
 Member-only story


Reimagining Diffusion Models: Autoregressive Priors for Efficient Initialization


Exploring a Novel Approach to Diffusion Initialization with Intuitive Illustrations, Applications





Shenggang Li · Following


Published in Towards AI · 11 min read · Mar 8, 2025


 123











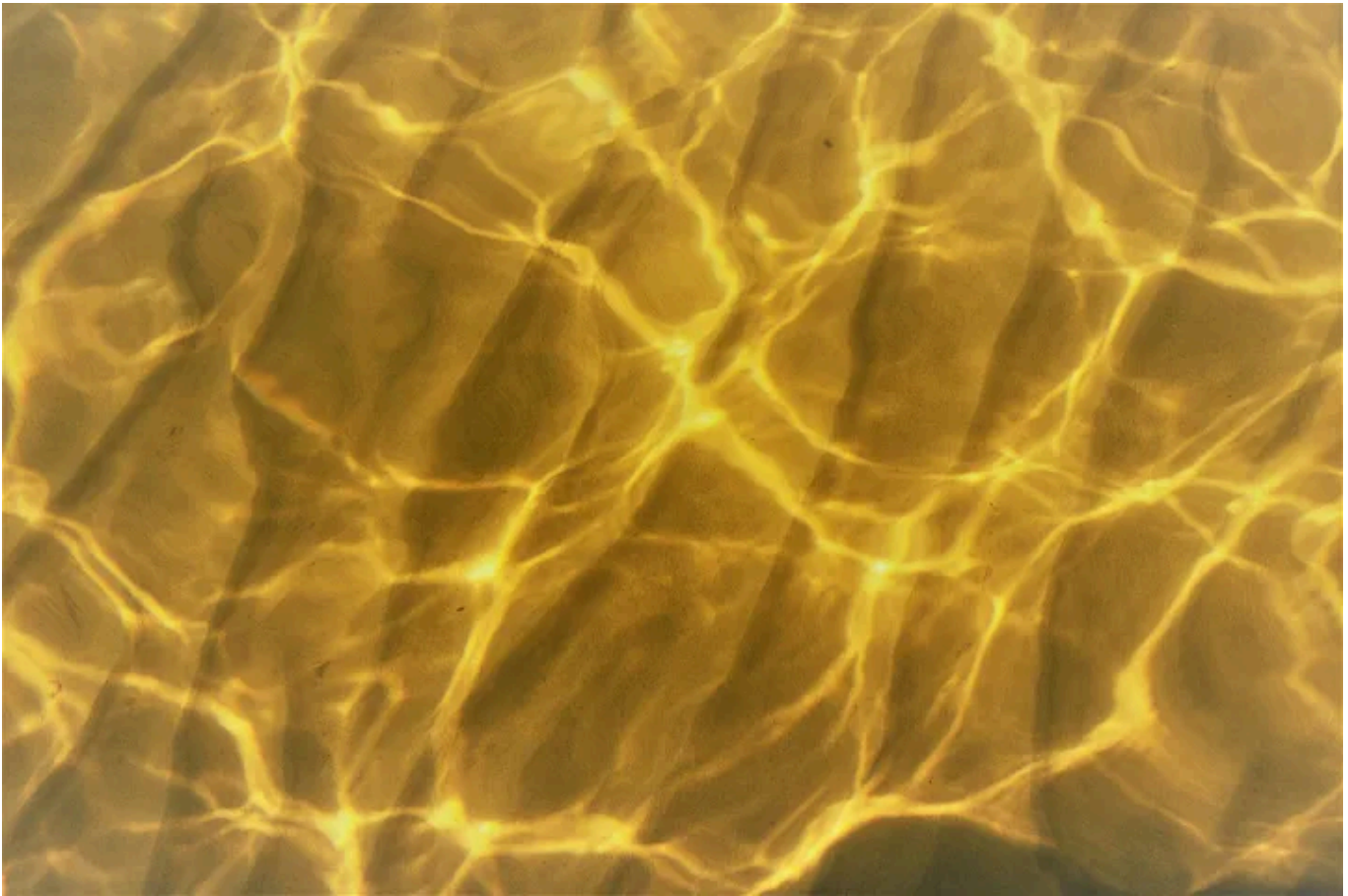


Photo by [Gary Fultz](#) on [Unsplash](#)

Introduction

Diffusion models have become a cornerstone of modern AI, especially in generative tasks like creating realistic images or high-quality audio. They're like digital artists, transforming random noise into stunningly detailed outputs step-by-step. This meticulous approach has made diffusion models a game-changer in the AI world.

Typically, these models begin their work with pure Gaussian noise, which acts as the blank canvas. While effective, this starting point doesn't take advantage of prior knowledge about the data structure, potentially slowing down the process and affecting sample quality. Imagine if we could give these models a smarter head start.

That's where Autoregressive Priors (*ARPs*) come in. I introduce a new approach that integrates Autoregressive Models (*ARMs*) at the start of the diffusion process, adding structure instead of relying on pure Gaussian noise. This speeds up generation and enhances sample quality. I will explore how *ARPs* improve diffusion models, break down their mechanics, and compare them with traditional methods.

Understanding Diffusion Models

Imagine restoring a faded photograph: you begin with a nearly blank canvas (random noise) and repeatedly refine it, eventually recovering the original image. Diffusion models operate similarly — they start from random noise and iteratively denoise until meaningful data emerges. In this sense, they can generate images, audio, or text from pure randomness.

To visualize the process, think of cleaning a messy room. Initially, it's so cluttered you can't see the floor (random noise). Each cleanup step reveals a bit more of the underlying room structure (denoising). Eventually, the space is neat and organized — just like a diffusion model uncovering real data from noise.

These models follow a probabilistic framework in which noise is deliberately added to data, and the model is trained to reverse that process. The sequence of adding noise and learning how to remove it underpins the entire diffusion mechanism.

Let's break down how diffusion models work step by step.

Forward Process: Add Noise

$X(0) \xrightarrow{\text{noise}} X(1) \xrightarrow{\text{noise}} \dots \xrightarrow{\text{noise}} X(T-1) \xrightarrow{\text{noise}} X(T)$

(Clear / Original)

(Mostly Noise)

[Noisy intermediate states at each time step]

Reverse Process: Remove Noise

$X(T) \xrightarrow{\text{denoise}} X(T-1) \xrightarrow{\text{denoise}} \dots \xrightarrow{\text{denoise}} X(1) \xrightarrow{\text{denoise}} X(0)$

(Almost random)

(Recovered / Clean)

[Model learns to reverse each noising step]

Step 1: Forward Process (Adding Noise)

In the forward process, the goal is to gradually corrupt the original data x_0 with Gaussian noise over T steps. This process is modeled as a Markov chain:

$$\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \dots \rightarrow \mathbf{x}_T$$

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1 - \alpha_t)I)$$

Here:

- x_0 : Original data (e.g., an image or other structured data).

- x_t : The noisy version of the data at step t .
- α_t : A variance schedule controlling the noise added at each step.

By the final step T , the data x_T becomes indistinguishable from pure Gaussian noise. This forward diffusion process ensures that the original data is gradually degraded in a controlled manner.

Step 2: Reverse Process (Denoising)

The **reverse process** reconstructs the original data x_0 by learning to undo noise at each step:

$$p_{\theta}(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

Here:

- μ_{θ} : Mean of the predicted data distribution, learned by the neural network.
- Σ_{θ} : Variance, which can be learned or fixed.
- θ : Neural network parameters optimized during training.

The model process iteratively predicts the noise and removes it to recover the original data.

Step 3: Training Objective

The key to training a diffusion model is minimizing the difference between the true reverse process $q(x_{t-1} / x_t)$ and the learned reverse process $p_{\theta}(x_{t-1} / x_t)$. This is achieved by minimizing the *Kullback-Leibler (KL)* divergence between the two distributions.

The loss function is expressed as:

$$L = \mathbb{E}_{x_0, t, \epsilon} [\|\epsilon - \epsilon_{\theta}(x_t, t)\|^2]$$

Where,

- ϵ : True noise added during the forward process.
- ϵ_{θ} : Predicted noise by the neural network.

The model learns to predict the noise at each step, effectively enabling it to reverse the forward process.

Step 4: Sampling

Once trained, the model generates new data by starting with pure noise $x_T \sim N(0, I)$ and applying the reverse process iteratively:

$$\mathbf{x}_{t-1} = \mu_{\theta}(\mathbf{x}_t, t) + \Sigma_{\theta}^{1/2} \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

The model denoises x_t at each step until it produces the reconstructed data x_0 .

We execute the traditional diffusion model on the MNIST dataset using the implementation available at the following repository:

[GitHub: Traditional Diffusion Model](#).

The model generates the following output:

```
Loaded MNIST images: (60000, 28, 28)
Training Diffusion Model...
Diffusion Epoch 1/60, Loss: 0.5296
Diffusion Epoch 2/60, Loss: 0.3034
Diffusion Epoch 3/60, Loss: 0.2428
Diffusion Epoch 4/60, Loss: 0.2070
Diffusion Epoch 5/60, Loss: 0.1834
.....
Diffusion Epoch 9/60, Loss: 0.1440
Diffusion Epoch 10/60, Loss: 0.1379
Diffusion Epoch 11/60, Loss: 0.1354
.....
Diffusion Epoch 55/60, Loss: 0.0927
Diffusion Epoch 56/60, Loss: 0.0920
Diffusion Epoch 57/60, Loss: 0.0919
Diffusion Epoch 58/60, Loss: 0.0913
Diffusion Epoch 59/60, Loss: 0.0910
Diffusion Epoch 60/60, Loss: 0.0906
```



Incorporating Autoregressive Priors into Diffusion Models

A standard diffusion model typically assumes Gaussian noise at the highest step T , from which it gradually denoises. However, if the data distribution is highly structured (e.g., images with spatial correlations or text with sequential dependencies), pure Gaussian noise can be a weak starting point. By contrast, autoregressive models learn a more data-aware prior that respects local or sequential structure.

A Gaussian prior in diffusion is:

Open in app ↗

Medium

Search

Write

7



This assumes no structure beyond zero-mean white noise. If we use an autoregressive model (*ARM*), such as *PixelCNN*, factorizes $p(x)$ into a chain of conditional probabilities:

$$p(\mathbf{x}) = \prod_{i=1}^N p(x_i \mid x_1, \dots, x_{i-1})$$

which naturally captures correlations across space or time. If we jump straight into diffusion from an *AR*-generated sample, we can often converge faster and end up with sharper results, because z_{ARP} (the *AR* prior) already lies closer to the real manifold than pure white noise.

Puzzle Analogy:

Imagine putting together a jigsaw puzzle:

- **Gaussian Noise** → all pieces scattered randomly.
- **AR Prior** → partial edges and corners already assembled, so the puzzle (diffusion) only needs smaller fixes each step.

Autoregressive models sequentially generate data by modeling conditional probabilities. For images, *ARMs* like *PixelCNN* and *PixelSNAIL* model pixels as a sequence:

$$p(I) = \prod_{i=1}^N p(I_i \mid I_1, I_2, \dots, I_{i-1})$$

Algorithm: Autoregressive Priors for Diffusion Initialization

I propose initializing the diffusion process using structured priors generated by an autoregressive model (*ARM*), rather than relying solely on pure Gaussian noise.

Two primary options emerge for applying the *AR* prior.

Method 1: AR Consistency Loss Diffusion

In the first method — named the *AR Consistency Loss Diffusion* — the diffusion model is trained with an augmented loss function that explicitly enforces consistency with a separately trained *AR* prior. Mathematically, given a clean image x_0 and a noise sample $\epsilon \sim N(0, I)$, the forward process produces:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

Here,

$$\bar{\alpha}_t = \prod_{s=1}^t \alpha_s \text{ and } \alpha_s = 1 - \beta_s$$

The model f_θ is trained to predict the added noise via the diffusion loss

$$L_{\text{diff}} = \|f_\theta(x_t) - \epsilon\|^2$$

Then, the model reconstructs an estimate of the original image:

$$x_{0,\text{pred}} = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} f_\theta(x_t)}{\sqrt{\bar{\alpha}_t}}$$

and an AR prior model g_ϕ generates a structured prior $x_{0,\text{AR}}$. The AR consistency loss:

$$L_{\text{AR}} = \|x_{0,\text{pred}} - x_{0,\text{AR}}\|^2$$

is added to the diffusion loss (weighted by a hyperparameter λ_{AR} so that the overall training loss becomes:

$$L = L_{\text{diff}} + \lambda_{AR} L_{AR}$$

This integrated loss mechanism forces the diffusion model not only to denoise but also to generate outputs that closely match the structural cues provided by the *AR* prior, which can result in higher fidelity and more structured reconstructions. However, this comes at the cost of increased training complexity and sensitivity to the choice of λ_{AR} .

The **AR Consistency Loss Diffusion (Method 1)** was executed using the implementation available at the following repository:

[GitHub: AR Consistency Loss Diffusion](#).

The model produced the following results/output:

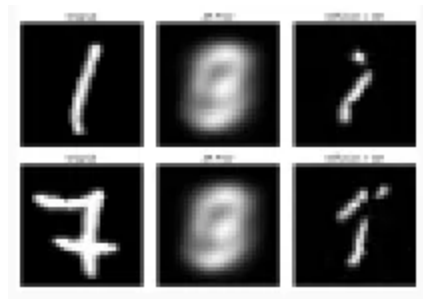
```
Loaded MNIST images: (60000, 28, 28)
Training AR Prior...
AR Prior Epoch 1/10, Loss: 0.3390
AR Prior Epoch 2/10, Loss: 0.2698
AR Prior Epoch 3/10, Loss: 0.2696
AR Prior Epoch 4/10, Loss: 0.2695
.....
AR Prior Epoch 8/10, Loss: 0.2693
AR Prior Epoch 9/10, Loss: 0.2693
AR Prior Epoch 10/10, Loss: 0.2693

Training Diffusion Model with AR consistency loss...
Diffusion Epoch 1/60, Total Loss: 0.6107, Diff Loss: 0.4860, AR Loss: 0.6236
Diffusion Epoch 2/60, Total Loss: 0.3097, Diff Loss: 0.2520, AR Loss: 0.2883
Diffusion Epoch 3/60, Total Loss: 0.2499, Diff Loss: 0.2016, AR Loss: 0.2417
```

```

Diffusion Epoch 4/60, Total Loss: 0.2206, Diff Loss: 0.1762, AR Loss: 0.2219
Diffusion Epoch 5/60, Total Loss: 0.2002, Diff Loss: 0.1584, AR Loss: 0.2092
Diffusion Epoch 6/60, Total Loss: 0.1874, Diff Loss: 0.1471, AR Loss: 0.2012
Diffusion Epoch 7/60, Total Loss: 0.1777, Diff Loss: 0.1387, AR Loss: 0.1952
Diffusion Epoch 8/60, Total Loss: 0.1702, Diff Loss: 0.1319, AR Loss: 0.1913
Diffusion Epoch 9/60, Total Loss: 0.1644, Diff Loss: 0.1267, AR Loss: 0.1884
.....
Diffusion Epoch 55/60, Total Loss: 0.1102, Diff Loss: 0.0750, AR Loss: 0.1757
Diffusion Epoch 56/60, Total Loss: 0.1099, Diff Loss: 0.0748, AR Loss: 0.1755
Diffusion Epoch 57/60, Total Loss: 0.1093, Diff Loss: 0.0743, AR Loss: 0.1750
Diffusion Epoch 58/60, Total Loss: 0.1092, Diff Loss: 0.0742, AR Loss: 0.1749
Diffusion Epoch 59/60, Total Loss: 0.1086, Diff Loss: 0.0736, AR Loss: 0.1753
Diffusion Epoch 60/60, Total Loss: 0.1088, Diff Loss: 0.0737, AR Loss: 0.1754

```



Method 2: AR Prior Blending Diffusion

The second method — termed the *AR Prior Blending Diffusion* — uses a conventional diffusion training procedure where the forward process adds pure Gaussian noise, and the model is trained with the standard noise prediction loss:

$$L_{\text{diff}} = \|f_{\theta}(x_t) - \epsilon\|^2$$

without any additional *AR*-related loss term. Here, the forward process is defined identically as:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

At inference, however, an AR prior model g_ϕ is used to generate a structured sample $x_{\{0,AR\}}$ from a random latent vector z . This AR output is then blended with Gaussian noise using a blending factor α to form an improved initialization for the reverse process:

$$x_T^{\text{init}} = \alpha x_{0,AR} + (1 - \alpha) \eta$$

with $\eta \sim N(0, I)$. The reverse diffusion process then proceeds from this blended initialization to produce the final output.

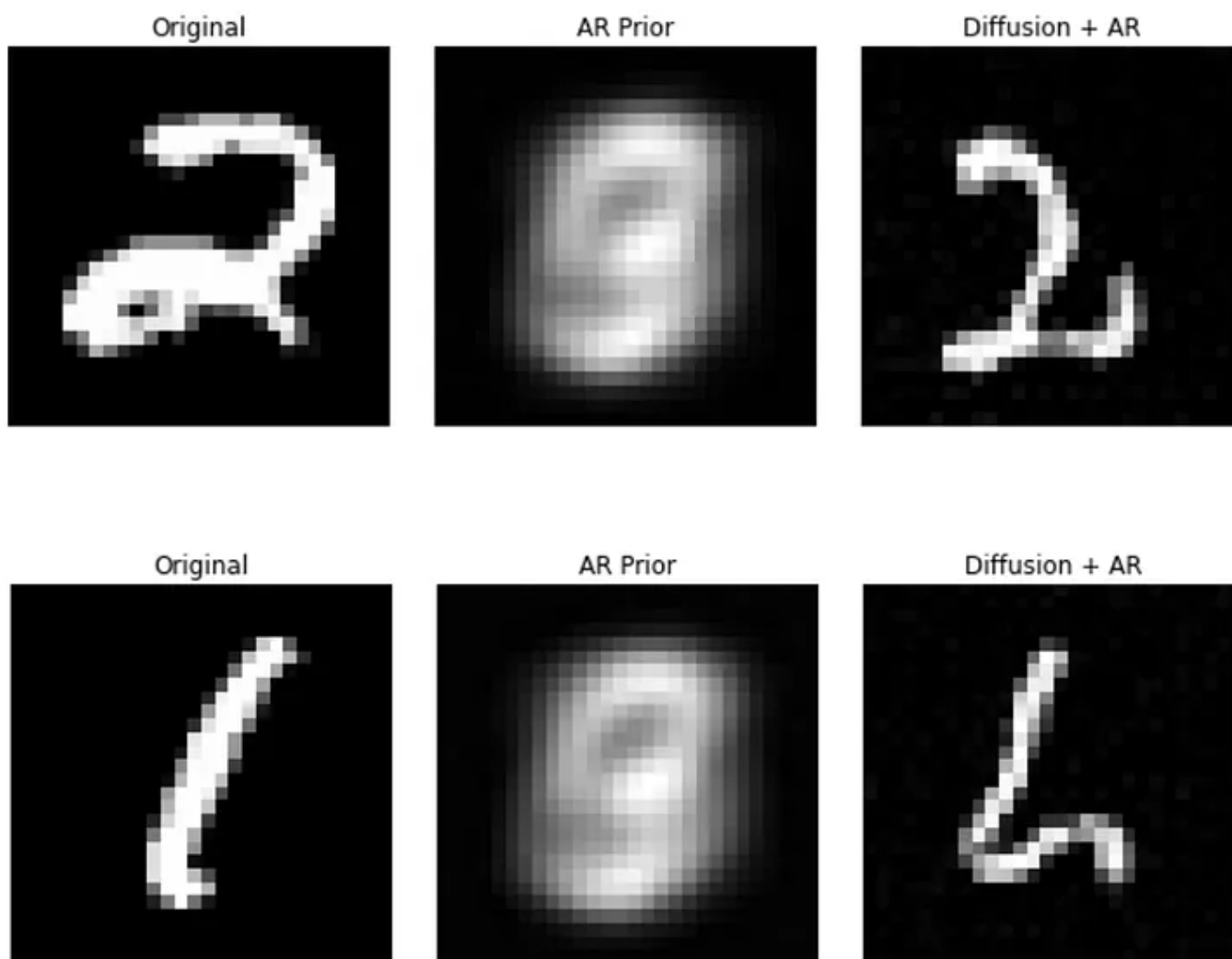
The following output was generated by executing the **AR Prior Initialization Diffusion (Method 2)** using the implementation available at:

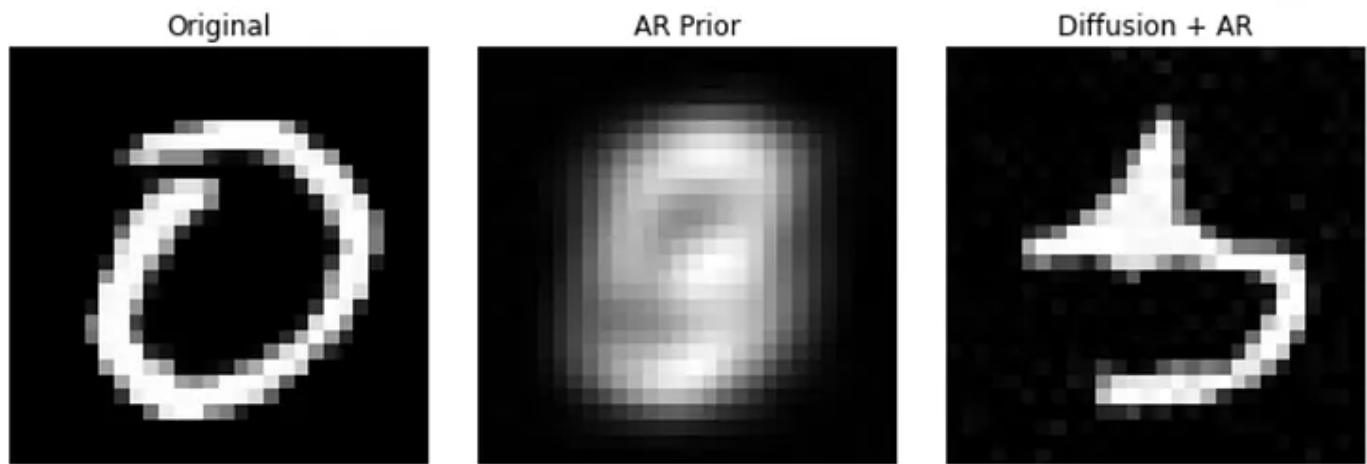
[GitHub: AR Prior Initialization Diffusion.](#)

```
#####
Loaded MNIST images: (60000, 28, 28)
Training AR Prior...
AR Prior Epoch 1/10, Loss: 0.3384
AR Prior Epoch 2/10, Loss: 0.2698
AR Prior Epoch 3/10, Loss: 0.2696
AR Prior Epoch 4/10, Loss: 0.2695
.....
AR Prior Epoch 8/10, Loss: 0.2693
AR Prior Epoch 9/10, Loss: 0.2693
AR Prior Epoch 10/10, Loss: 0.2693

Training Diffusion Model...
Diffusion Epoch 1/60, Loss: 0.4971
Diffusion Epoch 2/60, Loss: 0.2690
```

```
Diffusion Epoch 3/60, Loss: 0.2173
Diffusion Epoch 4/60, Loss: 0.1880
Diffusion Epoch 5/60, Loss: 0.1682
Diffusion Epoch 6/60, Loss: 0.1520
Diffusion Epoch 7/60, Loss: 0.1392
Diffusion Epoch 8/60, Loss: 0.1293
.....
Diffusion Epoch 55/60, Loss: 0.0694
Diffusion Epoch 56/60, Loss: 0.0698
Diffusion Epoch 57/60, Loss: 0.0690
Diffusion Epoch 58/60, Loss: 0.0689
Diffusion Epoch 59/60, Loss: 0.0682
Diffusion Epoch 60/60, Loss: 0.0682
```





These two methods illustrate different approaches to embedding *AR* priors into diffusion models: one by directly incorporating an *AR* consistency term during training, and the other by using the *AR* output to enhance the initial state at inference.

Comparison of Three Diffusion Methods

We tested three diffusion-based generative models on MNIST: Traditional Diffusion (no *AR*), *AR* Consistency Loss Diffusion (Method 1), and *AR* Prior Initialization Diffusion (Method 2).

Traditional Diffusion (No *AR*) starts from pure Gaussian noise and gradually refines images. It achieved a final loss of 0.0906 , with clear but slightly noisy digits. Since it learns structure from scratch, more training is required.

***AR* Consistency Loss Diffusion (Method 1)** adds an extra loss to align generated samples with an *AR* model's output. This method stabilizes training but converges to a slightly higher loss (0.1088). While *AR* supervision improves structure, it may introduce blurring and over-smooth digits.

AR Prior Initialization Diffusion (Method 2) starts diffusion from an *AR*-generated prior instead of pure noise. This led to the lowest loss (0.0682) and sharpest digit reconstructions. By using a structured prior, diffusion refines details rather than reconstructing from scratch, making training more efficient.

Best Approach?

AR Prior Initialization (Method 2) performs best in loss reduction, training speed, and image sharpness. Traditional Diffusion struggles with structure, and Method 1's consistency loss improves training but limits fine details. Future work could mix AR priors with controlled noise to balance structure and randomness for even better results.

Conclusion and Future Work

This paper explored how Autoregressive Priors (*ARPs*) can improve diffusion models, making them faster and improving sample quality. By starting with structured noise rather than pure Gaussian randomness, *AR*-based priors reduce the burden on diffusion models, leading to better results with fewer training steps.

Beyond diffusion, AR methods could improve *VAEs* and *GANs*. In *VAEs*, AR priors can structure latent space, reducing issues like posterior collapse and improving reconstructions. In *GANs*, AR-based noise can stabilize training and prevent mode collapse, leading to more diverse and realistic samples.

For structured data like financial trends, weather models, or medical imaging, AR priors can guide generative models to produce more accurate and interpretable sequences. Instead of purely noise-driven sampling,

combining *AR* sequence prediction with diffusion denoising could lead to a balance of structure and flexibility.

Looking ahead, hybrid models mixing *AR*, diffusion, and adversarial training could push generative AI further. *AR*-based initialization makes models more data-aware, stable, and efficient, opening doors for practical applications in finance, healthcare, and beyond.

About me

With over 20 years of experience in software and database management and 25 years teaching IT, math, and statistics, I am a Data Scientist with extensive expertise across multiple industries.

You can connect with me at:

Email: datalev@gmail.com | [LinkedIn](#) | [X/Twitter](#)

[Diffusion Models](#)[Autoregressive](#)[AI](#)[Generative Ai Tools](#)[Python](#)

Published in Towards AI

78K Followers · Last published just now

[Follow](#)

The leading AI community and content platform focused on making AI accessible to all. Check out our new course platform:

<https://academy.towardsai.net/courses/beginner-to-advanced-llm-dev>



Written by Shenggang Li

2.2K Followers · 77 Following

Following

No responses yet



Alex Mylnikov

What are your thoughts?

More from Shenggang Li and Towards AI



In Towards AI by Shenggang Li

Reinforcement Learning for Business Optimization: A Genetic...

Applying PPO and Genetic Algorithms to Dynamic Pricing in Competitive Markets



In Towards AI by Gao Dalie (高達烈)

Manus AI + Ollama: Build & Scrape ANYTHING (First-Ever General AI...

Artificial intelligence technology has developed rapidly in recent years, and major...

★

Mar 17

👤

171

💬

3

🔖

+

...

★

Mar 11

👤

2K

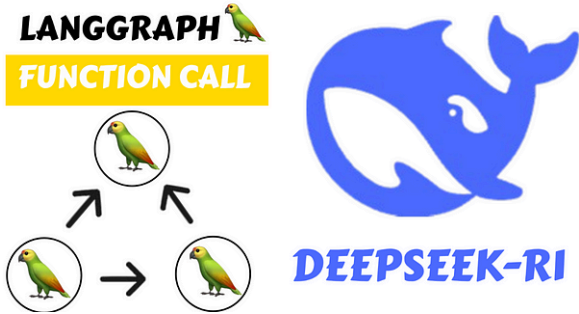
💬

19

🔖

+

...



Towards AI

In Towards AI by Gao Dalie (高達烈)

LangGraph + DeepSeek-R1 + Function Call + Agentic RAG...

In this video, I have a super quick tutorial showing you how to create a multi-agent...

★

Feb 23

👤

1K

💬

9

🔖

+

...



Towards AI

In Towards AI by Shenggang Li

Practical Guide to Distilling Large Models into Small Models: A Nove...

Comparing Traditional and Enhanced Step-by-Step Distillation: Adaptive Learning,...

★

Mar 3

👤

222

💬

2

🔖

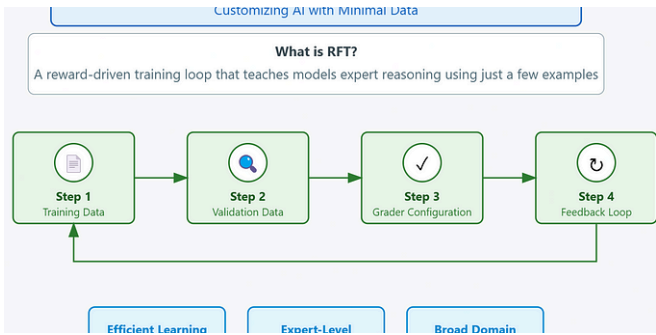
+

...

See all from Shenggang Li

See all from Towards AI

Recommended from Medium

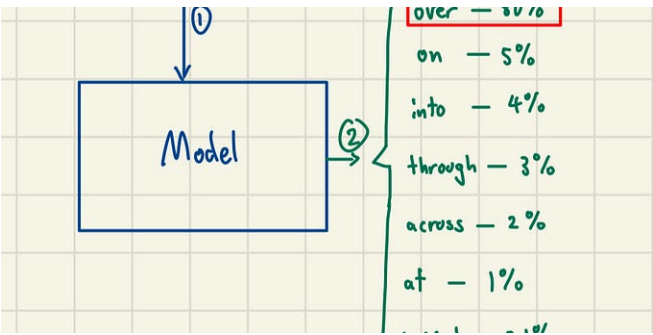


In Towards AI by Louis-François Bouchard

Reinforcement Fine-Tuning (RFT)

Teaching AI Without Huge Datasets

★ Mar 18 🖱 95 💬 2 📌 ⋮

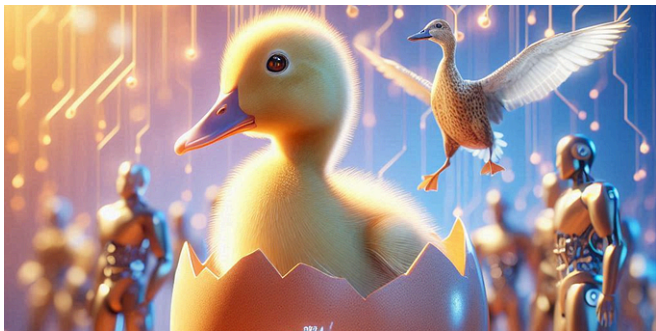


Samuel Zhu

Demystifying the Transformer Model

Note: this blog post is a final paper for my UCSB WRIT105SW course. As such, it is a...

Mar 18 🖱 153 💬 8 📌 ⋮

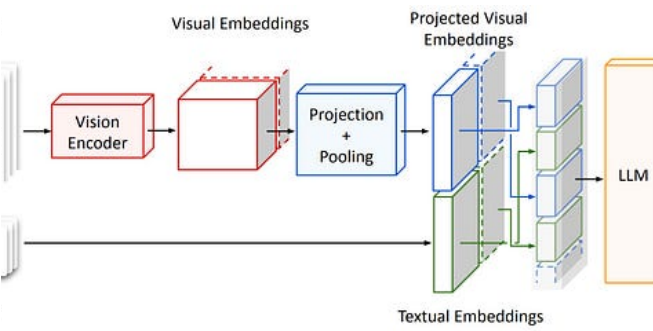


In Data Science Collective by Buse Şenol

SmolDocling: A New Era in Document Processing — OCR

A model that outperforms its competitors 27 times its size with the DocTags format

★ 3d ago 🖱 400 💬 2 📌 ⋮

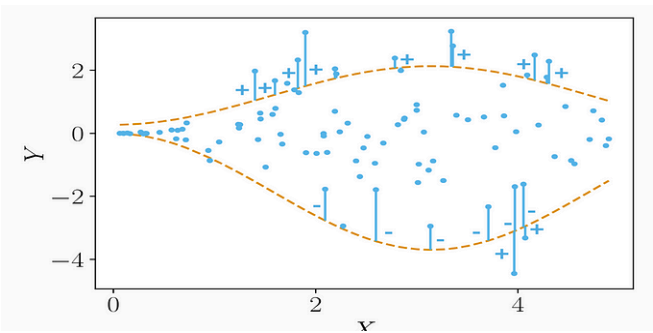
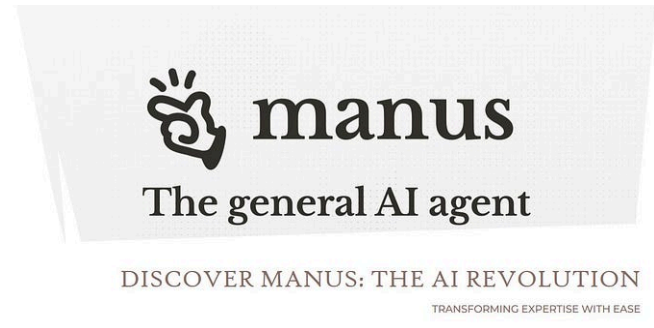


In GoPenAI by Jenray

SmolDocling: Revolutionizing AI Document Conversion with Tiny...

SmolDocling appears to be a cutting-edge AI model focused on converting document...

★ Mar 18 🖱 53 💬 2 📌 ⋮





 In Generative AI by Anwesh Agrawal

Manus: The AI That’s Quietly Making Experts Sweat (And Why...

From cloning bacteria in a virtual petri dish to designing your dream bedroom—this AI...

★ Mar 16 🖱 353 💬 6 📌 ⋮

 Valeriy Manokhin, PhD, MBA, CQF 

Conformalized Quantile Regression: Smarter Uncertainty...

Quantifying uncertainty in predictions is crucial for informed decision-making....

6d ago 🖱 179 📌 ⋮

See more recommendations