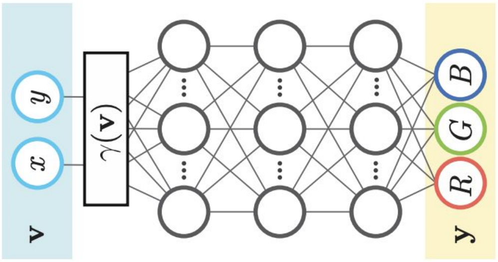
# Neural Implicit Dictionary Learning via Mixture-of-Expert Training

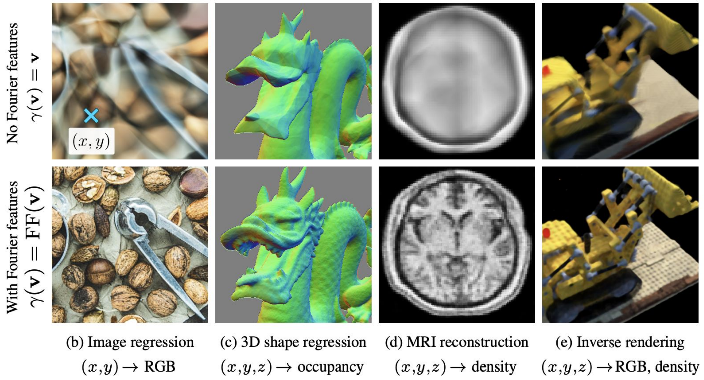
Peihao Wang, Zhiwen Fan, Tianlong Chen, Zhangyang Wang
University of Texas at Austin

# Implicit Neural Representations

1. Implicit Neural Representation (INR) is served as a powerful tool to solve inverse problems in computational photography.
    a. Parameterize signals using fully connected layers with sinusoidal activations.
    b. Construct differentiable forward function to simulate rendering/imaging process.
    c. Minimize the difference between simulated results and captured measurements via gradient descent on the network parameters.




[Tancik et al., 2021]



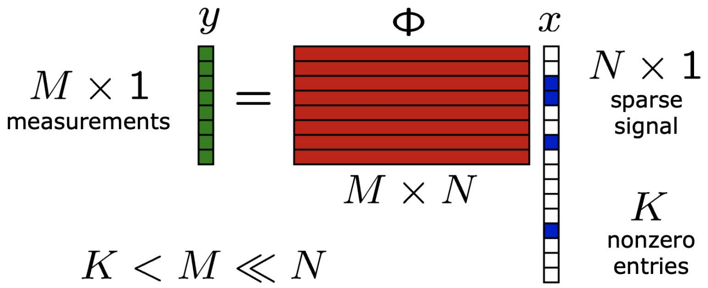
# Opportunities and Challenges

1. **Pros:**
   a. Continuous modeling of real-world signals
   b. More compactness and unlimited resolution
   c. Closed-form computation of derivatives

2. **Cons:**
   a. Fitting INR requires tedious per-scene training
   b. Solving inverse problems with INR relies on densely captured measurements
   c. INR representation is vulnerable to noisy inputs.

# Classic Solution: Dictionary Learning

1. Dictionary learning learns an over-complete basis from data and represent each sample as a sparse combination of the basis.
2. Pros:
   a. Efficient recovery of signals
   b. Robust to sparse and noisy measurements
3. Problem: Previous dictionaries are only designed in the regime of discrete signals.



# Best of two worlds: Neural Implicit Dictionary (NID)

1. We represent an INR as a sparse combination of a function dictionary
2. Each basis function is parameterized by a small coordinate-based network.
3. During training, we inverse problem *R* by jointly optimizing the basis functions and sparse coefficients.
4. When transferring to new data,  we re-use the learned dictionary and only fit the coefficients.

$$f(\boldsymbol{x}) = \alpha_1 b_1(\boldsymbol{x}) + \cdots + \alpha_n b_n(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^m$$

$$\underset{\substack{\theta_1, \cdots, \theta_n \\ \boldsymbol{\alpha}^{(1)}, \cdots, \boldsymbol{\alpha}^{(T)}}}{\arg\min} \sum_{i=1}^{T} \sum_{j=1}^{t_i} \mathcal{L}\left(\mathcal{R}(f^{(i)} | \boldsymbol{\Omega}_j^{(i)}), \boldsymbol{Y}_j^{(i)}\right) + \lambda \mathcal{P}\left(\boldsymbol{\alpha}^{(i)}, \cdots, \boldsymbol{\alpha}^{(T)}\right),$$

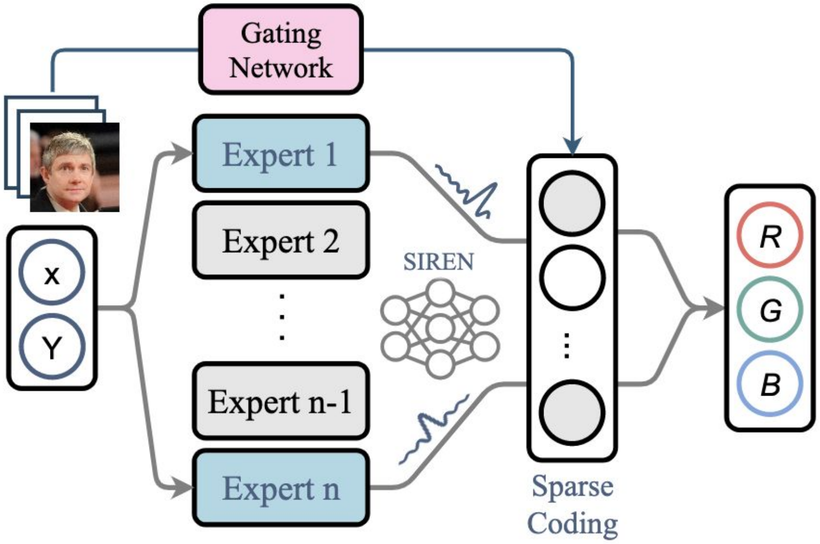
$$\text{subject to } f^{(i)}(\boldsymbol{x}) = \sum_{j=1}^{n} \alpha_j^{(i)} b_{\theta_j}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^m$$

# Implementation: Mixture-of-Expert

1. To implement sparse training of NID, we borrow the computational paradigm from mixture-of-expert layers.
2. Each expert corresponds to a function basis in the NID.
3. The gating network outputs the sparse coefficients for input sample.
4. We balance the load of experts via:

$$\mathcal{P}_{CV}\left(\boldsymbol{\alpha}^{(i)}, \cdots, \boldsymbol{\alpha}^{(T)}\right) = \frac{\mathrm{Var}(\bar{\boldsymbol{\alpha}})}{\left(\sum_{i=1}^{n} \bar{\boldsymbol{\alpha}}_i / n\right)^2},$$

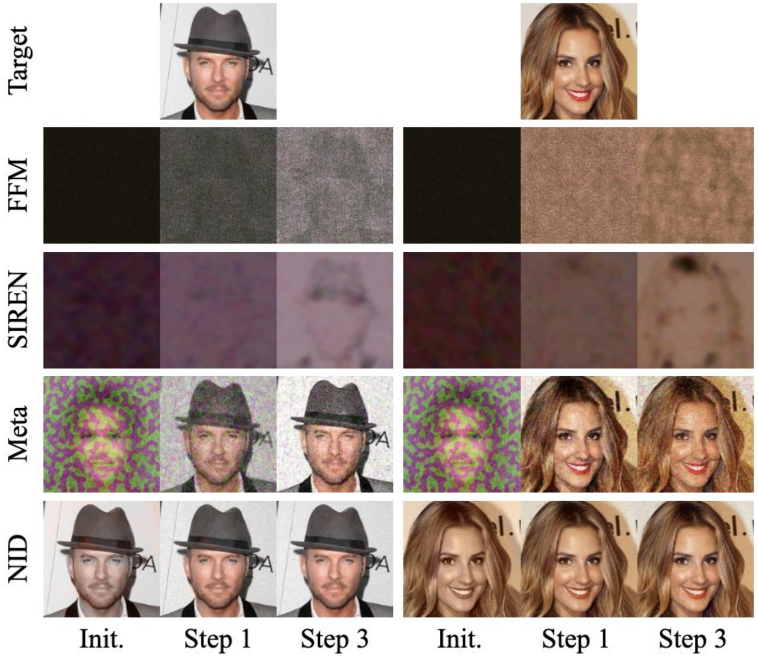
$$\text{where } \bar{\boldsymbol{\alpha}} = \sum_{i=1}^{T} \boldsymbol{\alpha}^{(i)}.$$



# Application: Instant Image Regression

1. We fit an NID on the CelebA dataset.
2. We choose a 4-layer ResNet as the gating network.
3. When fitting a new image, we first let gating network output the coefficients, and fine-tune them for three steps.

| Methods | PSNR (↑) | SSIM (↑) | LPIPS (↓) | # Params | FLOPs | Throughput |
|---|---|---|---|---|---|---|
| FFM (Tancik et al., 2020) | 22.60 | 0.636 | 0.244 | 147.8 | 20.87 | 0.479 |
| SIREN (Sitzmann et al., 2020b) | 26.11 | 0.758 | 0.379 | 66.56 | 4.217 | 0.540 |
| Meta + 5 steps (Tancik et al., 2021) | 23.92 | 0.583 | 0.322 | 66.69 | 4.217 | 0.536 |
| Meta + 10 steps (Tancik et al., 2021) | 29.64 | 0.651 | 0.182 | 66.69 | 4.217 | 0.536 |
| NID + init. ($k = 128$) | 28.75 | 0.892 | 0.061 | 8.972 | 23.30 | 30.37 |
| NID + 5 steps ($k = 128$) | 33.57 | 0.941 | 0.027 | 8.972 | 23.30 | 30.37 |
| NID + 10 steps ($k = 128$) | 35.10 | 0.954 | 0.021 | 8.972 | 23.30 | 30.37 |
| NID + init. ($k = 256$) | 30.26 | 0.919 | 0.045 | 8.972 | 29.55 | 21.23 |
| NID + 5 steps ($k = 256$) | 35.09 | 0.960 | 0.019 | 8.972 | 29.55 | 21.23 |
| NID + 10 steps ($k = 256$) | 37.75 | 0.971 | 0.012 | 8.972 | 29.55 | 21.23 |



# Application: Facial Image Inpainting

1. With the NID pre-trained on CelebA, we re-fit a group of coefficients on the NID for an image corrupted by a patch.
2. We adopt L1 error as the loss function where we assume noises are sparsely distributed.

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\arg\min} \sum_{(x,y) \in [0,D]^2} \left\| \sum_{i=1}^{n} \alpha_i b_{\theta_i}(x,y) - \boldsymbol{Y}_{xy} \right\|_1 ,$$
$$\text{subject to } \|\boldsymbol{\alpha}\|_0 \leq k,$$

Clean Corrupted SIREN Meta NID

# Application: Robust PCA on Surveillance Video

1. We train an NID on a clip of surveillance video by imposing a structured sparsity on the coefficients.
2. Then we visualize the principal components to decompose the background from the the video.

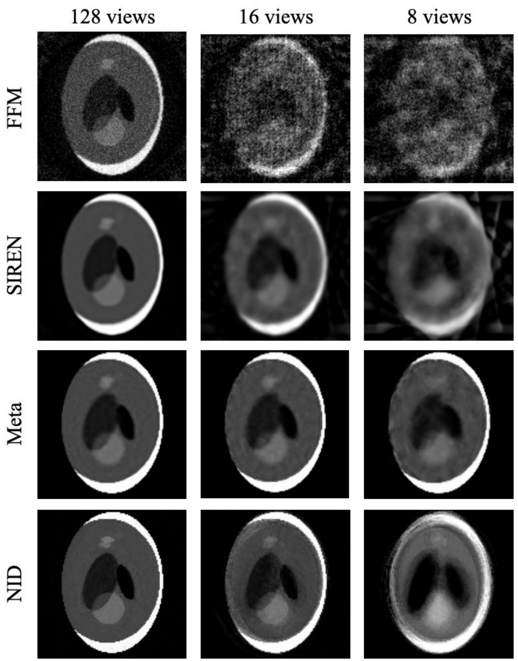
$$\underset{\substack{\theta_1,\cdots,\theta_n,\\ \alpha(t)}}{\arg\min} \sum_{t=1}^{T} \sum_{(x,y)} \left\| \sum_{i=1}^{n} \alpha_i(t) b_{\theta_i}(x,y) - \boldsymbol{Y}_{xy}^{(t)} \right\|_1 + \lambda \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{|\alpha_i(t)|}{\exp(-\beta i)},$$

Reference Frame | Annotated Transient Noises | Decomposed Background | Decomposed Transient Noise

# Application: Computed Tomography (CT) Reconstruction from Sparse Views

1. We fit an NID on the synthetic Shepp-Logan phantoms dataset.
2. Then given a sparse set of CT measurements, we fit sparse coefficients to inverse the imaging problem below:

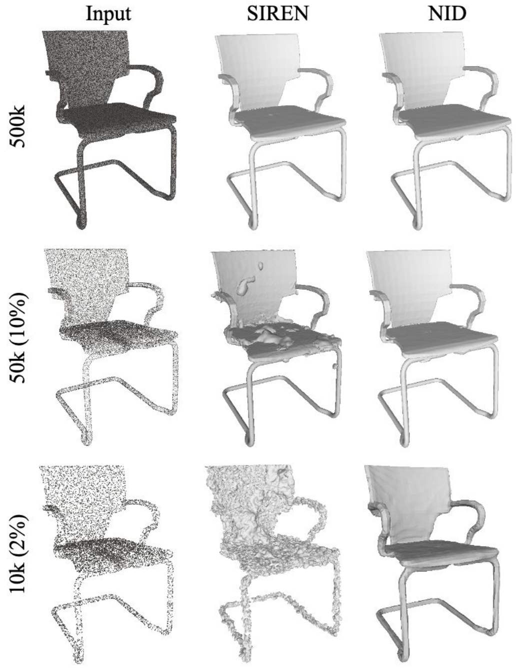
$$Y(r,\phi) = \int_{\mathbb{R}^2} f(x,y)\delta(r - x\cos\phi - y\sin\phi)\mathrm{d}x\mathrm{d}y$$



# Application: Signed Distance Function (SDF) Reconstruction from Sparse Point Clouds

1. We fit an NID on the SDFs from a category of objects in the ShapeNet.
2. Then given a sparse point cloud, we fit sparse coefficients to minimize the following objective to reconstruct SDF:

$$\arg\min_{f} \int_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x})| \mathrm{d}\boldsymbol{x} + \int_{\boldsymbol{x}\in\mathbb{R}^3\setminus\Omega} |f(\boldsymbol{x}) - d(\boldsymbol{x},\Omega)| \mathrm{d}\boldsymbol{x}$$



# Thanks for Listening