the real-world images are obtained by projecting an image into the corresponding latent space through the gradient descent optimization. We use the standard protocol from [36] to do this for the both models. KPL is computed for both $\mathcal{Z}$-space and $\mathcal{W}$-space. For our model, this corresponds to G input noise vectors z and penultimate hidden representations (i.e. hidden representations before the output projection into INR parameter space). For StyleGAN2, this corresponds to mapping network's input and output space. We also compute KPL on top of random vectors to check that a better performance in keypoints prediction is not due to their reduced variability. The results of this experiment are presented on Table 2.

Accelerated inference of lower-resolution images. Several classifiers can produce a prediction faster when an input is easy to classify [77, 29]. A generative model should have a similar property: when we ask a model to generate an image of lower resolution than the model was trained on — it should be able to do it faster. However, traditional convolutional decoders lack this property: to produce a lower-resolution image, one would need to perform the full inference and then downsample the resulted image with standard interpolation techniques. In contrast, INR-based decoders are capable of doing this naturally: for this, we should just evaluate them on a sparser grid compared to what they were trained on. To state the claim rigorously, we compute an amount of multiply-accumulate (MAC) operations for our INR-based generator and StyleGAN2 generator trained on FFHQ 1024×1024 for different lower-resolution image sizes. To produce the low-resolution image with StyleGAN2 we first produce the full-resolution image and them downsample it with nearest neighbour interpolation. For our model, we just evaluate it on a grid of the given resolution. The results are reported on Fig. 9. To the best of our knowledge, our work is the first one that explores the accelerated generation of lower-resolution images — an analog of early-exit strategies [77] for classifier models for image generation task.

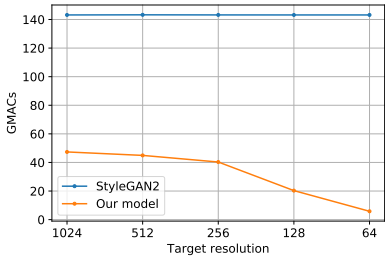


Figure 9: **Accelerated low-resolution image generation**. We measure a decoder's efficiency in terms of #MACs on generating an image of lower resolution compared to what it has been trained on. Since INR can do this by evaluating on a sparser grid, this allows it to save a lot of computation. Traditional convolutional decoders require performing a full inference first and then downsampling the produced image.

Out-of-the-box superresolution. Our INR-based decoder is able to produce images of higher resolution than it was trained on. For this, we just evaluate our model on a denser coordinates grid. To measure this quantitatively, we propose UpsampledFID: a variant of FID score where fake data statistics are computed based on upsampled low-resolution images. We compare our UpsampledFID score to three common upsampling techniques: nearest neighbour, bilinear and bicubic interpolation. Fig. 10 demonstrates that INR-upsampling outperforms them by 40-200%. Qualitative comparison is presented on Fig. 3.

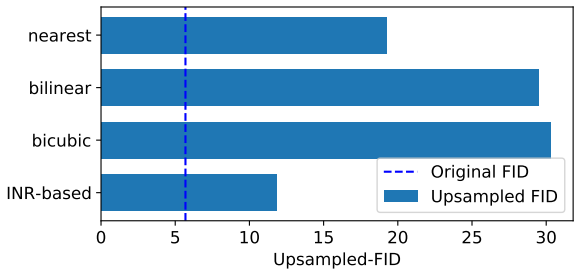


Figure 10: **Upsampled FID** scores for different upsampling techniques. We train the model on LSUN Bedroom 128×128 and then generate 2× upsampled images with the corresponding upsampling technique. After that FID score is computed for LSUN bedroom 256×256 dataset.

5. Additional potential of INR-based decoders

In Section 4.2, we explored several exciting properties of INR-based decoders and tested them in practice. Here, we highlight their additional potential and leave its exploration for future work.

An ability to backpropagate through pixel positions. Since coordinate positions (x, y) are transformed via well-differentiable operations, it gives a possibility to backpropagate through coordinate positions. This opens a large range of possible work like using spatial transformer layers [32] in the generator and not only discriminator. Or producing coordinates grid with the discriminator for zooming-in into specific parts of an image. Also, this opens the doors for differentiable rendering applications [49, 50].

Faster inference speed. Since INRs do not use local context information during inference, it does not spend computation on aggregating it like convolutions do. This is equivalent to using convolutions with a kernel size of 1 and thus works much faster [26]. Table 1 demonstrates that our model uses 3 times fewer MACs for its inference process, which is due to ignoring context information.

Parallel pixel computation Non-autoregressive models like Parallel WaveNet [63, 21] are valued by their ability to generate an arbitrarily long sequence in parallel so their inference speed decreases linearly with more compute being added. INR-based decoders also have this property due to their independent pixel generation nature. Convolutional decoders, in contrast, are forced to use local context during the inference process which limits their parallel inference.

Universal decoder architecture One can employ the same decoder architecture for training a decoder model in any domain: 2D images, 3D shapes, video, audio etc. They would only differ at what coordinates are being passed as input to the corresponding INR. For 2D images, these would be $(x, y)\mathbb{R}^2$ coordinates, for 2D video, this would be $(x, y, t)\mathbb{R}^2 \times \mathbb{R}_+$ with the additional timestep $t \in \mathbb{R}_+$ input, for 3D shapes — $(x, y, z) \in \mathbb{R}^3$ coordinates, etc. That has already been partially explored in [47, 71].

Biological plausbility. While convolutional *encoders* have a very intimate connection to how a human eye works [43], convolutional *decoders* do not have much resemblance to any human brain mechanism. In contrast, [69] argue that hypernetworks have a very close relation to how a prefrontal cortex influences other brain parts. It does so by modulating the activity in several different areas at once, which is precisely the way of how our hypernetwork-based generator applies modulation to different INR layers.

6. Limitations

The limitations of INR-based decoders come from their strengths: not using any local information during the generation process. Multi-Scale INR architecture partially alleviates this issue by grounding pixels on the shared representation, but there are only several such operations, while convolutions exploit context in each layer. We believe that this is the main reason of why INR-based decoders are inferior in terms of FID scores to convolutional counterparts. Stronger information sharing between pixels can be improved by better communication between G and $\mathsf{F}_{\boldsymbol{\theta}}$. As Table 1 demonstrates, hypernetworks is the first step towards improving the performance compared to a simple conditioning on latent codes. More expressive modulation layers are required to improve this communication between the components, which we leave for future work. We noticed that INR-based decoders may become oversensitive to high-frequency coordinates features. This may produce "texture" artifacts: a generated image having a random transparent texture spanned across it which is more noticeable for higher resolutions. We discuss this problem in more detail in the supplementary material and leave the quest of solving it for future work.

7. Conclusion

In this paper, we explored an adversarial generation of continuous images represented in implicit neural representations (INRs) form. We proposed two principled architectural techniques: factorized multiplicative modulation (FMM) and multi-scale INRs that alleviate two main problems of INR-based decoders: 1) a prohibitively large size and training instability of the underlying hypernetwork and 2) a prohibitively expensive computation cost of evaluating INRs on high-resolution images. We obtained strong state-of-the-art results for a continuous image generation task outperforming other methods by ×6-40 on modern real-world datasets. Without any bells and whistles, we obtain FID scores of 6.27 and 16.32 on LSUN bedroom 256×256 and FFHQ 1024×1024 datasets greatly reducing the gap from the pixel-based decoders. Apart from that, we explore several exciting properties of INR-based decoders, like out-

of-the-box superresolution, meaningful image-space interpolation, accelerated inference of low-resolution images, an ability to extrapolate outside of image boundaries and strong geometric prior. Finally, we discuss further potential and limitations of the INR-based paradigm.

References

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 3

[2] Tristan Bepler, Ellen Zhong, Kotaro Kelley, Edward Brignole, and Bonnie Berger. Explicitly disentangling image content from translation and rotation with spatial-vae. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15409–15419. Curran Associates, Inc., 2019. 1, 3, 5, 6

[3] Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 523–531. Curran Associates, Inc., 2016. 3

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 4

[5] Adrian Bulat and Georgios Tzimiropoulos. Super-fan: Integrated facial landmark localization and super-resolution of real-world low resolution faces in arbitrary poses with gans. *arXiv preprint arXiv:1712.02765*, 2017. 7, 13

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[7] Oscar Chang, Lampros Flokas, and Hod Lipson. Principled weight initialization for hypernetworks. In *International Conference on Learning Representations*, 2020. 1, 3

[8] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3, 4

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 3, 5, 6

[10] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P Woodruff. Algorithms for ℓ_p low rank approximation. *arXiv preprint arXiv:1705.06730*, 2017. 4

[11] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 3

[12] Elad Eban, Yair Movshovitz-Attias, Hao Wu, Mark Sandler, Andrew Poon, Yerlan Idelbayev, and Miguel A. Carreira-Perpinan. Structured multi-hashing for model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4, 6

[13] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 6202–6211, 2019. 5

[14] Tomer Galanti and Lior Wolf. Comparing the parameter complexity of hypernetworks and the embedding-based alternative. *arXiv preprint arXiv:2002.10006*, 2020. 3

[15] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[16] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252, 2017. 3

[17] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Deep structured implicit functions. *arXiv preprint arXiv:1912.06126*, 2019. 2

[18] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7154–7164, 2019. 2

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 7

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 3

[21] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017. 8

[22] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 3

[23] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 1, 3, 6

[24] Christian Henning, Johannes von Oswald, João Sacramento, Simone Carlo Surace, Jean-Pascal Pfister, and Benjamin F Grewe. Approximating the predictive distribution via adversarially-trained hypernetworks. *2018 NeurIPS Workshop on Bayesian Deep Learning*, 2018. 3

[25] 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*, pages 6626–6637, 2017. 2, 5

[26] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4, 5, 8

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3, 4, 6

[28] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087, 2018. 3

[29] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 7

[30] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 6

[31] Yerlan Idelbayev and Miguel A. Carreira-Perpinan. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[32] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 8

[33] Timothy Jeruzalski, Boyang Deng, Mohammad Norouzi, John P Lewis, Geoffrey Hinton, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. *arXiv preprint arXiv:1912.03207*, 2019. 3

[34] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 6, 12

[35] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 5, 6, 12

[36] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3, 6, 7, 12, 14

[37] Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps, 2020. 3

[38] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016. 3

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[40] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. *arXiv preprint arXiv:2002.00349*, 2020. 6

[41] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *International Conference on Artificial Neural Networks*, pages 496–510. Springer, 2019. 3, 4

[42] Y. LECUN. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998. 3

[43] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 1995. 8

[44] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 3

[45] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Cocogan: Generation by parts via conditional coordinating. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[46] Etai Littwin, Tomer Galanti, and Lior Wolf. On the optimization dynamics of wide hypernetworks. *arXiv preprint arXiv:2003.12193*, 2020. 3

[47] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 4, 5, 6, 8

[48] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9605–9616. Curran Associates, Inc., 2018. 2, 3, 7, 14

[49] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019. 8

[50] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems*, pages 8295–8306, 2019. 8

[51] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017. 3

[52] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018. 3, 6

[53] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2, 3

[54] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[55] 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. *arXiv preprint arXiv:2003.08934*, 2020. 2

[56] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 3, 4

[57] Phuoc Nguyen, Truyen Tran, Sunil Gupta, Santu Rana, Hieu-Chi Dam, and Svetha Venkatesh. Hypervae: A minimum description length variational hyper-encoding network, 2020. 3

[58] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019. 2

[59] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 442–450. Curran Associates, Inc., 2015. 4

[60] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 5

[61] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4531–4540, 2019. 2

[62] Geunseob Oh and Jean-Sebastien Valois. Hcnaf: Hyper-conditioned neural autoregressive flow and its application for probabilistic occupancy map forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[63] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018. 8

[64] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017. 3

[65] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 3, 5

[66] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[67] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[68] Neale Ratzlaff and Li Fuxin. HyperGAN: A generative model for diverse, performant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5361–5369, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 3

[69] Jacob Russin, Randall O'Reilly, and Yoshua Bengio. Deep learning needs a prefrontal cortex. *Bridging AI and Cognitive Science ICLR 2020 Workshop*, 2020. 8

[70] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for image classification. *P.h.D. thesis*, 2014. 4

[71] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 1, 2, 3, 4, 5, 8, 14

[72] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 1121–1132. Curran Associates, Inc., 2019. 2

[73] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[74] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. 2

[75] Joseph Suarez. Language modeling with recurrent highway hypernetworks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3267–3276. Curran Associates, Inc., 2017. 3

[76] 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. *NeurIPS*, 2020. 1, 3, 4, 5, 6, 14

[77] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 7

[78] Kenya Ukai, Takashi Matsubara, and Kuniaki Uehara. Hypernetwork-based implicit posterior estimation and model averaging of cnn. In *Asian Conference on Machine Learning*, pages 176–191, 2018. 3

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3, 14

[80] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. 3

[81] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic, faster and stronger. *arXiv preprint arXiv:2003.10152*, 2020. 2

[82] Zhenyi Wang and Olga Veksler. Location augmentation for cnn. *arXiv preprint arXiv:1807.07044*, 2018. 2

[83] Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019. 3

[84] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop, 2016. 5

[85] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017. 4

[86] K. Zafeirouli, A. Dimou, A. Axenopoulos, and P. Daras. Efficient, lightweight, coordinate-based network for image super resolution. In *2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–9, 2019. 3

[87] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *International Conference on Learning Representations*, 2019. 3

[88] Zhengxia Zou. Castle in the sky: Dynamic sky replacement and harmonization in videos, 2020. 3

A. Additional implementation details

As being said, we use R1 regularization for our discriminator D. However, since our D is a small version of StyleGAN2 discriminator (i.e. config-e from the paper [36]), we apply it on each iteration instead of using lazy regularization.

We use the learning rate of 0.00001 for G, 0.0005 for the shared parameters of an INR (which is a part of G) and 0.003 for D. For both models, we use Adam optimizer with the parameters $\beta_1 = 0.0, \beta_2 = 0.98$. We found it beneficial to use learning rate of 0.0005 for the shared parameters of an INR.

We also employ skip-connections for coordinates inside each multi-scale INR block. We apply them by concatenating the coordinates to inner representations.

We didn't employ any ProGAN [34], StyleGAN [35], StyleGAN2 [36] training tricks, like path regularization, progressive growing, equalized learning rate, noise injection, pixel normalization, latent mixing, etc.

For our 256×256 experiments (for both LSUN and FFHQ), we use 2 multi-scale INR blocks of resolutions 128 and 256. Each block contains 4 layers of 512 dimensions each. For FFHQ 1024×1024, we used 4 multi-scale INR blocks of resolutions 128, 256, 512 and 1024. First 2 blocks had 3 layers of dimensionality of 512, 512 resolution had 2 layers of resolution 128, the final block had 2 layers of dimensionality of 32.

Our G architecture is the same for all the experiments and consists on 3 non-linear layers with the hidden dimension of 1024 and residual connections. Our noise vector $\boldsymbol{z}$ has the dimensionality of 512.

For additional implementation details, we refer a reader to the accompanying source code.

In all the experiments, we compute FID scores based on 50k images using the tensorflow script from BigGAN pytorch repo[1].

B. Experiments details

B.1. Zooming out

On Fig. 11 we present additional examples of our zooming operation, but evaluating the model on $[-1.5, 1.5]^2$ grid instead of $[-0.3, 1.3]^2$ like we did for Fig. 2 in the main body. This is done to demonstrate the extent to which the model can extrapolate.

B.2. Keypoints prediction

As being said, we train a linear model to predict the keypoints from the latent codes. To train such a model, we first generate $n = 10^4$ latent codes $\boldsymbol{w}_1, ..., \boldsymbol{w}_n$ for each model,

[1] https://github.com/ajbrock/BigGAN-PyTorch/blob/master/inception_tf13.py

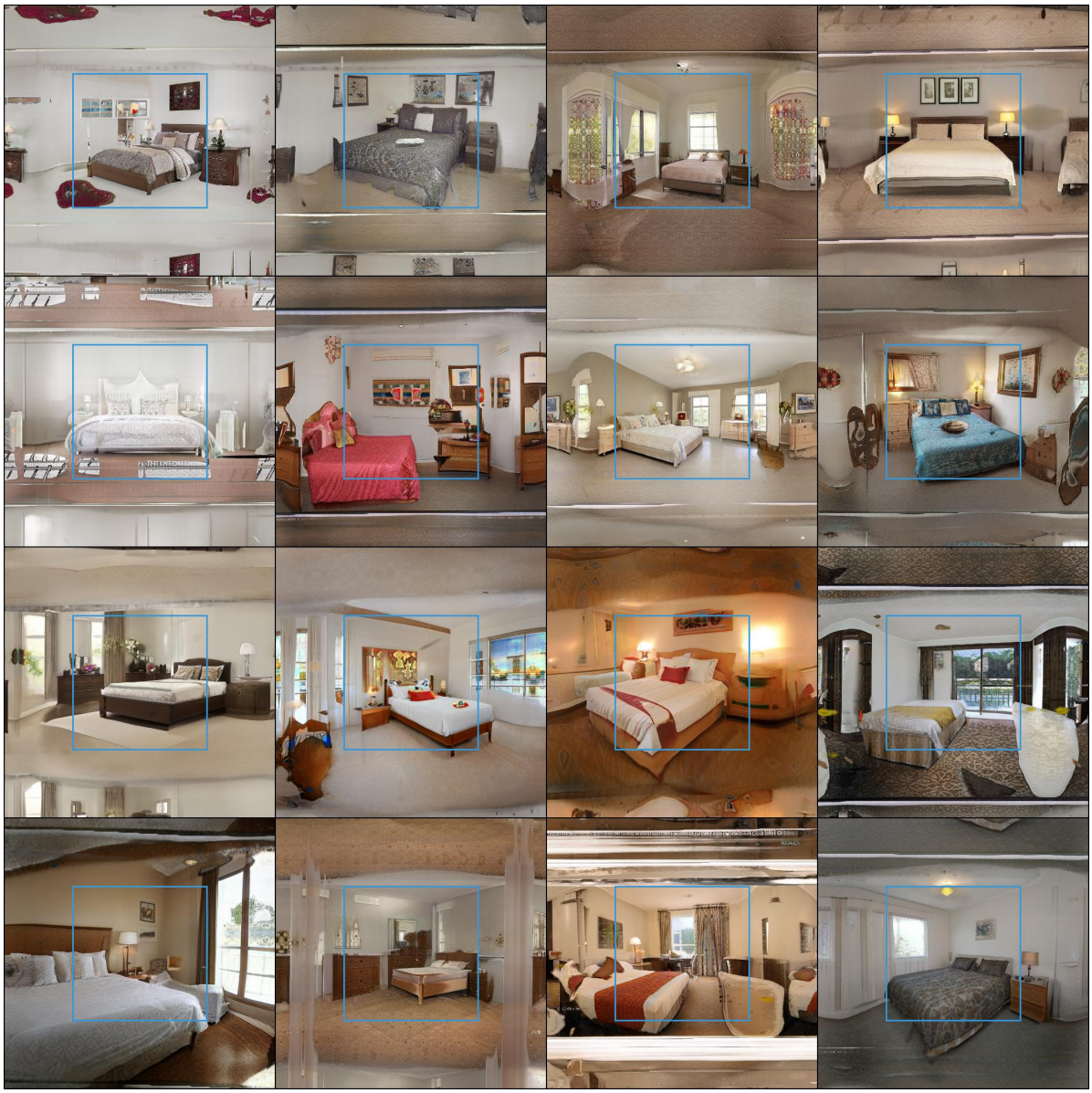


Figure 11: After training our INR-based GAN on LSUN 256×256 dataset, we feed larger coordinates grid $[-0.5, 1.5]^2$ into it. This is larger than on Figure 2 to demonstrate the extent to which the model extrapolates. As one can see, extending the grid outside of $[-0.4, 1.4]^2$ makes the quality to decrease rapidly.

then we decode them into images $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. After that, we predict keypoints vector $\boldsymbol{y}_i$ for each image with Super-FAN model [5]. Then we fit a linear regression model to predict $\boldsymbol{y}_i$ from $\boldsymbol{w}_i$.

Measuring quality based on the synthesized images may be unfair since the variability in keypoints of each model can be different: imagine a generator that always produces a face with the same keypoints. This is why we compute the test quality by embedding real FFHQ images into each model. But to additionally demonstrate that the variability is equal for the both models, we fit a linear regression model on *randomly permuted* latent codes and check its score: if the prediction accuracy is high, than the variability in keypoints is low and the prediction task is much easier. We call this metric KPL (random) and depict the corresponding values on Table 2. It clearly demonstrates that the both models have equal variability in terms of keypoints.

We project FFHQ images into a latent space using the latent space projection procedure from the official repo[2].

[2]https://github.com/NVlabs/stylegan2/blob/master/projector.py

We used default hyperparameters except for it, except that we didn't optimize the injected noise since this would take away some of the information that is better to be stored in the latent code. We depict additional qualitative results for random samples of FFHQ on Fig. 12.

B.3. Additional samples

On Fig. 13, we present additional samples with the truncation factor of 0.9 from our INR-based model trained on FFHQ1024. We perform the truncation in similar nature to StyleGAN2 [36] by linearly interpolating an inner representation inside G to its averaged value. On Fig. 14, we present common artifacts found in the produced images. On Fig. 15, we present additional superresolution samples from our model trained on LSUN 128^2. On Fig. 16, we present additional uncurated samples of our model trained on LSUN bedroom 256^2.

C. Positional encoding of coordinates

Recent works [71, 76] demonstrate that using positional embeddings [79] like Fourier features greatly increases the expressivity of a model, allowing to fit more complex data. Our positional encoding of coordinates follows [76] design and consists on a linear matrix $\boldsymbol{W} \in \mathbb{R}^{n\times 2}$ applied to raw coordinates vector $\boldsymbol{p} = (x, y)$ and followed by sine/cosine non-linearities and concatenated:

$$\boldsymbol{e}(\boldsymbol{p}) = \begin{bmatrix} \sin(\boldsymbol{W}\boldsymbol{p}) \\ \cos(\boldsymbol{W}\boldsymbol{p})^\top \end{bmatrix} \tag{1}$$

Matrix $\boldsymbol{W}$ is produced by our generator G without any factorization since it has only 2 columns. Each row of this matrix corresponds to the parameters of the Fourier transform. The norm of a row corresponds to the frequency of the corresponding wave. We depict frequencies distribution learned by our generator on Figure 17.

D. Geometric prior

Adding coordinates to network input induces powerful prior on the geometric shapes, since now different pixels, otherwise created equal are ordered through the euclidean (or any other) coordinate system. It is a well-know fact that complex geometric shapes could be compactly represented with the use of euclidean coordinates (for example one can write down an ellipse equation as $\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} = 1$). On the other hand without any form of prior one would hope to fit complex patterns with dedicated filters which would potentially consume much more parameters.

It is worthy to discuss the synergy between the coordinate representations and hypernetworks. Lets imagine a learnable system consisting of sequentially connected linear layer $W \in \mathbb{R}^{2\times 2}$, which is modulated by a hypernetwork, and INR, taking Euclidean coordinates as an input $f(X), X \in \mathbb{R}^{2\times 1}$ The whole model then could be written as $f(W(z) \times X)$. Now, it could be seen, that introducing the hypernetwork to the pipeline allows to apply linear transformation to the coordinates, rotating and zooming the image encoded by INR f. Thus, hypernetworks allow to easily perform transformations non-trivial for traditional deep learning systems.

Finer control over form and placement Recent studies show that convolutional NN struggle with such simple and crucial tasks as accurately predicting the coordinates of a drawn point [48] (and vice versa, drawing a point given the coordinates). One should expect, that such an important skill is necessary for the generative model for accurate placement of the different object parts and for precise representation of the object proportions.

E. FMM as a generalizaton of the common weight modulation schemes

In this section we show that Factorized Matrix Multiplication could be seen as a general framework for weight modulation, with Squeeze-and-Excitation, AdaIN and "vanilla" hypernetworks as its particular cases.

Squeeze-and-Excitation Let us look at the l-th FMM layer of our network with the effective rank of 1. In this case $\boldsymbol{A}^l$ and $\boldsymbol{B}^l$ are matrices (actually vectors) of the sizes $n_{in} \times 1$ and $1 \times n_{out}$ respectively. Thus, following the rules of matrix multiplication, we get that $\boldsymbol{W}^l_{h\,i,j} = \boldsymbol{A}^l_i \cdot \boldsymbol{B}^l_j$. On the other hand, lets look at the Squeeze-and-Excitation mechanism. Here we are modulating the output of the each neuron by multiplicating it by the predicted coefficient, which is equivalent to the multiplication of the corresponding weight matrix column by this coefficient. Using our notions and denoting the pre-activation (before non-linearity) vector of modulation coefficients as $\boldsymbol{A}$ we get that in this case $\boldsymbol{W}^l_{h\,i,j} = \boldsymbol{A}^l_i$. While at the first glance it looks like our model is more expressive lets not forget about the fact that the next layer is by itself modulated with its own Squeeze-and-Excitation, from which (omitting relu non-linearity and the fact that it has its own sigmoid) we can get the $\boldsymbol{B}^l_j$ multiplier. Thus it could be seen that squeeze-and-excitation modulation is roughly equivalent to the FMM layer with the rank of 1. The same reasoning is applicable to the AdaIN case, though, with AdaIN obviously we have the additional normalization layer and do not have sigmoid non-linearity for the style vector which influences the learning dynamics in its own way. We can say that squeeze-and-excitation is the least powerful weight modulation scheme, which uses the matrix of the rank 1 to modulate the main shared weights, on the other hand it is cheap and simple.

Hypernetworks Vanilla hypernetwork is perhaps the most straightforward (and the most expensive but flexible) approach to the weight modulation. It is as simple as predicting each weight as an output of MLP. So let's demonstrate

(a) StyleGAN2 generator

(b) Our INR-based generator

Figure 12: Predicting keypoints from latent codes for random FFHQ images. The corresponding scores are presented in Table 2.

that any weight dynamic that can be modeled by hypernetwork could be fitted with the FMM of high enough rank. Let's assume that n_{in} is larger than n_{out} (which is our case, but not essential for the generality of the proof) and choose the FMM rank of n_{in}. In this case $\boldsymbol{A}^l$ and $\boldsymbol{B}^l$ are matrices of the sizes $n_{in} \times n_{in}$ and $n_{in} \times n_{out}$ respectively. Since A is a square matrix we can set it to identity constant (which is a solution easily learnt by a NN just by setting bias) and get $\boldsymbol{W}^l_{h\,i,j} = \boldsymbol{B}^l_{i,j}$. In this case any hypernetwork could be "simulated" with FMM by fitting $\boldsymbol{B}^l_{i,j} = \sigma^{-1}(\frac{\hat{\boldsymbol{W}}^l_{i,j}}{\boldsymbol{W}^l_{s\,i,j}})$, where $\hat{\boldsymbol{W}}$ denotes the weight matrix predicted by the hypernetwork. While σ^{-1} definitely imposes some restrictions, caused by the positiveness and the range, in our experiments adding sigmoid has not resulted in any harm, perhaps because of the flexible calibration of the $\boldsymbol{W}^l_s$.

We have shown that main weight modulation schemes could be seen as the boundary particular cases of our approach. Our approach to the weight modulation is somewhere in between of these two extremes, while reaping the benefits of the both of them. Studying the behaviour of this transmission is specially important for the shading light on the weight modulation at whole and going beyond straightforward approaches.

Figure 13: Random (uncurated) samples from our model trained on FFHQ 1024×1024 dataset with the truncation factor of 0.9. FID: 16.32

Figure 14: Common artifacts found in our model's samples when sampling without truncation. As one can see, the most severe ones are "stains" and patterned texture.

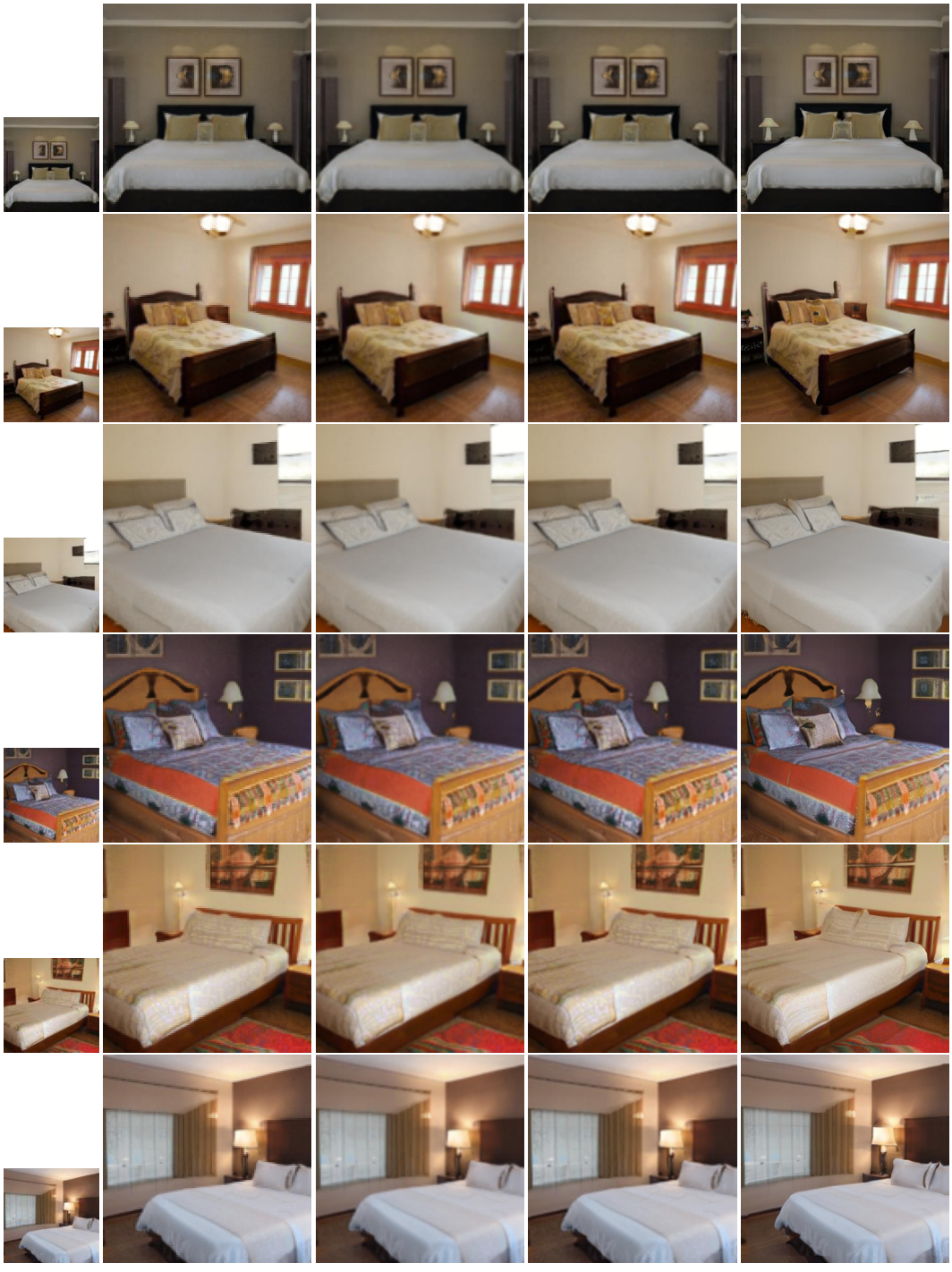


Figure 15: Additional samples from our model to show superresolution properties. We trained the model on LSUN 128×128 and upsampled to 256×256.

Figure 16: Random samples of our model on LSUN bedroom 256^2 dataset. FID: 6.27.

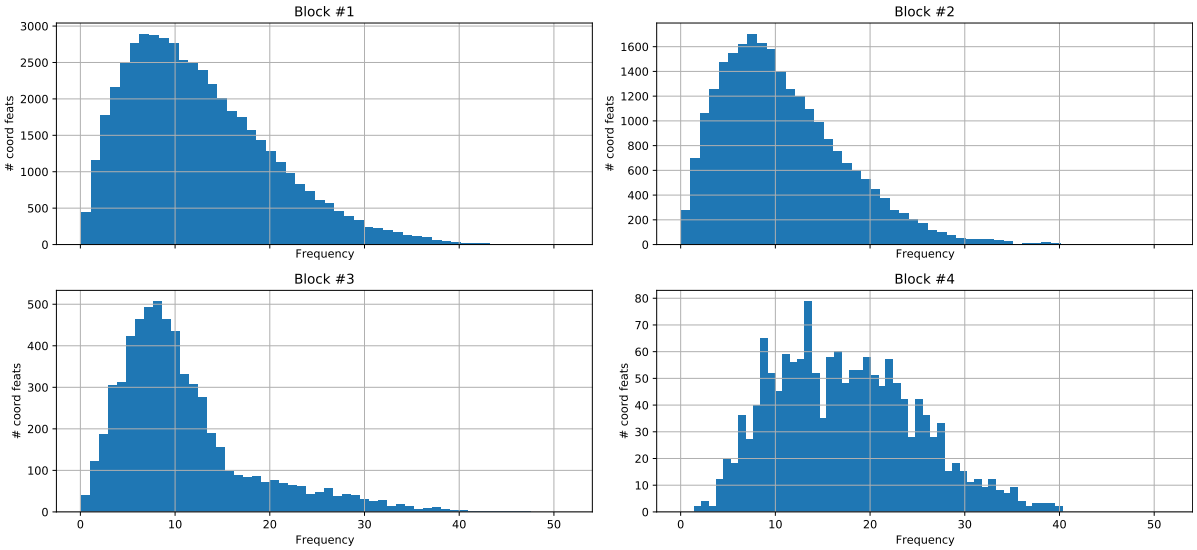


Figure 17: Frequencies distributions for different multi-scale INR blocks of our INR-based GAN trained on FFHQ 1024^2. To produce the plot, we sample 128 images in an INR-based form and computed the norms of the positional encoding layers, i.e. those layers which take raw coordinates as an input. As one can see, the model tries to use more high-frequent positional embeddings for the last layer since they are more important for drawing fine-grained details. Early layers determine the structure of an image and hence use smaller frequencies to operate on a larger scale.