中國人民大學
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

# Deep Equilibrium Approaches to Diffusion Models


**Ashwini Pokle，Zhengyang Geng，Zico Kolter**

**Carnegie Mellon University**


**(NeurIPS 2022)**


**Fanmeng Wang**


2022-12-29

# Outline

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

# Introduction

## Diffusion Models

- A powerful class of generative models
- They have been used successfully in various settings, outperforming many other methods.
- However, diffusion models require a **long diffusion chain** (i.e. many denoising steps) to generate high-fidelity samples.
- This presents many challenges not only from **the lenses of sampling time**, but also from **the inherent difficulty in backpropagating** through these chains in order to accomplish tasks.

# Introduction

- **How to tackle this long diffusion chain ?**
  - All of existing methods still rely on **a fundamentally sequential sampling process**, imposing challenges on accelerating the sampling and for other applications like differentiating through the entire generation process.
  - This paper models the generative process of a specific class of diffusion model, the **denoising diffusion implicit model** (DDIM), as a **deep equilibrium (DEQ) model**.
  - In other words, we model **the entire sampling chain** as **a joint, multivariate fixed point system**.
  - This setup provides an elegant unification of diffusion and equilibrium models.

DDIM + DEQ

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

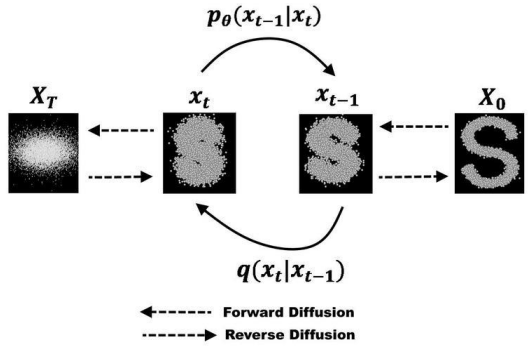
# Diffusion Models

## ■ Basic compositions

- Forward Diffusion Process： $X_0 \rightarrow X_T$ gradually convert the **original distribution** to **Standard Gaussian distribution**(or other prior distribution) by adding Gaussian noise.
- Reverse Diffusion Process： $X_T \rightarrow X_0$ gradually recover the **original distribution** from the **Standard Gaussian distribution** by denoising neural network.

## ■ Note

- Only **Reverse Diffusion Process** can be learned and $X_{0:T}$ have same dimension.
- We can also use **other noise distribution** to replace Standard Gaussian distribution.



# Diffusion Models (DDPM)



## ■ Denoising diffusion probabilistic models (DDPM)

- The Forward Diffusion Process is a **Markov chain** with fixed parameters that gradually convert the **original distribution** to the **Standard Gaussian distribution**.

$$q(x_{1:T}|x_0) = \frac{q(x_{0:T})}{q(x_0)} = \prod_{t=1}^{T} \frac{q(x_{0:t})}{q(x_{0:t-1})} = \prod_{t=1}^{T} q(x_t|x_{0:t-1}) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

$$q(x_t|x_0) = \boldsymbol{N}(x_t; \sqrt{\overline{\alpha_t}}x_0, \sqrt{1-\overline{\alpha_t}}\,\boldsymbol{I})$$

- The Reverse Diffusion Process is also a **Markov chain**, trying to recover the **original distribution** from **Standard Gaussian distribution** by **denoising neural network**.

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t,t), \sum_\theta(x_t,t))$$

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} z_\theta(x_t,t)\right)$$

In DDPM, we just use $\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$ to replace it

Noise predicted by denoising network

# Diffusion Models (DDPM)



## ■ Denoising diffusion probabilistic models (DDPM)

➢ We can use a surrogate **variational lower bound** to train this model:

$$L = \mathbb{E}_q[-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \sum_t D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) + D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)]$$

then we can further get

$$L_t^{simple} = E_{x_0,z}[||z_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t, t)||^2]$$

so we just need to minimize $||\bar{z}_t - z_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{z}_t, t)||$ to train this model

➢ However, **the length T of a diffusion process is usually large** (e.g., T = 1000) as it contributes to a better approximation of Gaussian conditional distributions in the generative process.

➢ Therefore, **sampling from DDPM can be visibly slower** compared to other deep generative models like GANs.

# Diffusion Models (DDIM)

## ■ Denoising diffusion implicit models (DDIM)

- DDIM rewrites the forward process of DDPM into a **non-Markovian one** that leads to a "shorter" and deterministic generative process.
- Essentially, DDIM constructs **a nearly non-stochastic scheme** that can quickly sample from the learned data distribution without introducing additional noises.
- In fact, DDIM can just use the trained DDPM as long as

$$\int p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)p(\boldsymbol{x}_t|\boldsymbol{x}_0)d\boldsymbol{x}_t = p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)$$

- And the scheme to generate a sample $x_{t-1}$ by given $x_t$ is:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t) + \sigma_t\boldsymbol{\epsilon}_t$$

setting $\sigma_t = 0$ for all $t$ gives to a DDIM

**Train a DDPM**

**Then using a DDIM to generate samples based on the trained DDPM**

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

# Deep Equilibrium Models (DEQ)



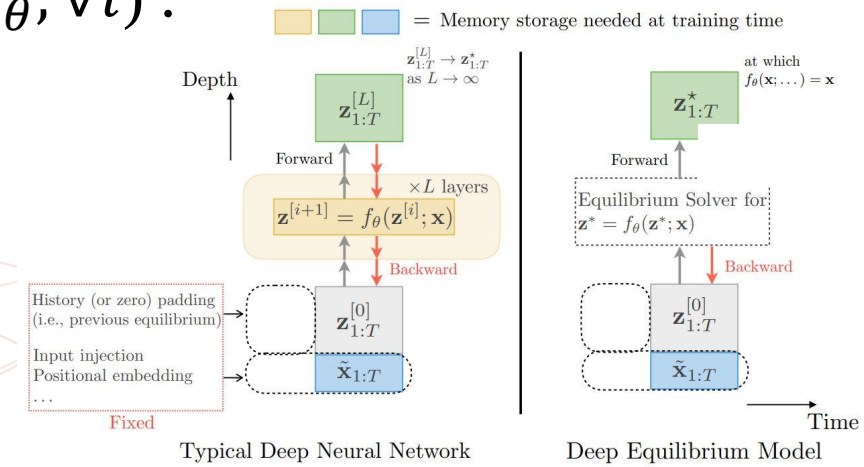
## ■ Definition

➢ For typical deep neural network, it can only have **limited layers** and

$$\mathbf{z}^{[i+1]} = f_\theta^{[i]}\left(\mathbf{z}^{[i]}; \mathbf{x}\right) \quad \text{for } i = 0, \ldots, L-1$$

where $x$ is the input injection, $z^{[i]}$ is the hidden state of $i$-th layer, and $f_\theta^{[i]}$ is a layer that defines the feature transformation.

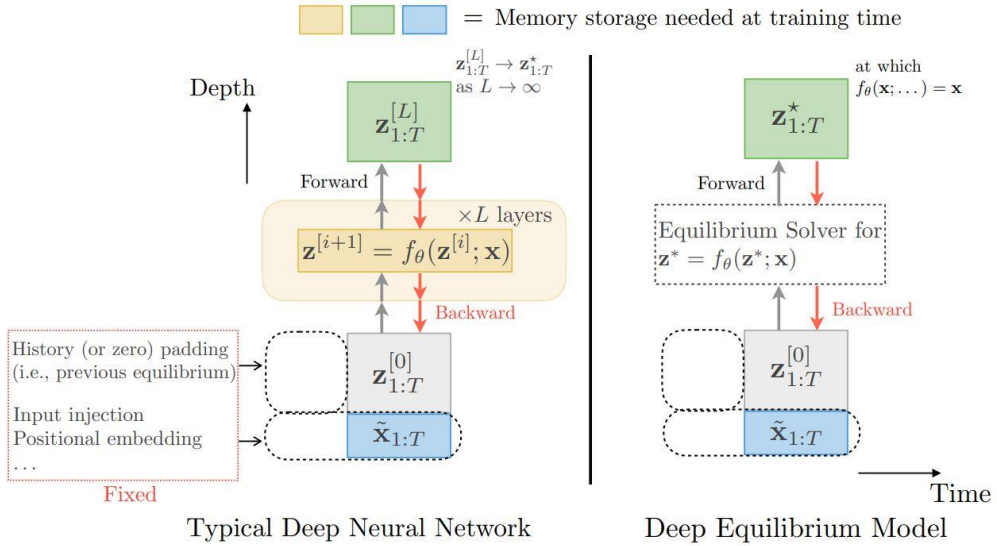
➢ For deep equilibrium models, it has **infinite depth** and the same parameters for each layer (i.e. $f_\theta^{[i]} = f_\theta, \forall i$) .



# Deep Equilibrium Models (DEQ)



## ■ Definition




- For deep equilibrium models, it has **infinite depth** but the same parameters for each layer (i.e. $f_\theta^{[i]} = f_\theta, \forall i$) .
- Deep equilibrium models are proposed to directly compute this fixed point $z^*$ as the output, i.e.

$$f_\theta(\mathbf{z}^*; \mathbf{x}) = \mathbf{z}^*$$

# Deep Equilibrium Models (DEQ)



## ■ Advantages

- We just need to storage $z^*$, so DEQ can save memory.
- In fact, DEQ is known as a principled framework for characterizing convergence and energy minimization in deep learning.

## ■ How to find the fixed point $z^*$ ?

- The equilibrium state $z^*$ can be solved by **black-box solvers** like Broyden's method or Anderson acceleration.
- In this paper, we use **Anderson acceleration** to find the fixed point $z^*$.

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

# Method: DEQ-DDIM



## A DEQ formulation of DDIMs (DEQ-DDIM)

- We build a DEQ version of the DDIM involves representing all the states $x_{0:T}$ **simultaneously** within the DEQ state.
- For DDIM, we can generate a sample $x_{t-1}$ by given $x_t$

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}} \cdot \boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t), \quad t = [1, \ldots, T] \qquad (7)$$

first we rearrange the terms in Eq. (7):

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\mathbf{x}_t + \left(\sqrt{1-\alpha_{t-1}} - \sqrt{\frac{\alpha_{t-1}(1-\alpha_t)}{\alpha_t}}\right)\epsilon_\theta^{(t)}(\mathbf{x}_t) \qquad (8)$$

let $c_1^{(t)} = \sqrt{1-\alpha_{t-1}} - \sqrt{\frac{\alpha_{t-1}(1-\alpha_t)}{\alpha_t}}$, then we can get

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\mathbf{x}_t + c_1^{(t)}\epsilon_\theta^{(t)}(\mathbf{x}_t) \qquad (9)$$

# Method: DEQ-DDIM



## ■ A DEQ formulation of DDIMs (DEQ-DDIM)

➢ Since

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}}\mathbf{x}_t + c_1^{(t)}\epsilon_\theta^{(t)}(\mathbf{x}_t) \tag{9}$$

By induction, we can rewrite the above equation as:

$$\mathbf{x}_{T-k} = \sqrt{\frac{\alpha_{T-k}}{\alpha_T}}\mathbf{x}_T + \sum_{t=T-k}^{T-1}\sqrt{\frac{\alpha_{T-k}}{\alpha_t}}c_1^{(t+1)}\epsilon_\theta^{(t+1)}(\mathbf{x}_{t+1}), \quad k \in [0, .., T] \tag{10}$$

➢ This defines a "**fully-lower-triangular**" inference process, where the update of $x_t$ depends on the noise prediction network $\epsilon_\theta$ applied to all subsequent states $x_{t+1:T}$

# Method: DEQ-DDIM



## ■ A DEQ formulation of DDIMs (DEQ-DDIM)

$$\mathbf{x}_{T-k} = \sqrt{\frac{\alpha_{T-k}}{\alpha_T}}\mathbf{x}_T + \sum_{t=T-k}^{T-1} \sqrt{\frac{\alpha_{T-k}}{\alpha_t}} c_1^{(t+1)} \epsilon_\theta^{(t+1)}(\mathbf{x}_{t+1}), \quad k \in [0, .., T] \qquad (10)$$

➢ $h(\cdot)$ represent the function that performs the operations in the equations (10) for a latent $x_t$ **at timestep *t***

$\tilde{h}(\cdot)$ represent the function that performs the same set of operations across **all the timesteps simultaneously**.

We can write the above set of equations as a fixed point system:

$$\begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{x}_{T-2} \\ \vdots \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}_T) \\ h(\mathbf{x}_{T-1:T}) \\ \vdots \\ h(\mathbf{x}_{1:T}) \end{bmatrix}$$

or

$$\mathbf{x}_{0:T-1} = \tilde{h}(\mathbf{x}_{0:T-1}; \mathbf{x}_T)$$

# Method: DEQ-DDIM



- **A DEQ formulation of DDIMs (DEQ-DDIM)**

$$\begin{bmatrix} \mathbf{x}_{T-1} \\ \mathbf{x}_{T-2} \\ \vdots \\ \mathbf{x}_0 \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}_T) \\ h(\mathbf{x}_{T-1:T}) \\ \vdots \\ h(\mathbf{x}_{1:T}) \end{bmatrix}$$

$$\mathbf{x}_{0:T-1} = \tilde{h}(\mathbf{x}_{0:T-1}; \mathbf{x}_T)$$

- The above system of equations represent a DEQ with $x_T \sim \mathrm{N}(0, \mathrm{I})$ as input injection.
- We can simultaneously solve for the roots of this system of equations through black-box solvers like **Anderson acceleration**.
- Let $g(\mathbf{x}_{0:T-1}; \mathbf{x}_T) = h(\mathbf{x}_{0:T-1}; \mathbf{x}_T) - \mathbf{x}_{0:T-1}$, then we have

$$\mathbf{x}^*_{0:T} = \mathrm{RootSolver}(g(\mathbf{x}_{0:T-1}; \mathbf{x}_T))$$

# Method: DEQ-DDIM

## ■ Efficient Inversion of DDIM / How to train DDIM ?

➢ For an input image $x_0$, and a generated image $\hat{x}_0$, our task is to **minimize the squared-Frobenius distance** between these images:

$$\mathcal{L}(\mathrm{x}_0, \hat{\mathrm{x}}_0) = \|\mathrm{x}_0 - \hat{\mathrm{x}}_0\|_F^2$$

➢ **The native approach**:

Randomly sample $x_T \sim N\,(0, I)$, and update it via gradient descent by first estimating $x_0$ using the generative process in Eq. (7) and backpropagating through this process after computing the loss objective in (14).

**Algorithm 1** A naive algorithm to invert DDIM

**Input:** A target image $\mathrm{x}_0 \sim \mathcal{D}$, $\epsilon_\theta(\mathrm{x}_t, t)$ a trained denoising diffusion model, $N$ the total number of epochs

▷ $f$ denotes the sampling process in Eq (7)

**Initialize** $\hat{\mathrm{x}}_T \sim \mathcal{N}(0, \mathbf{I})$

**for** epochs from 1 to $N$ **do**

  **for** $t = T, ..., 1$ **do**

    Sample $\hat{\mathrm{x}}_{t-1} = f(\hat{\mathrm{x}}_t; \epsilon_\theta(\hat{\mathrm{x}}_t, t))$

  **end for**

  Take a gradient descent step on

$$\nabla_{\hat{\mathrm{x}}_T} \|\hat{\mathrm{x}}_0 - \mathrm{x}_0\|_F^2$$

**end for**

**Output:** $\hat{\mathrm{x}}_T$

# Method: DEQ-DDIM



## ■ Efficient Inversion of DDIM / How to train DDIM ?

➢ We can use the DEQ formulation to develop **a much more efficient inversion method**

We apply **implicit function theorem** (IFT) to the fixed point to **compute gradients of the loss** $\mathcal{L}(\mathbf{x}_0, \hat{\mathbf{x}}_0) = \|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_F^2$ w.r.t. $(\cdot)$:

$$\frac{\partial \mathcal{L}}{\partial (\cdot)} = -\frac{\partial \mathcal{L}}{\partial \mathbf{x}^*_{0:T}} \left( J^{-1}_{g_\theta} \Big|_{\mathbf{x}^*_{0:T}} \right) \frac{\partial \tilde{h}(\mathbf{x}^*_{0:T-1}; \mathbf{x}_T)}{\partial (\cdot)} \tag{15}$$

where $(\cdot)$ could be any of the latent states $x_1$,..., $x_T$ , and $J^{-1}_{g_\theta}|_{\mathbf{x}^*_{0:T}}$ is the inverse Jacobian of $g(x_{0:T-1}; x_T)$ evaluated at $x^*_{0:T}$

**Algorithm 2** Inverting DDIM with DEQ

**Input:** A target image $\mathbf{x}_0 \sim \mathcal{D}$, $\epsilon_\theta(\mathbf{x}_t, t)$ a trained denoising diffusion model, $N$ the total number of epochs

▷ $g$ is the function in Eq. (12)

**Initialize** $\hat{\mathbf{x}}_{0:T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

**for** epochs from 1 to $N$ **do**

▷ Disable gradient computation

$\mathbf{x}^*_{0:T} = \text{RootSolver}(g(\mathbf{x}_{0:T-1}); \mathbf{x}_T)$

▷ Enable gradient computation

Compute Loss $\mathcal{L}(\mathbf{x}_0, \mathbf{x}^*_0)$

Use the 1-step grad to compute $\partial \mathcal{L} / \partial \mathbf{x}_T$

Take a gradient descent step using above

**end for**

**Output:** $\mathbf{x}^*_T$

# Method: DEQ-DDIM



## ■ Efficient Inversion of DDIM / How to train DDIM ?

➢ We can use the DEQ formulation to develop **a much more efficient inversion method**

$$\frac{\partial \mathcal{L}}{\partial(\cdot)} = -\frac{\partial \mathcal{L}}{\partial \mathbf{x}^*_{0:T}} \left( J^{-1}_{g_\theta}\big|_{\mathbf{x}^*_{0:T}} \right) \frac{\partial \tilde{h}(\mathbf{x}^*_{0:T-1}; \mathbf{x}_T)}{\partial(\cdot)} \tag{15}$$

We can instead use an approximation to Eq. (15), i.e.,

$$\frac{\partial \mathcal{L}}{\partial(\cdot)} = -\frac{\partial \mathcal{L}}{\partial \mathbf{x}^*_{0:T}} \mathbf{M} \frac{\partial \tilde{h}(\mathbf{x}^*_{0:T-1}; \mathbf{x}_T)}{\partial(\cdot)}$$

where M is an approximation of $J^{-1}_{g_\theta}\big|_{\mathbf{x}^*_{0:T}}$ when setting M = I, i.e.1-step gradient.

**Algorithm 2** Inverting DDIM with DEQ

**Input:** A target image $\mathbf{x}_0 \sim \mathcal{D}$, $\epsilon_\theta(\mathbf{x}_t, t)$ a trained denoising diffusion model, $N$ the total number of epochs

▷ $g$ is the function in Eq. (12)

**Initialize** $\hat{\mathbf{x}}_{0:T} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

**for** epochs from 1 to $N$ **do**

▷ Disable gradient computation

$\mathbf{x}^*_{0:T} = \text{RootSolver}(g(\mathbf{x}_{0:T-1}); \mathbf{x}_T)$

▷ Enable gradient computation

Compute Loss $\mathcal{L}(\mathbf{x}_0, \mathbf{x}^*_0)$

Use the 1-step grad to compute $\partial\mathcal{L}/\partial\mathbf{x}_T$

Take a gradient descent step using above

**end for**

**Output:** $\mathbf{x}^*_T$

# Method: DEQ-DDIM



## ■ Efficient Inversion of DDIM / How to train DDIM ?

➢ We can use the DEQ formulation to develop **a much more efficient inversion method**

In this work, we follow Geng et al. [28] to further add a damping factor to the 1-step gradient. **The forward pass is given by**:

$$\mathbf{x}^*_{0:T} = \text{RootSolver}(g(\mathbf{x}_{0:T-1}); \mathbf{x}_T)$$
$$\mathbf{x}^*_{0:T} = \tau \cdot \tilde{h}(\mathbf{x}^*_{0:T-1}; \mathbf{x}^*_T) + (1-\tau) \cdot \mathbf{x}^*_{0:T}$$

The gradients for the backward pass can be computed through **standard autograd packages.**

**Algorithm 3** PyTorch-style pseudocode for inversion with DEQ-DDIM

```
# x0: a target image for inversion
# all_xt: all the latents in the diffusion chain or its subsequence; xT is at
    index 0, x0 is at the last index
# func: A function that performs the required operations on the fixed-point
    system for a single timestep
# solver: A fixed-point solver like Anderson acceleration
# optimizer: an optimization algorithm like Adam
# tau: the damping factor τ for phantom gradient
# num_epochs: the max number of epochs

def forward(func, x):
    with torch.no_grad():
        z = solver(func, x)
    z = tau * func(z) + (1 - tau) * z
    return z

def invert(func, all_xt, x0, optimizer, num_epochs):
    for epoch in range(num_epochs):
        optimizer.zero_grad()
        xt_pred = forward(func, all_xt)
        loss = (xt_pred[-1] - x0).norm(p='fro')
        loss.backward()
        optimizer.step()
    return all_xt[0]
```

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

# Experiments

## Setup

- We consider **four datasets** that have images of different resolutions for our experiments: CIFAR10(32×32), CelebA (64×64), LSUN Bedroom (256×256) and LSUN Outdoor Church (256×256).
- For all the experiments, we use **Anderson acceleration** as the default fixed point solver.
- While training DEQs for model inversion, we use the **1-step gradient** to compute the backward pass.

# Experiments

## ■ Convergence of DEQ-DDIM

➢ We verify that **DEQ-DDIM converges** to a fixed point by plotting the values of $\|\tilde{h}(\mathbf{x}_{0:T}) - \mathbf{x}_{0:T}\|_2$ over Anderson solver steps.

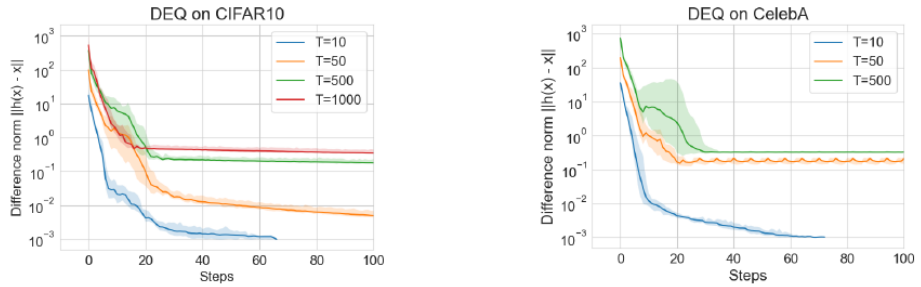



Figure 1: DEQ-DDIM finds an equilibrium point. We plot the absolute fixed-point convergence $\|\tilde{h}(\mathbf{x}_{0:T}) - \mathbf{x}_{0:T}\|_2$ during a forward pass of DEQ for CIFAR-10 (left) and CelebA (right) for different number of steps $T$. The shaded region indicates the maximum and minimum value encountered during any of the 25 runs.

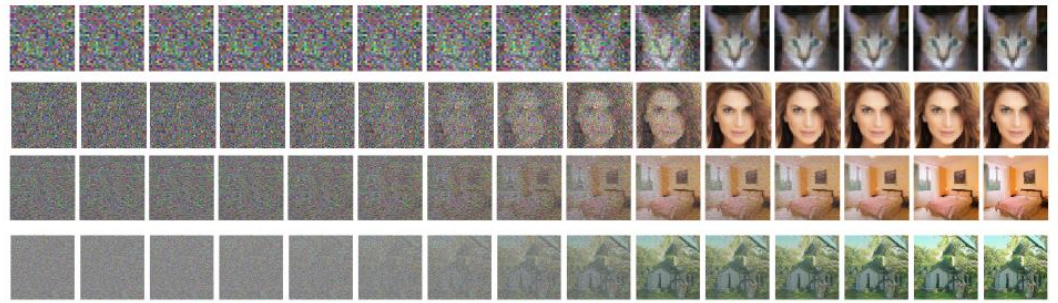


Figure 2: Visualization of intermediate latents $\mathbf{x}_t$ of DEQ-DDIM after 15 forward steps with Anderson solver for CIFAR-10 (first row, $T = 500$), CelebA (second row, $T = 500$), LSUN Bedroom (third row, $T = 50$, and LSUN Outdoor Church (fourth row, $T = 50$). For $T = 500$, we visualize every $50^{th}$ latent, and for $T = 50$, we visualize every $5^{th}$ latent. In addition, we also visualize $\mathbf{x}_{0:4}$ in the last 5 columns.

Our experiments demonstrate that DEQ-DDIM is able to generate high-quality images in as few as 15 Anderson solver steps on diffusion chains that were trained on a much larger number of steps T.

# Experiments

## ■ Sample quality of images generated with DEQ-DDIM

➢ In Table-1, we verify that DEQ-DDIM can generate images of comparable quality to DDIM by reporting Fréchet Inception Distance (**FID**).

| Dataset | T | DDPM | | DDIM | | DEQ-DDIM | |
|---|---|---|---|---|---|---|---|
| | | FID | Time | FID | Time | FID | Time |
| CIFAR10 | 1000 | **3.17** | 24.45s | 4.07 | 20.16s | 3.79 | **2.91s** |
| CelebA | 500 | 5.32 | 14.95s | 3.66 | 10.31s | **2.92** | **5.12s** |
| LSUN Bedroom | 25 | 184.05 | 1.72s | 8.76 | **1.19s** | **8.73** | 3.82s |
| LSUN Church | 25 | 122.18 | 1.77s | **13.44** | **1.68s** | 13.55 | 3.99s |

Table 1: FID scores and time for single image generation for DDPM, DDIM and DEQ-DDIM.

# Experiments

## ■ Model Inversion of DDIM with DEQs

➢ We observe that DEQ-DDIM converges faster and has much lower loss values than the baseline method induced by DDIM.

| Dataset | T | Baseline | | DEQ-DDIM | |
|---|---|---|---|---|---|
| | | Min loss ↓ | Avg Time (mins) ↓ | Min loss ↓ | Avg Time (mins) ↓ |
| CIFAR10 | 100 | $15.74 \pm 8.7$ | $49.07 \pm 1.76$ | $\mathbf{0.76 \pm 0.35}$ | $12.99 \pm 0.97$ |
| CIFAR10 | 10 | $2.59 \pm 3.67$ | $14.36 \pm 0.26$ | $\mathbf{0.68 \pm 0.32}$ | $2.54 \pm 0.41$ |
| CelebA | 20 | $14.13 \pm 5.04$ | $30.09 \pm 0.57$ | $\mathbf{1.03 \pm 0.37}$ | $28.09 \pm 1.76$ |
| Bedroom | 10 | $1114.49 \pm 795.86$ | $26.41 \pm 0.17$ | $\mathbf{36.37 \pm 22.86}$ | $33.7 \pm 1.05$ |
| Church | 10 | $1674.68 \pm 1432.54$ | $29.7 \pm 0.75$ | $\mathbf{47.94 \pm 24.78}$ | $33.54 \pm 3.02$ |

Table 3: Comparison of minimum loss and average time required to generate an image. All the results have been reported on 100 images. See Appendix A for detailed training settings.

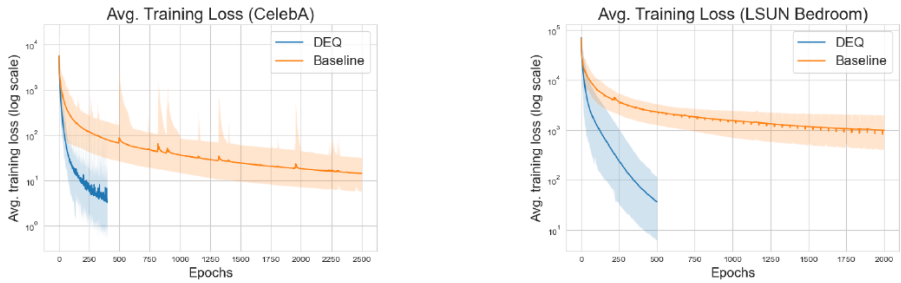



Figure 3: Training loss for CelebA and LSUN Bedroom over epochs. DEQ-DDIM converges in fewer epochs, and achieves lower values of loss compared to the baseline. The shaded region indicates the maximum and minimum value of loss encountered during any of the 100 runs.

- Introduction
- Diffusion Models (DDPM && DDIM)
- Deep Equilibrium Models (DEQ)
- A Deep Equilibrium Approach to DDIMs (DEQ-DDIM)
- Experiments
- Conclusion

# Conclusion

- We propose an approach to elegantly unify **diffusion models** and **deep equilibrium (DEQ) models**.
- We model the entire sampling chain of the denoising diffusion implicit model (DDIM) as **a joint, multivariate (deep) equilibrium model**. This setup replaces the traditional sequential sampling process with **a parallel one**, thereby enabling us to enjoy speedup obtained from multiple GPUs.
- Further, we can leverage **inexact gradients** to optimize the entire sampling chain quickly, which results in **significant gains in model inversion**.

# Thank You for listening!

2022-12-29