BOSTON UNIVERSITY

Deep Learning for Data Science DS 542

Lecture 24
Neural Fields

Topics

Last couple weeks focused on generative models

- Generative adversarial networks
- Variational Autoencoders
- Normalizing Flows
- Diffusion Models

Remaining topics

- Neural fields (more about fitting, sometimes generative)
- Reinforcement learning

A Motivating Example

https://www.youtube.com/watch?v=JuH79E8rdKc

What is a Field?

"A field is a quantity defined for all spatial and/or temporal coordinates."

Table 1: **Examples of fields in physics and visual computing.**

Examples	Field Quantity	Scalar/Vector	Coordinates
Gravitational Field	Force per unit mass (N/kg)	Vector	$\mathbb{R}^n$
3D Paraboloid: $z = x^2 + y^2$	Height z	Scalar	$\mathbb{R}^2$
2D Circle: $r^2 = x^2 + y^2$	Radius r	Scalar	$\mathbb{R}^2$
Signed Distance Field (SDF)	Signed distance	Scalar	$\mathbb{R}^n$
Occupancy Field	Occupancy	Scalar	$\mathbb{R}^n$
Image	RGB intensity	Vector	$\mathbb{Z}^2$ pixel locations x, y
Audio	Amplitude	Scalar	$\mathbb{Z}^1$ time t

"Neural Fields in Visual Computing and Beyond" by Xie et al (2022)

Challenges Modeling Fields

- We often do not have closed form analytical solutions.
- Even if we know a general rule for calculation, we may not have all the inputs.
  - Gravity - do we know every object in the solar system / galaxy / universe?
- Often working with approximations based on sample data
  - Tradeoffs between sampling requirements (Nyquist limits) and appropriate adaptive data structures

Neural Fields

A neural field is a neural network implementing the field “interface”.

- Coordinates in.
- Field values out.

You saw these in some of the homework problems.

- (x,y) input → (R,G,B) color values out

Sometimes this is called an implicit neural representation (INR) or coordinate MLPs (MLP = multilayer perceptron, perceptron was a zero hidden layer neural network).

Application-Driven Focus

Compared to previous coverage of generative models,

- Much more focused on good approximations of the field
- Overfitting comparatively low concern (but not absent)
- More likely to make practical trade-offs vs elegant clean models

Neural Fields vs Generative Models

Exact comparisons depend on the context…

- Neural fields are sometimes trained on one data set from scratch.
  - No interest or attempt to generalize to different data sets.
- But, there are “parameterized” neural fields
  - Parameters ~ latent variables
  - Is there an explicit distribution of latents? Usually not…

Applications of Neural Fields

- Image fitting / reconstruction / modeling
- Shape representation
- Scene representation
- Scene rendering

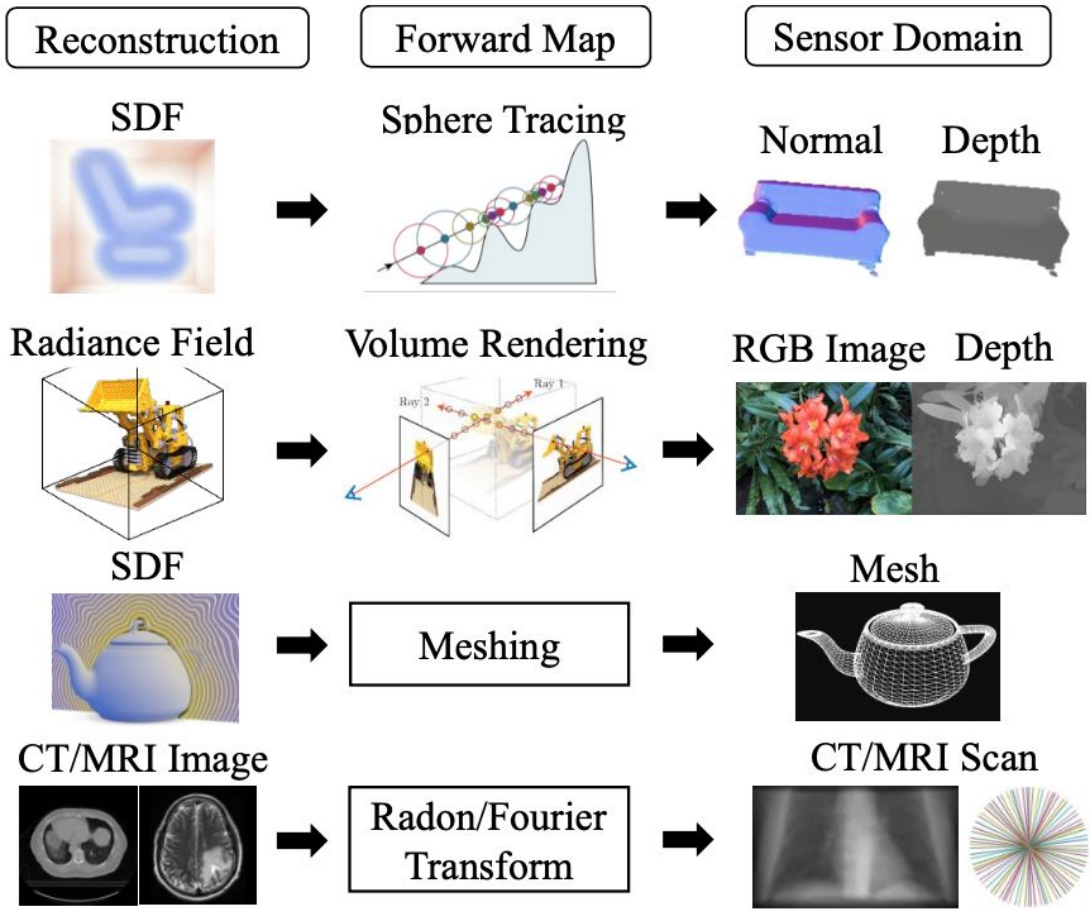
Reconstruction vs Sensor Domains

May be working with multiple fields at once.

- Reconstruction domain = what we explicitly model
- Sensor domain = how we get data

Example:

- Take several photographs and build a 3D model (NeRF)



Forward Mappings

- Train with data from sensor domain to model reconstruction domain.
- Then generate new samples from sensor domain from different “viewpoints”.

Problem	Sensor	Sensor Domain	Forward Module	Reconstruction Domain
3D Reconstruction	Digital Camera	Image (2D discrete array)	Rendering	Geometry, Appearance
Geodesy Estimation	Accelerometer	Gravitational acceleration	$F = m\vec{a} = Gm_1m_2/\vec{r}^2$	Density (mass)
CT Reconstruction	X-ray Detector	Projection domain	Radon Transform	Density/Intensity
MRI Reconstruction	RF Detector	Frequency domain	Fourier Transform	Density/Intensity
Audio Reconstruction	Microphone	Waveform	Fourier Transform	Spectrogram
Synthetic Aperture Sonar	Microphone	Waveform	Convolution (w/ PSF)	Point Scattering Distribution

“Neural Fields in Visual Computing and Beyond” by Xie et al (2022)

Neural Fields for Images

- Saw this already: (x, y) pixel coordinates → (R, G, B) color values

- Applications
  - image reconstruction (is this efficient?)
  - super resolution
  - In-painting
  - Out-painting

General Challenges of Neural Fields

- Neural network bias towards smooth functions
- Difficulty with abrupt transitions between regimes
- Is the representation good if it takes the same space as the original image?

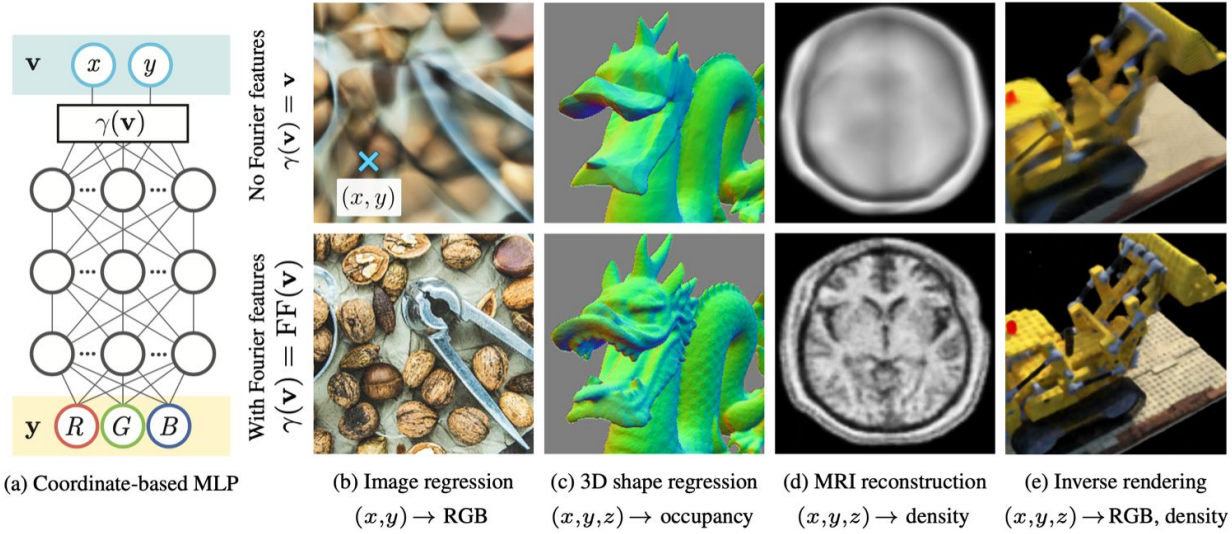
Will see these in full force when evaluating images.

Fourier Features

Re: bias towards smooth functions,

- Neural networks tend to need a very wide network and a lot of training time to learn fine-grained detail.
- Functional form leans towards “big good, small bad” and does not isolate middle values as well.

“Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains” by Tancik et al (2020)

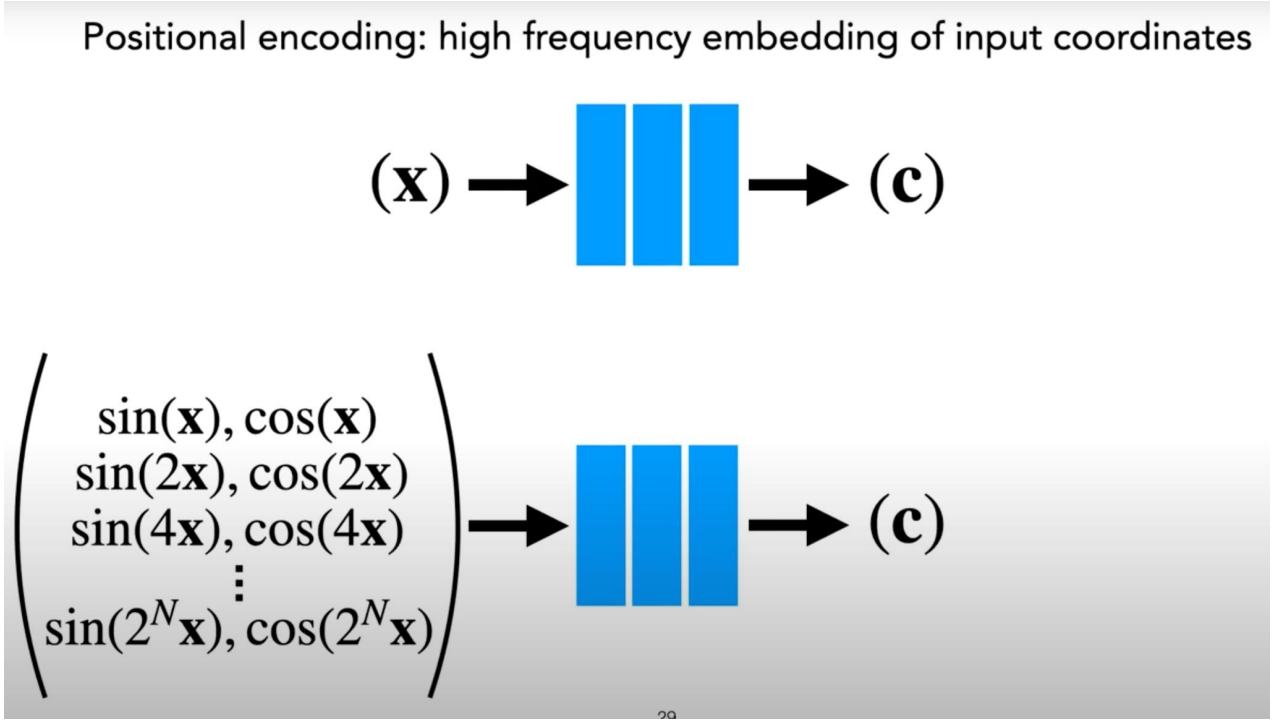


Fourier Features

Main idea:

- Add features that are high and low at different intermediate values.
- Make features with different spacings between high and low so neural network can pick different combinations for different intermediate points.
- Transformers use the same idea for positional encodings.

"Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains" by Tancik et al (2020)

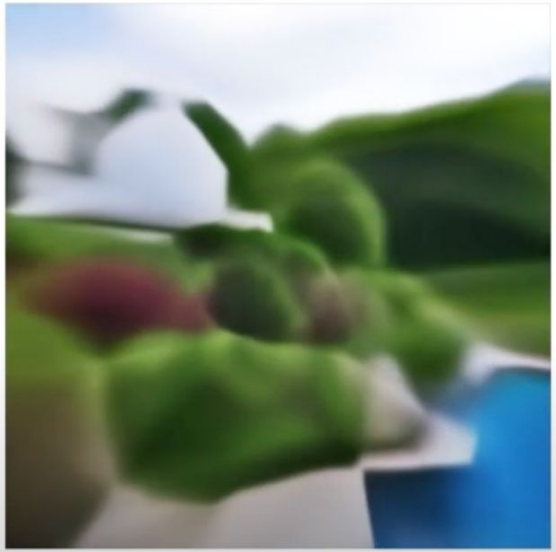


Fourier Features

Simple trick enables network to memorize images

Ground truth image

Standard fully-connected net

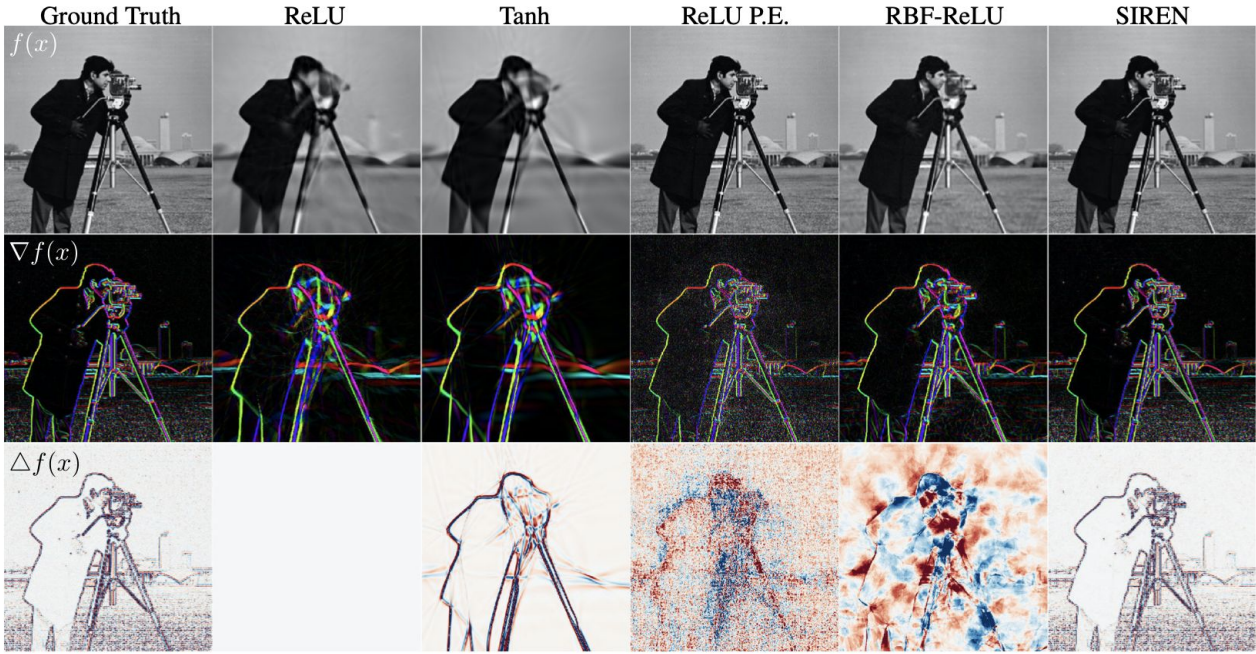


With “positional encoding”

Sinusoidal Representation Networks

Do Fourier features feel like a hack…

- What if we used sin() as the activation function?
- For the whole network?
- How is the training difficulty?



"Implicit Neural Representations with Periodic Activation Functions" by Sitzman et al (2020).

Alternative Activation Functions?

Turns out that the sine function is not the only better activation function.

- R1: “activation functions need to be parameterized where the first-order derivatives can be controlled via the hyperparameters”
- R2: “activations should consist of varying first-order derivatives across a considerable interval, and equivalently, non- negligible second-order derivatives (to obtain varying Lipschitz smoothness)”

“Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs” by Ramasinghe and Lucey (2022)

Even more Activation Functions

Most previous activation functions do not meet their requirements, but they suggest many that do…

"Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs" by Ramasinghe and Lucey (2022)

Activation (ψ)	Equation	parameterized	ψ'	ψ''	R1	R2
ReLU	$\max(0, x)$	✗	$\begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$	0	✗	✗
PReLU	$\begin{cases} x, & \text{if } x > 0 \\ ax, & \text{otherwise} \end{cases}$	✓	$\begin{cases} 1, & \text{if } x > 0 \\ a, & \text{otherwise} \end{cases}$	0	✓	✗
Sin	$\sin(ax)$	✓	$a\cos(ax)$	$-a^2\sin(ax)$	✓	✓
Tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	✗	$\frac{4e^{2x}}{\left(e^{2x}+1\right)^2}$	$-\frac{8\left(e^{2x}-1\right)e^{2x}}{\left(e^{2x}+1\right)^3}$	✗	✓
Sigmoid	$\frac{1}{1+e^{-x}}$	✗	$\frac{e^x}{(e^x+1)^2}$	$-\frac{(e^x-1)e^x}{(e^x+1)^3}$	✗	✓
SiLU	$\frac{x}{1+e^{-x}}$	✗	$\frac{e^x(e^x+x+1)}{(e^x+1)^2}$	$-\frac{e^x((x-2)e^x-x-2)}{(e^x+1)^3}$	✗	✓
SoftPlus	$\frac{1}{a}\log(1+e^{ax})$	✓	$\frac{e^{cx}}{1+e^{cx}}$	$\frac{ce^{cx}}{(e^{cx}+1)^2}$	✓	✗
Gaussian	$e^{\frac{-0.5x^2}{a^2}}$	✓	$-\frac{xe^{-\frac{x^2}{2a^2}}}{a^2}$	$\frac{\left(x^2-a^2\right)e^{-\frac{x^2}{2a^2}}}{a^4}$	✓	✓
Quadratic	$\frac{1}{1+(ax)^2}$	✓	$-\frac{2a^2x}{\left(a^2x^2+1\right)^2}$	$\frac{2a^2\left(3a^2x^2-1\right)}{\left(a^2x^2+1\right)^3}$	✓	✓
Multi Quadratic	$\frac{1}{\sqrt{1+(ax)^2}}$	✓	$-\frac{a^2x}{\left(a^2x^2+1\right)^{\frac{3}{2}}}$	$\frac{2a^4x^2-a^2}{\left(a^2x^2+1\right)^{\frac{5}{2}}}$	✓	✓
Laplacian	$e^{\left(\frac{-\lvert x\rvert}{a}\right)}$	✓	$\frac{xe^{\frac{\lvert x\rvert}{a}}}{a\lvert x\rvert}$	$\frac{e^{\frac{\lvert x\rvert}{a}}}{a^2}$	✓	✓
Super-Gaussian	$[e^{\frac{-0.5x^2}{a^2}}]^b$	✓	$-\frac{bxe^{-\frac{bx^2}{2a^2}}}{a^2}$	$\frac{b\left(bx^2-a^2\right)e^{-\frac{bx^2}{2a^2}}}{a^4}$	✓	✓
ExpSin	$e^{-\sin(ax)}$	✓	$ae^{\sin(ax)}\cos(ax)$	$-a^2e^{\sin(ax)}\left(\sin(ax)-\cos^2(ax)\right)$	✓	✓

Table 1: **Comparison of existing activation functions (top block) against the proposed activation functions (bottom block).** The proposed activations and the sine activations fulfill **R1** and **R2**, implying better suitability to encode high-frequency signals.

Even more Activation Functions

Most previous activation functions do not meet their requirements, but they suggest many that do…

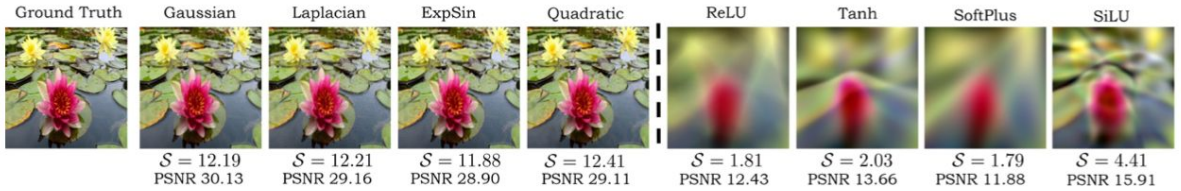


Fig. 2: **Proposed activations (left block) vs. existing activations (right block) and their respective stable ranks ($\mathcal{S}$) in image encoding without positional embeddings.** As predicted by Table 1, the proposed activations are better suited for encoding signals with high fidelity. As Sec. 3.2 stated, the stable ranks of the proposed activations are higher, indicating larger local Lipschitz constants which allow sharper edges.

"Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs" by Ramasinghe and Lucey (2022)

Gaussian Activation Functions

Comparing ReLU and Gaussian activation functions

- Both using positional encodings
- ReLU needs twice as many layers to match PSNR (peak signal to noise ratio)

Fig. 1: **ReLU vs Gaussian activations (ours).** Gaussian activations achieve better results with $\sim$ 50% less parameters. These non-periodic activations also allow embedding-free architectures (see Fig.3), and are robust to different random initializations of coordinate-MLPs than the sinusoid activations advocated in SIREN [42].

"Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs" by Ramasinghe and Lucey (2022)

Gaussian Activation Functions

Comparing ReLU and Gaussian activation functions

- ReLU loses fine details without positional encodings, but Gaussian manages to preserve them.

"Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs" by Ramasinghe and Lucey (2022)

Fig. 3: **Novel view synthesis without positional embedding (zoom in for a better view).** Gaussian activations can completely omit positional embeddings while producing results with significantly better fidelity. In contrast, the performance of ReLU-MLPs severely degrade when positional embeddings are not used. We use 8-Layer MLPs for this experiment.

Parameterized Neural Fields

So far, the neural fields discussed have been trained for one specific target.

- That is, one network per instance.
- Parameterized neural fields take in additional parameters to target a particular instance.
  - Think of the new inputs as a latent code (the parameters) plus spatial coordinates.
  - Lots of ways to integrate these into the neural field…
- SIREN paper did try out parameterized neural fields too.

SIREN Parameterized Neural Fields

SIREN paper tested their parameterized neural fields with “inpainting” tests.

- Basic idea: pick the parameters that match the context sampled.
- Note: these random samples are a lot harder than typical inpainting tests where a small contiguous area is missing.
- They tried a few different ways to map context to parameters.

“Implicit Neural Representations with Periodic Activation Functions” by Sitzman et al (2020).

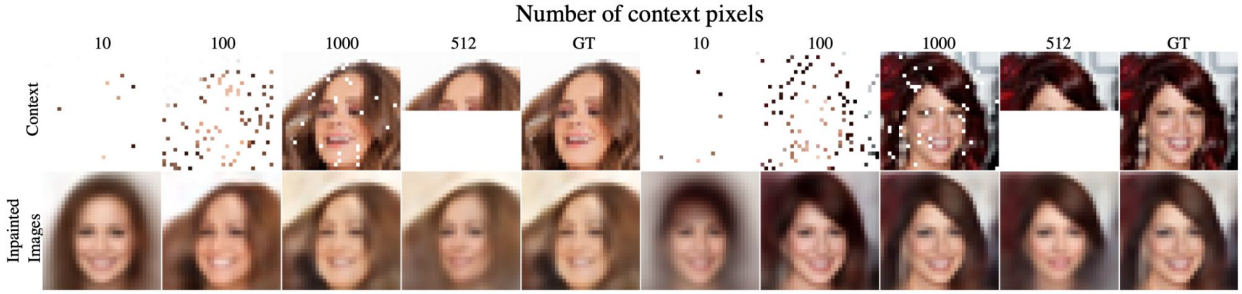


Figure 6: Generalizing across implicit functions parameterized by SIRENs on the CelebA dataset [49]. Image inpainting results are shown for various numbers of context pixels in O_j.

Aside: Hypernetworks

TLDR: a neural network that generates the parameters for some or all of another neural network.

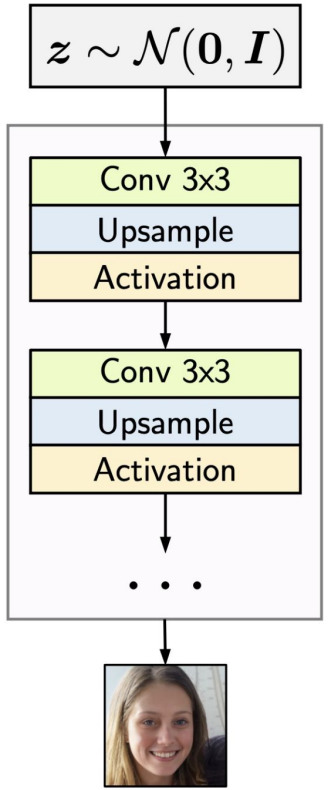
- Hypernetwork outputs tend to be large, so hypernetwork itself is pretty large?
- May still net a computation speedup if the output network is used repeatedly.
    - Say, if you are applying it to 1024 x 1024 pixels…

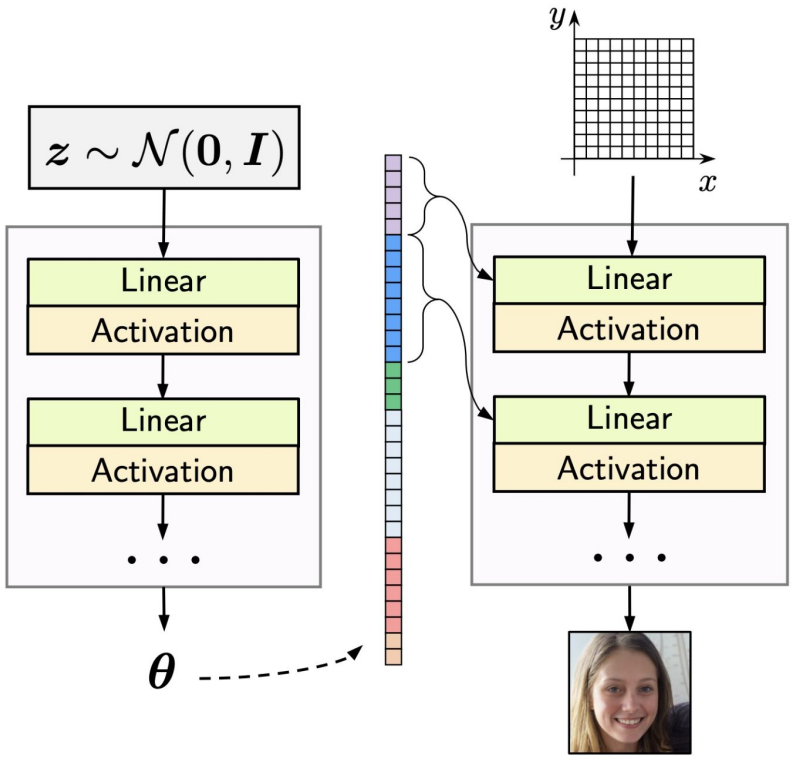
Adversarial Generation of Continuous Images

Designed a network INR-GAN combining a lot of techniques to get good image generation.

- Hypernetworks
- Multi-resolution generation
- Factorized multiplicative modulation (~low rank matrices)
- GAN training

"Adversarial Generation of Continuous Images" by Skorokhodov et al (2021)

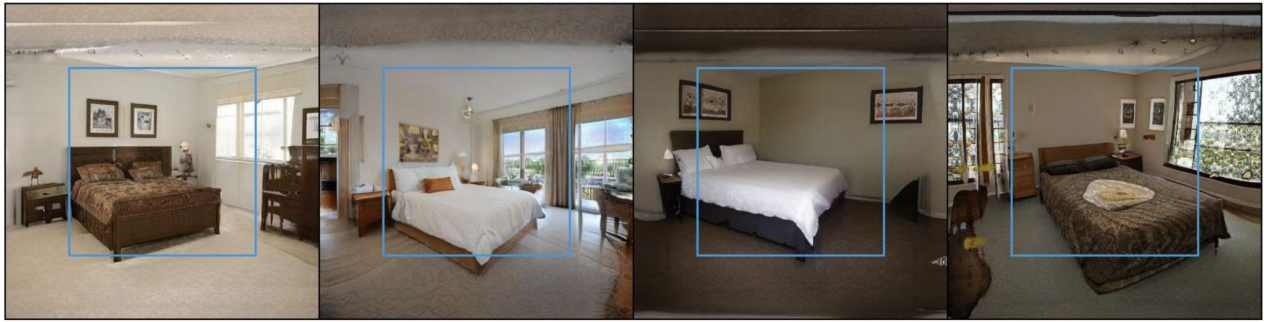




Adversarial Generation of Continuous Images

The resulting network was good at

- Out-painting

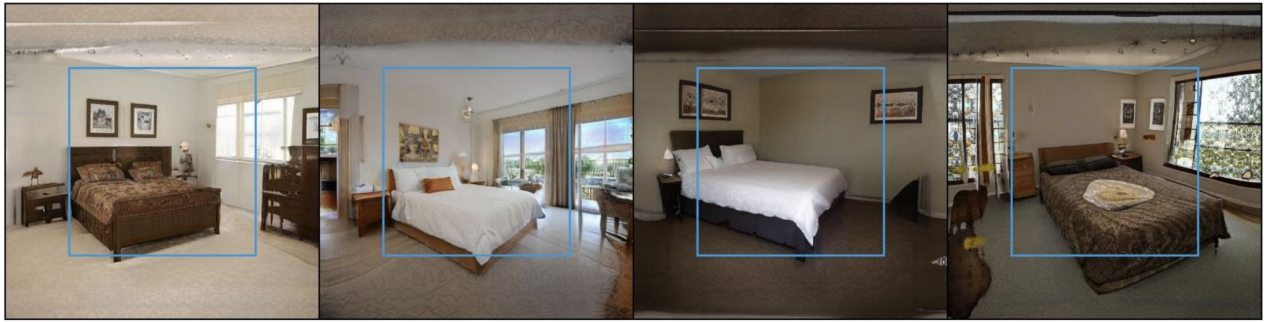


"Adversarial Generation of Continuous Images" by Skorokhodov et al (2021)

Adversarial Generation of Continuous Images

The resulting network was good at

- Out-painting



"Adversarial Generation of Continuous Images" by Skorokhodov et al (2021)

Adversarial Generation of Continuous Images

The resulting network was good at

- Image interpolation

(a) Image interpolation in the pixel-based form.

(b) Image interpolation in the INR-based form.

"Adversarial Generation of Continuous Images" by Skorokhodov et al (2021)

Adversarial Generation of Continuous Images

The resulting network was good at

- Super resolution

“INR-based decoder can perform super resolution out-of-the-box by evaluating on a denser coordinate grid.”

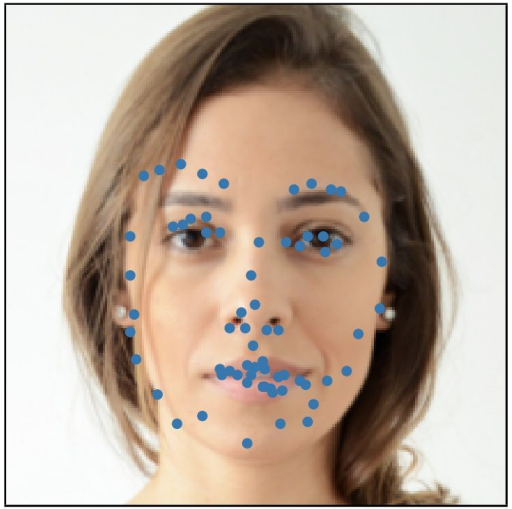
“Adversarial Generation of Continuous Images” by Skorokhodov et al (2021)

Adversarial Generation of Continuous Images

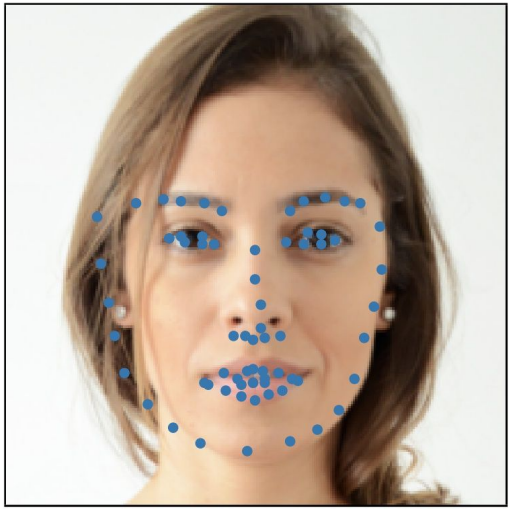
The resulting network was good at

- Key point prediction
- Just a linear model from the input parameters…

"Adversarial Generation of Continuous Images" by Skorokhodov et al (2021)



(a) StyleGAN2



(b) INR-GAN

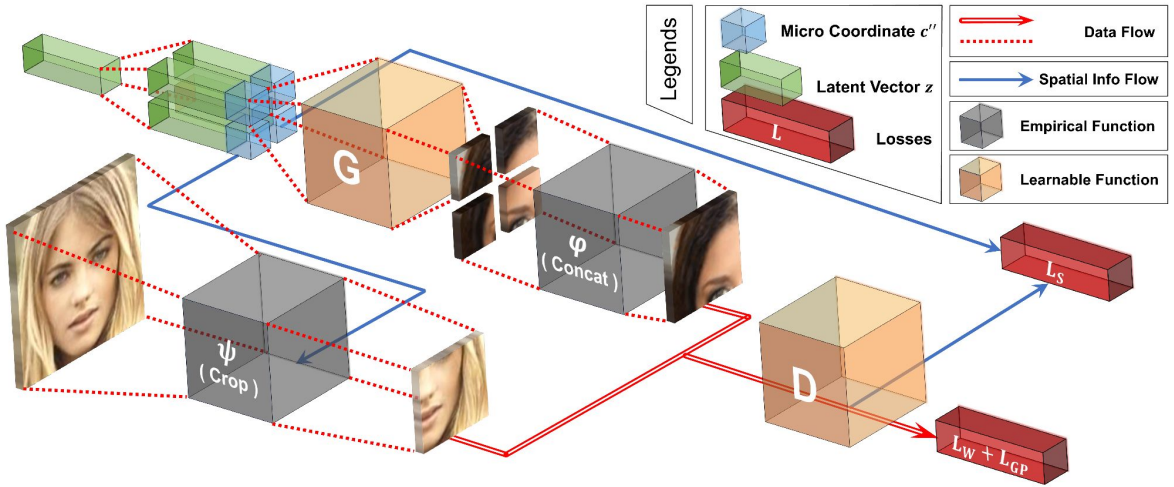
COCO-GAN

Not quite a neural field, but close.

Given a latent code,

- Break up image into “micro patches” and “macro patches”
- Neural network takes in latent code and micro patch coordinates.
- GAN training on larger macro patches.
- Can train just off macro patches instead of whole images.

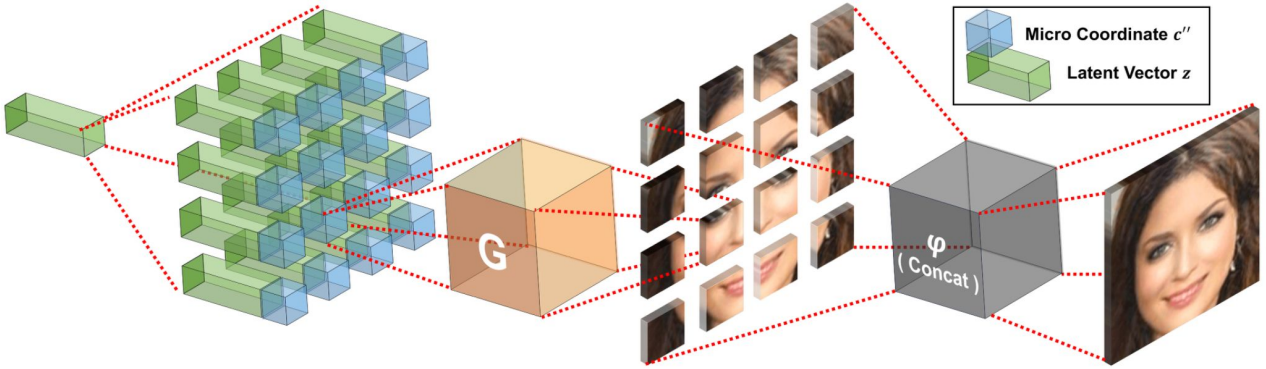
“COCO-GAN: Generation by Parts via Conditional Coordinating” by Lin et al (2019)



COCO-GAN

To generate image from latent code,

- Run network for each micro patch passing in the latent code and micro coordinates.
- Concatenate all micro patches together to get whole image.



"COCO-GAN: Generation by Parts via Conditional Coordinating" by Lin et al (2019)

COCO-GAN

COCO-GAN can also

- Interpolate between latent codes

Figure 5: The results of full-images interpolation between two latent vectors show that all micro patches are changed synchronously in response to the change of the latent vector. More interpolation results are available in Appendix G.

"COCO-GAN: Generation by Parts via Conditional Coordinating" by Lin et al (2019)

COCO-GAN

COCO-GAN can also

- Out-paint by using micro coordinates outside the normal input range.
  - Worked a bit without special training, but had some edge effects at original border.
  - Worked better after specifically training for it.

"COCO-GAN: Generation by Parts via Conditional Coordinating" by Lin et al (2019)

COCO-GAN

COCO-GAN can also

- Generate panoramas
  - Side effect of training with a cyclic coordinate system

"COCO-GAN: Generation by Parts via Conditional Coordinating" by Lin et al (2019)

Figure 8: The generated panorama is cyclic in the horizontal direction since COCO-GAN is trained with a cylindrical coordinate system. Here, we paste the same generated panorama twice (from 360° to 720°) to better illustrate the cyclic property of the generated panorama. More generation results are provided in Appendix H.

Patch-based Hybrids

Easy idea:

- Break image into many non-overlapping patches (not quite like CNN)
- Latent code per patch
  - Note: COGO-GAN looked like this, but that was micro-coordinates, not different latent codes.

Challenge:

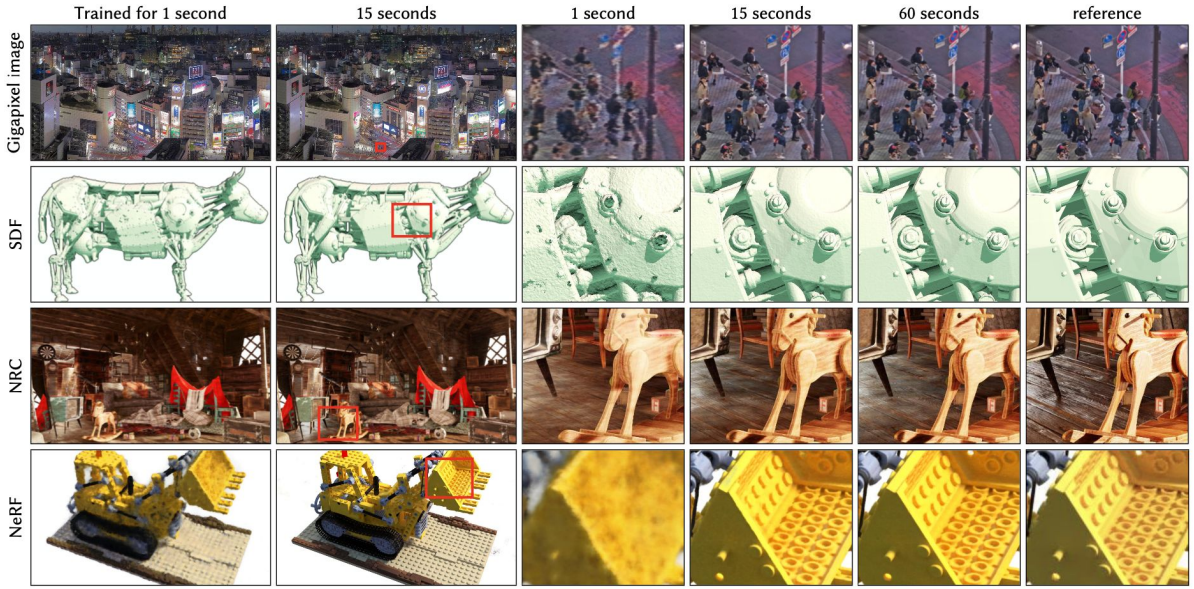
- Make the patches seamless
- How to trade off number/size of patches to balance quality vs latent size?

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Very fast, very parallelized

- Multi-resolution apprach
- Small neural network rendering many patches
- Coarse (large) patches get dedicated feature storage.
- Fine (small) patches stored in a hash table.
    - Biggest gradients win collisions.

"Instant Neural Graphics Primitives with a Multiresolution Hash Encoding" by Müller et al (2022)



Infinite Scene Generation

Unbounded extrapolation of latent vectors…

- Watch the video in the link below…

"Aligning Latent and Image Spaces to Connect the Unconnectable" by Skorokhodov et al (2021)

Shape Representation

How to represent a 3D shape with a neural field?

- Train field with positive value outside the shape and negative value inside the shape.
- Result is a signed distance function where zero represents the surface of the shape.

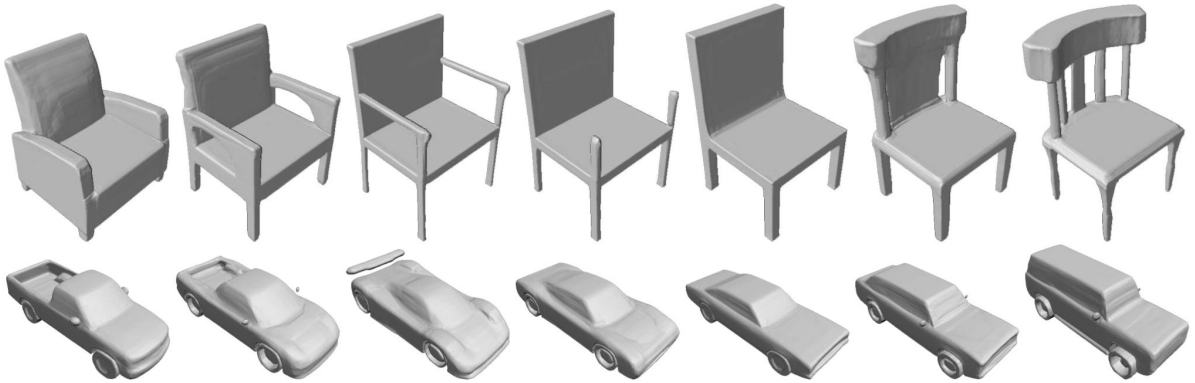


Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

"DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation" by Park et al (2019)

Shape Representation

- The surface can be constructed by querying various points to see if they are inside or outside.
- Computer graphics has standard methods to reconstruct the surface based on in/out status of a cube's vertices.

"DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation" by Park et al (2019)

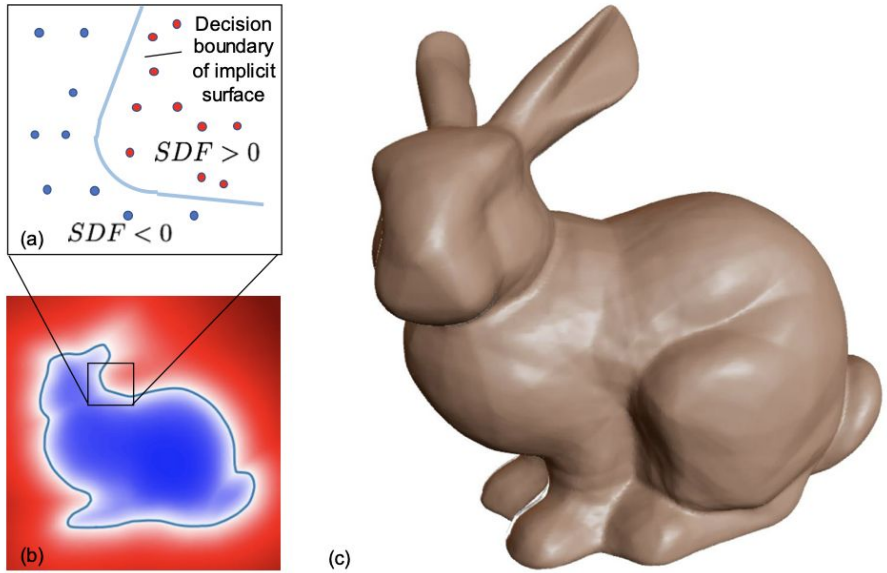


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

Aside: Auto-Decoders

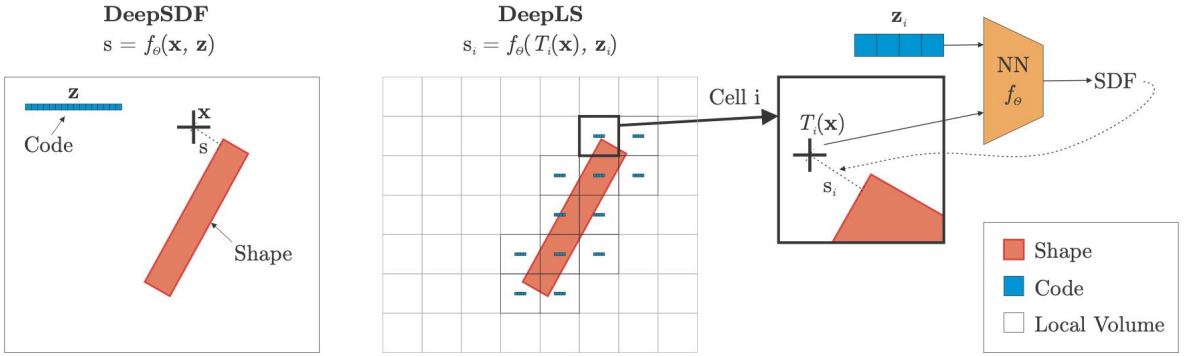
The DeepSDF work argued that it was better to just focus on the decoder component of parameterized neural fields.

- Skip encoder part and find best latent code via gradient descent.
- Then train latent code and network at the same time.
  - Technically, the latent codes become more parameters in the optimizer.

"DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation" by Park et al (2019)

Shape Representation with Patches

Like SDF, but lots of patches…



[Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction] by Chabra et al (2020)

Shape Representation with Patches

Generally makes finer-grained details easier…

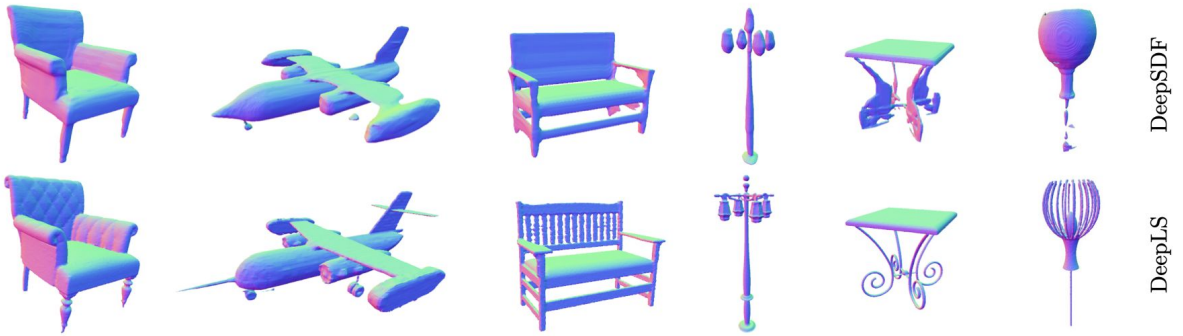


Fig. 4: Comparison between our DeepLS and DeepSDF on 3D Warehouse [1] dataset.

Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction by Chabra et al (2020)

Neural Radiance Fields

Inspired by previous (non-NN) work on volume rendering with a grid representation of space.

- Slightly expanded use of the term “field”.
  - Position + orientation as the inputs

“Representing Scenes as Neural Radiance Fields for View Synthesis” by Mildenhall et al (2020)

NeRF (neural radiance fields):
Neural networks as a *volume* representation, using volume rendering to do view synthesis.

$$(x, y, z, \theta, \phi) \rightarrow \text{color, opacity}$$

Neural Radiance Fields

Building a differentiable rendering pipeline:

- The image output can be expressed a formula based on many “looked up” values, so can compute derivatives with respect to those values.

“Representing Scenes as Neural Radiance Fields for View Synthesis” by Mildenhall et al (2020)

Volume rendering is trivially differentiable

Rendering model for ray r(t) = o + td:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

weights

colors

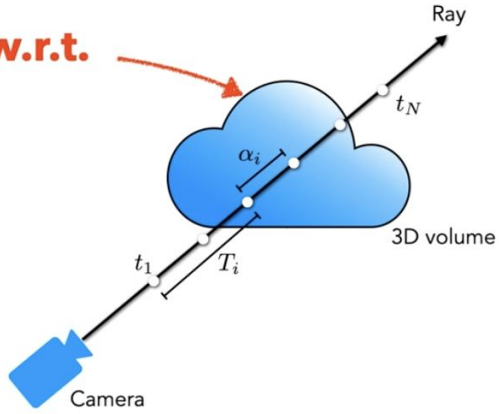
differentiable w.r.t.

How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

How much light is contributed by ray segment *i*:

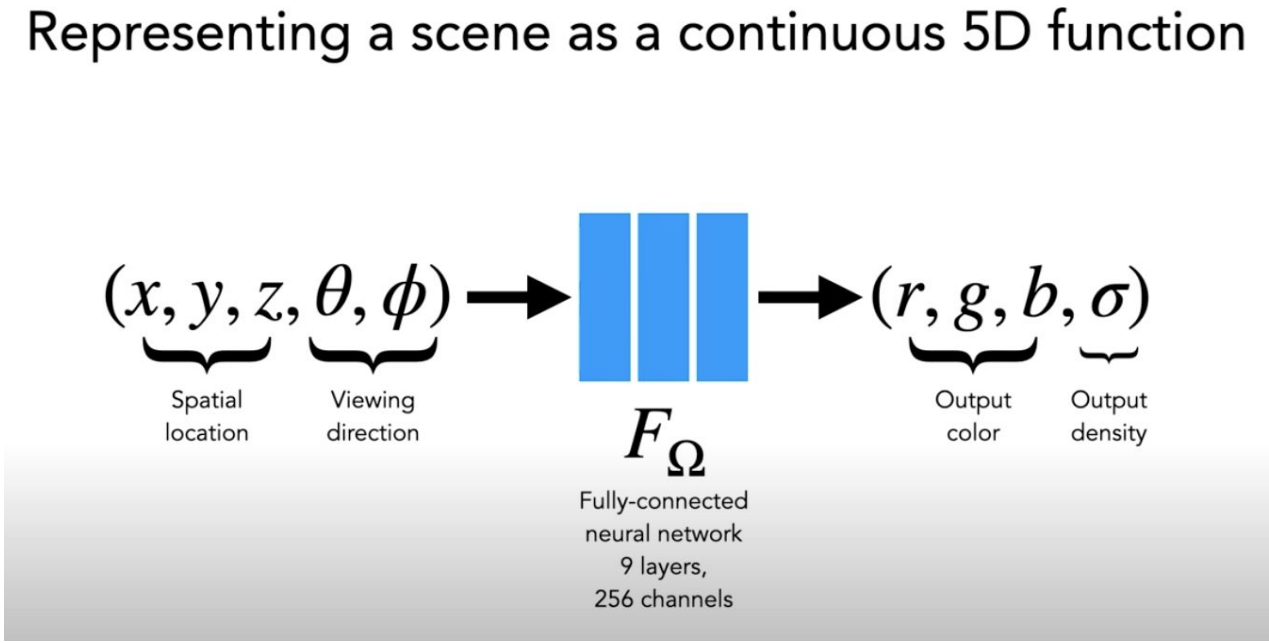
$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$



Neural Radiance Fields

Building a differentiable rendering pipeline:

- Then replace the “lookups” with a neural network.



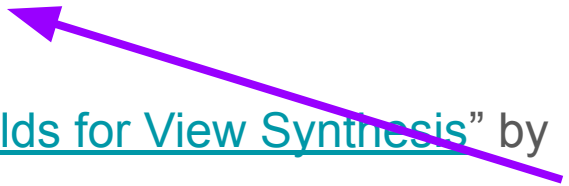
“Representing Scenes as Neural Radiance Fields for View Synthesis” by Mildenhall et al (2020)

Neural Radiance Fields

Their key points (verbatim from technical talk video)

- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering module to synthesize new views.
- Optimize rendering loss using one scene (no prior training)
- Apply positional encoding before passing coordinates into the network to recover high frequency details.

“Representing Scenes as Neural Radiance Fields for View Synthesis” by Mildenhall et al (2020)



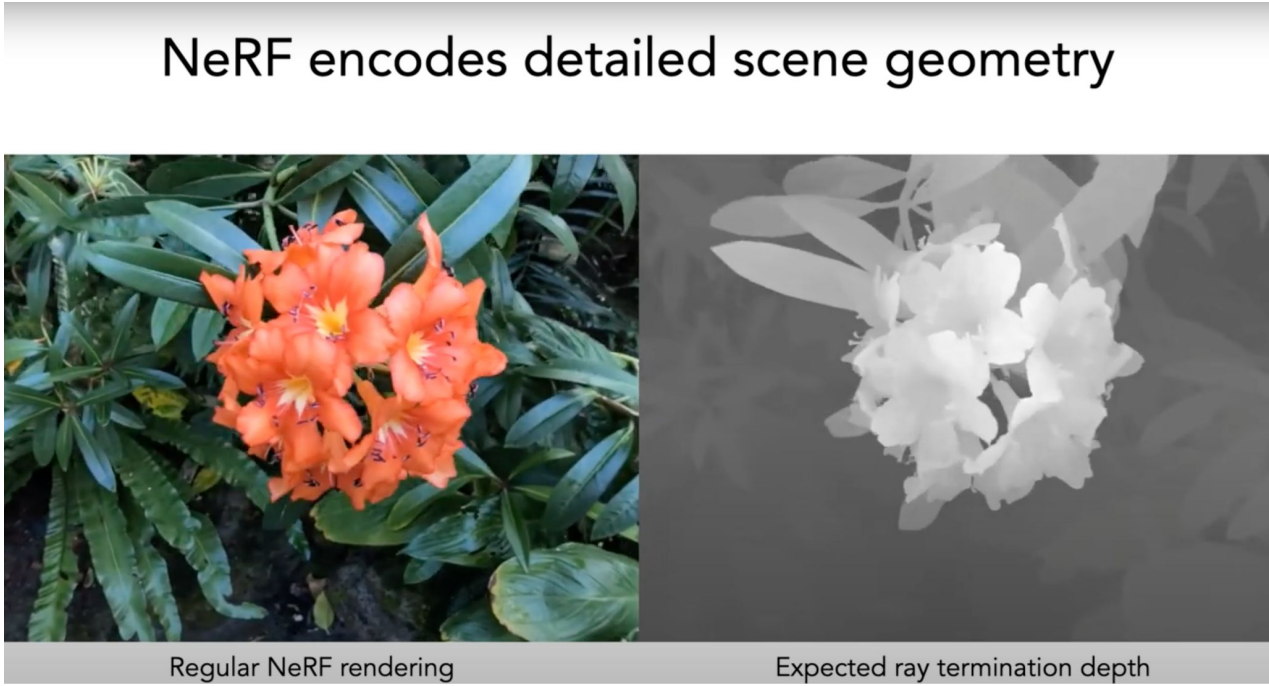
Application of Fourier features to neural fields came from this paper.

Neural Radiance Fields

The volumetric representation forces learning the 3D geometry.

- Density is one of the output parameters.
- Designed for rendering, more like transparency, but directly reveals shapes of visible objects.

"Representing Scenes as Neural Radiance Fields for View Synthesis" by Mildenhall et al (2020)



Weak Points of NeRF

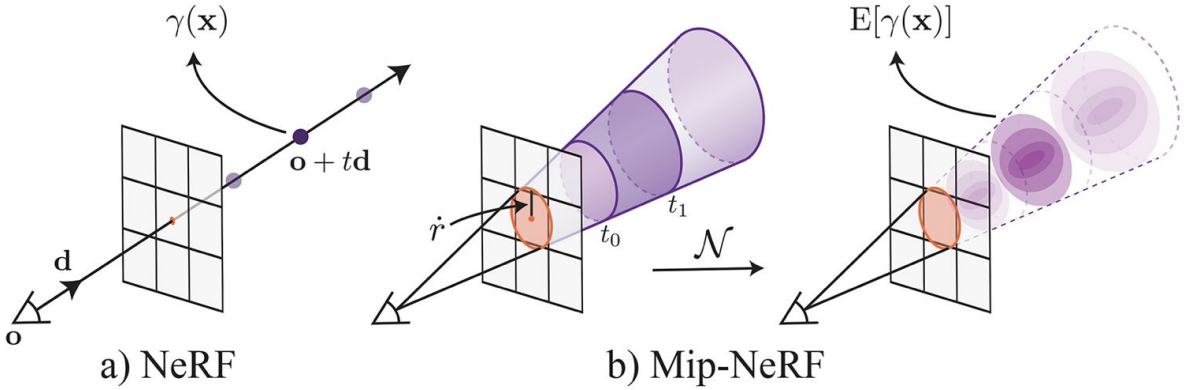
- All about rendering
- Captures relevant geometry
  - But needs camera poses.
- But no explicit semantics related to objects.
  - Much follow up work targeted here.
- Also, super slow!

Multiscale NeRF

TLDR: Use tricks with positional encoding to sample a Gaussian-shaped region of space instead of just a single point.

Let's just watch the video.

"Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields" by Barron et al (2021)

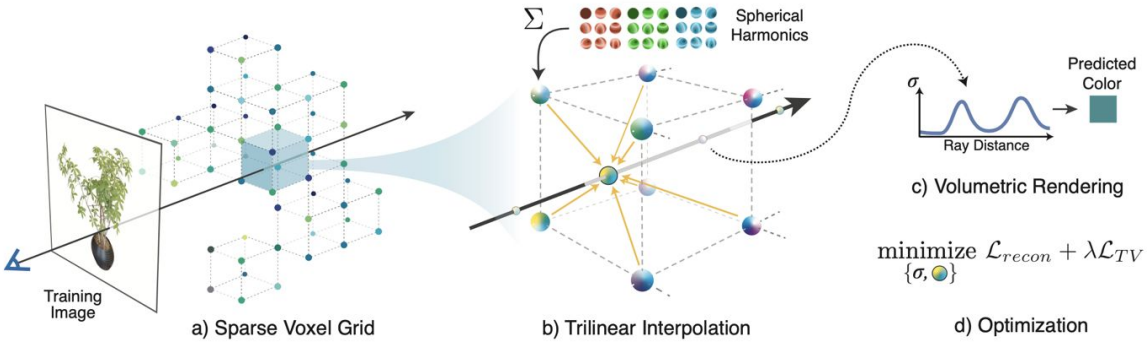


Plenoxels

Speedup NeRF by going back to grids???

- Build a model per grid location.
- Only model a sparse subset of the grid specific to the scene.
- Grid models are simpler.
- orientation → color + density
- Use spherical harmonics instead of neural networks.

"Plenoxels: Radiance Fields without Neural Networks" by Fridovich-Keil et al (2022)

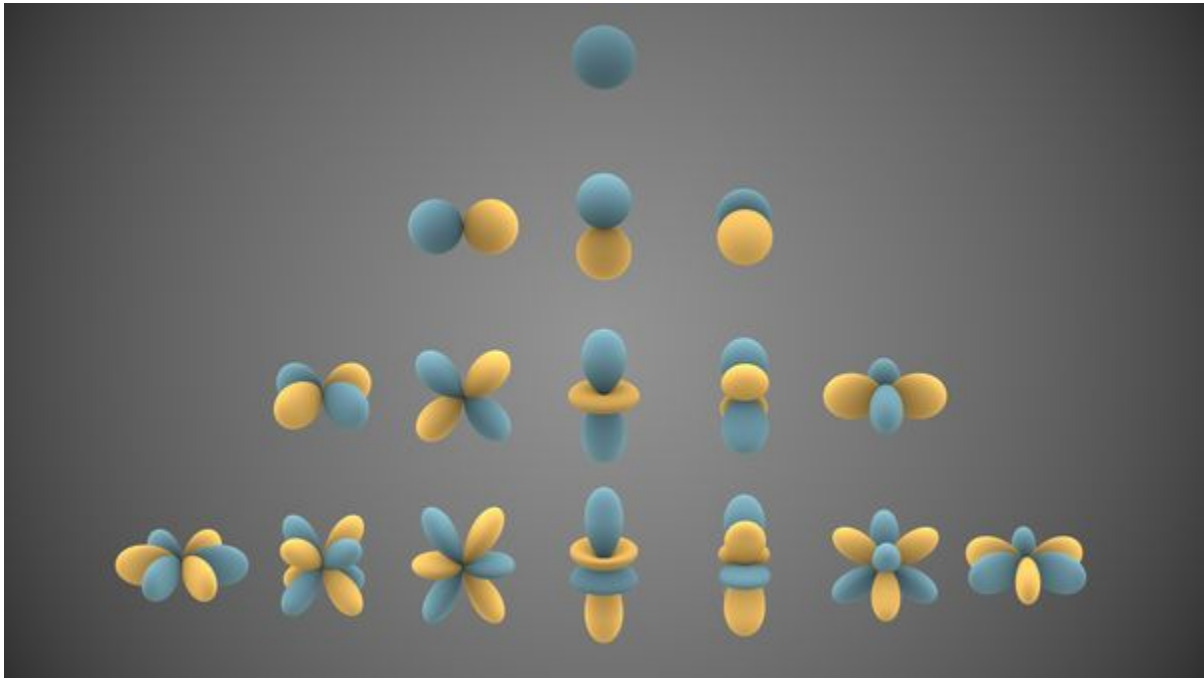


Aside: Spherical Harmonics

Like the Fourier transform but over the surface of a sphere…

“Visual representations of the first few real spherical harmonics. Blue portions represent regions where the function is positive, and yellow portions represent where it is negative. The distance of the surface from the origin indicates the absolute value of in angular direction.”

Source: https://en.wikipedia.org/wiki/Spherical_harmonics

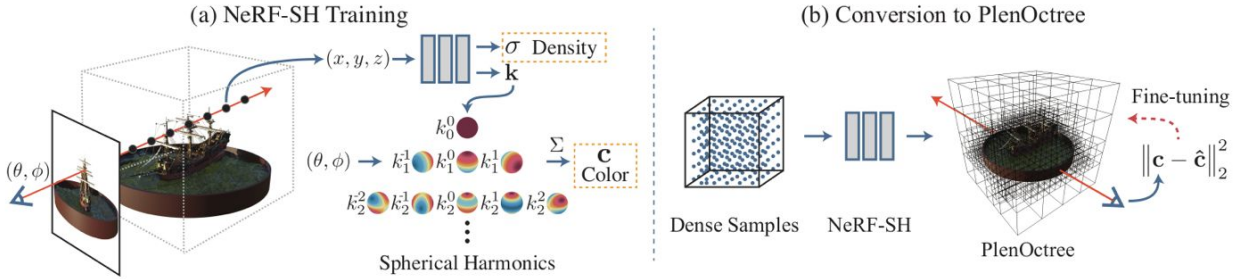


PlenOctrees

Original version:

- Build NeRF model.
- Convert into octree data structure giving extremely fast “inference”.
  - More of a lookup now.
  - Still uses spherical harmonics like Plenoxels.
  - 150fps at 800x800
- Optionally fine-tune PlenOctree rendering - still is differentiable.
- Follow up work directly trained PlenOctree from training images.

“PlenOctrees For Real-time Rendering of Neural Radiance Fields” by Yu et al (2021).



Plenoctrees

So what went right here?

- Much simpler evaluation process compared to neural network.
- Much clearer gradients. No interactions with rest of the scene.
- But hierarchical construction gave sharing effects and smoothing early on.

Plenoctrees

Loaded question:

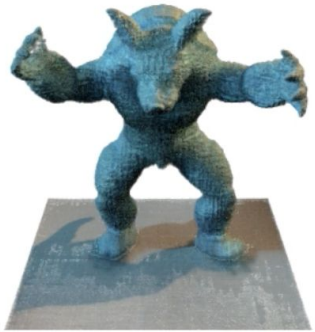
- Should neural networks always be replaced this way?
- No. At least specific to low numbers of dimensions…
  - Numbers of models and seams to fix will grow exponentially.
- Also still need base model
  - Fall back to small neural networks if small base is not available?
- End-to-end trainability
  - Rendering case fixes the path involved.
  - Other applications might involve path changes which break differentiability.

Editable NeRFs?

Since NeRFs have structure that is made for a rendering pipeline, can they be tweaked on the fly for different effects?

Watch the videos…

“NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis” by Srinivasan et al (2021)



Neural Fields and Semantics

- Neural field representations tend to be light on semantics…
  - Some structure is exposed for rendering, but relatively low level.
  - Open question how to better integrate knowledge of existing objects.
  - Current local parameterized versions may be good at identifying objects at those patches, but necessarily miss the bigger picture.

Rest of the Semester

- Thanksgiving break (11/27)
- Reinforcement learning (12/2)
- Project presentations (12/4)
- Project presentations (12/9)

Mail me if you would prefer to present 12/4. Will pick randomly otherwise.

Feedback?