Comparative Study of Generative Models on CIFAR-10

Archil Zhghenit, Nika Matcharadze

Free Uni of Tbilisi

January 31, 2026

CIFAR-10

- 60,000 color images (32×32)
- 10 classes (airplane, automobile, bird, cat, etc.)
- 50k training / 10k test images
- Images normalized to $[-1, 1]$ (or $[0, 1]$)

ConvVAE

- Convolutional Resnet Style encoder–decoder architecture
- Gaussian latent space with KL divergence regularization
- Optimizes ELBO objective

ConvVAE (β-VAE Variant)

- Convolutional encoder–decoder with Gaussian latent variables
- Objective:

$$\mathcal{L} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \beta \mathrm{KL}(q(z|x) \| p(z))$$

- β controls the trade-off between reconstruction fidelity and latent regularization

Effect of β in ConvVAE

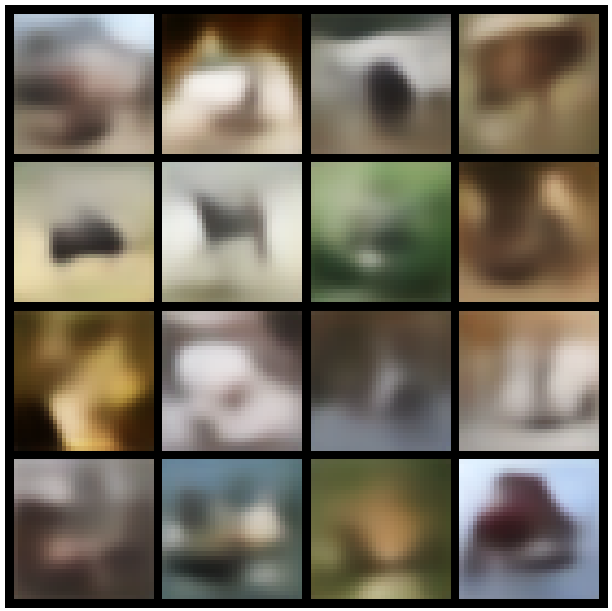
β	FID ↓	IS ↑	Mutual Information
0.1	129	3.66	5.54
1	160	2.00	5.53
10	313	1.70	0.00

Analysis: ConvVAE with Different β

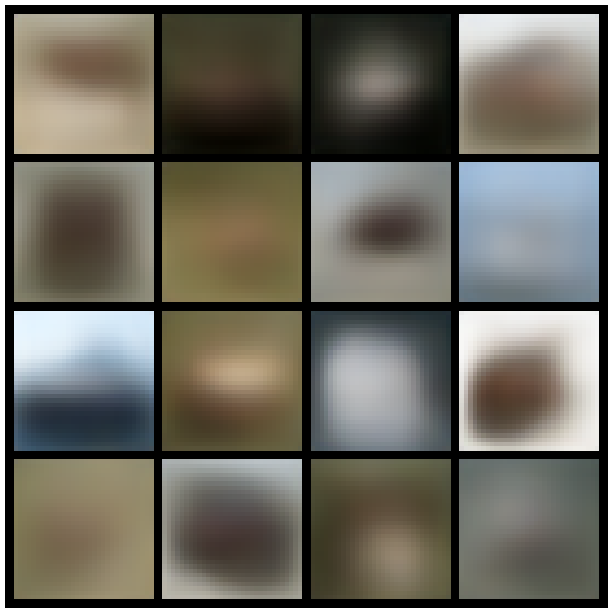
- $\beta = 0.1$: Best generative performance (lowest FID, highest IS)
- $\beta = 1$: Stronger regularization with moderate quality degradation
- $\beta = 10$: Posterior collapse observed (MI ≈ 0)
- High β harms sample quality on CIFAR-10 despite stronger constraints

Generated Samples (ConvVAE)

$\beta = 0.1$

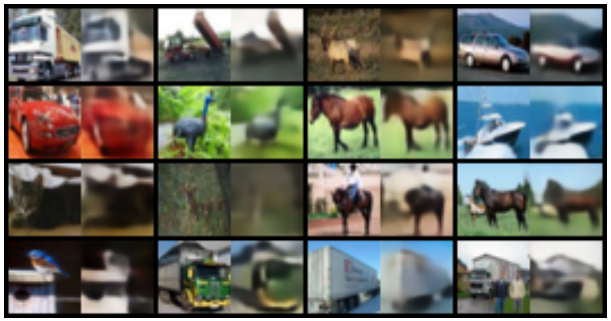


$\beta = 1$

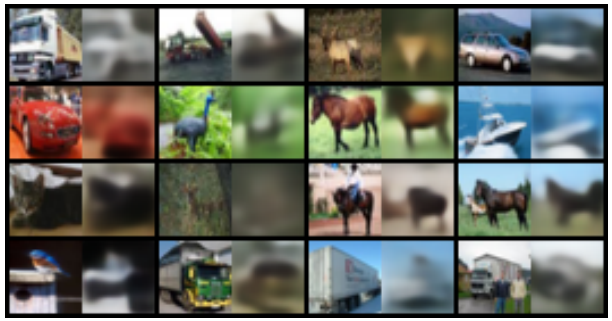


$\beta = 10$

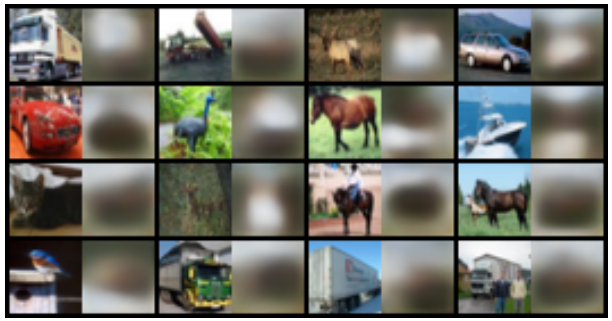
Reconstruction Samples (ConvVAE)



$\beta = 0.1$



$\beta = 1$



$\beta = 10$

MMD-VAE

- Replaces KL divergence with Maximum Mean Discrepancy (MMD)
- Encourages richer latent representations
- Better sample diversity than standard VAE

Evolution of the VAE Loss

- **Vanilla VAE Loss (ELBO)**

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \mathrm{KL}(q(z|x) \,\|\, p(z))$$

  - KL term enforces *local* prior matching for every input
  - Often leads to posterior collapse

- **MMD-VAE Loss**

$$\mathcal{L}_{\text{MMD-VAE}} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - \lambda\, \mathrm{MMD}(q(z), p(z))$$

  - MMD enforces *global* matching of the aggregated posterior
  - Encourages higher mutual information and better latent usage

Why MMD-VAE? Advantages over Vanilla VAE

- **Avoids Posterior Collapse**
  - Vanilla VAE:
$$q(z \mid x) \to \mathcal{N}(0, I), \quad \forall x$$
  - MMD-VAE:
$$\mathbb{E}_x \left[q(z \mid x) \right] \approx \mathcal{N}(0, I)$$
  - Prior matching is enforced *globally* (in expectation), not per-sample
  - Encourages higher mutual information between x and z
- **Better Balance Between Reconstruction and Regularization**
  - In Vanilla VAE, KL term often becomes negligible compared to reconstruction loss
  - MMD penalty remains influential throughout training
  - Leads to more meaningful latent representations

Mutual Information: ConvVAE vs MMD-VAE

- Experiment conducted using a **simple (non-complex) encoder–decoder architecture**
- Mutual Information (MI) measures dependency between input x and latent code z

Model	Mutual Information ↑
ConvVAE	5.41
MMD-VAE	5.54

MMD-VAE preserves slightly higher mutual information, indicating stronger latent utilization even with a simple architecture.

MMD-VAE: Overall Performance on CIFAR-10

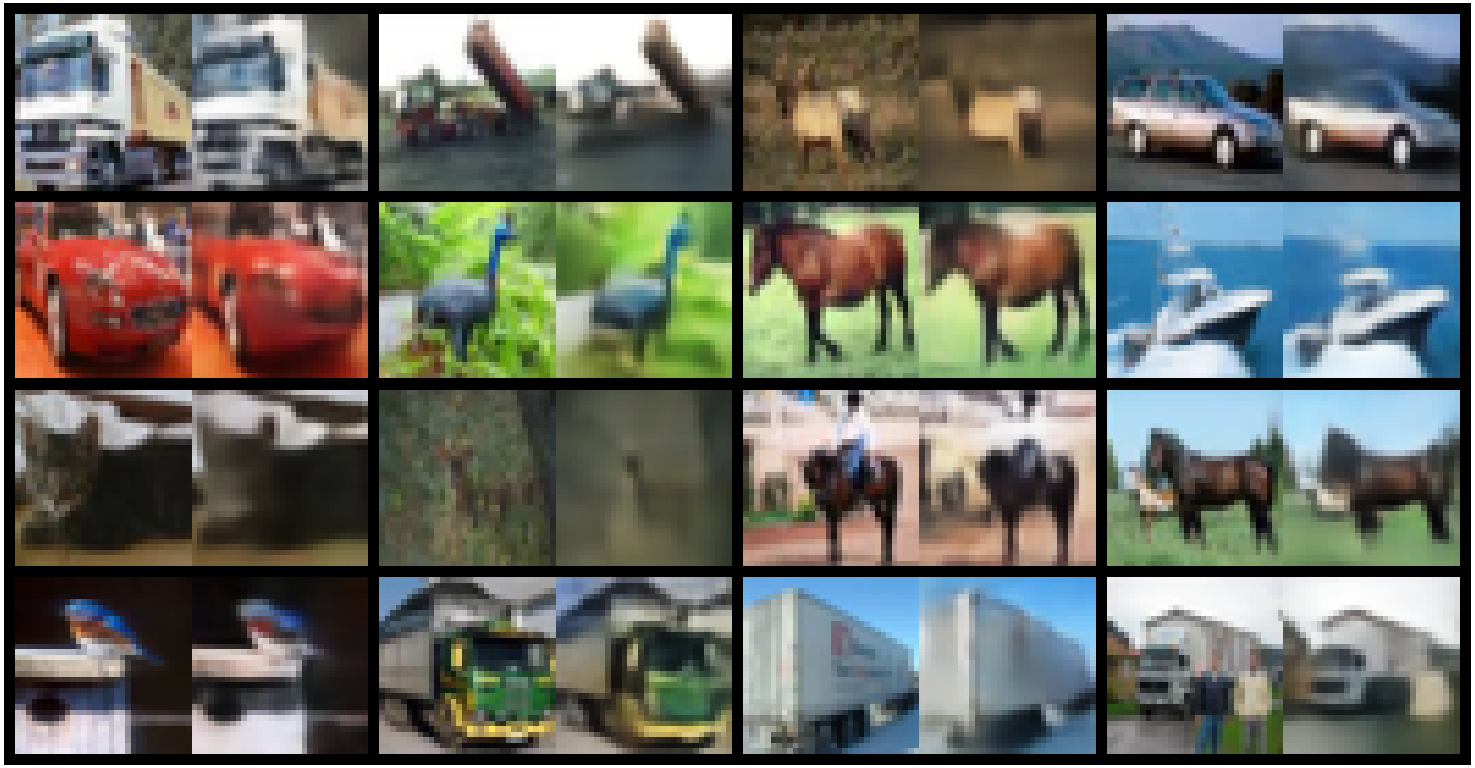
- Evaluated using sample quality, diversity, and latent utilization metrics

Metric	Value
FID ↓	116
IS ↑	4.105
Mutual Information ↑	5.54

MMD-VAE achieves strong generative performance while maintaining high latent information utilization.

Generated Samples (MMD Vae)

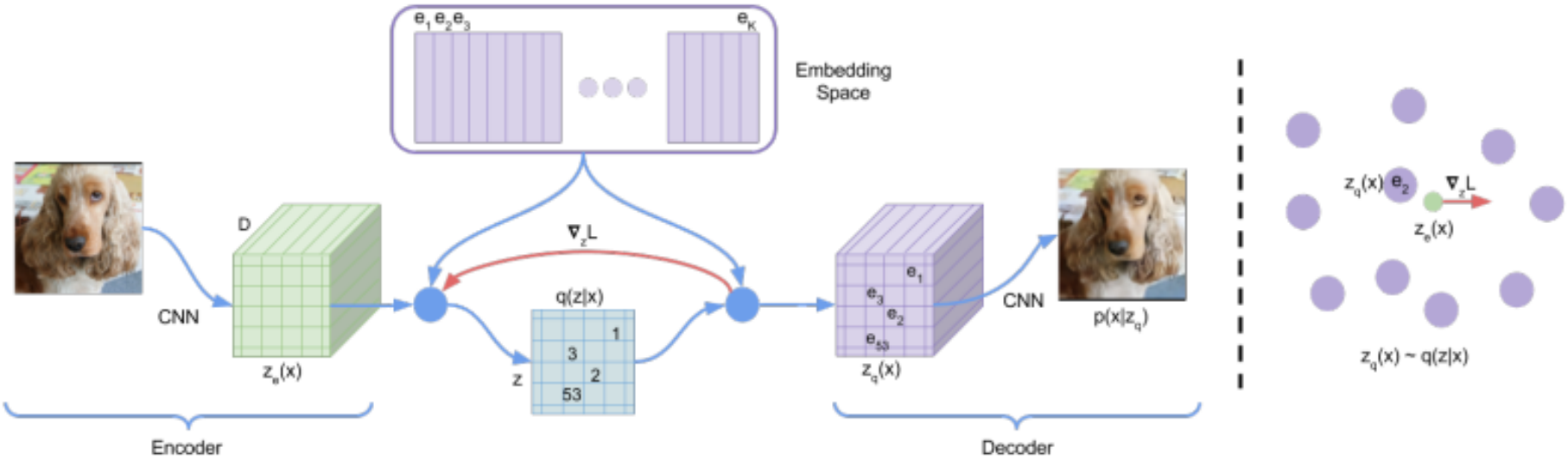
Reconstruction Samples (MMD-VAE)



VQ-VAE

- Discrete latent space with vector quantization
- Learns a codebook of embeddings
- Effective for high-quality reconstructions

VQ-VAE Architecture



VQ-VAE Loss Function

- VQ-VAE uses a **discrete latent space** with vector quantization

$$\mathcal{L}_{\text{VQ-VAE}} = \underbrace{\|x - \hat{x}\|^2}_{\text{Reconstruction}} + \underbrace{\|\text{sg}[z_e(x)] - e_k\|^2}_{\text{Codebook Loss}} + \underbrace{\beta\|z_e(x) - \text{sg}[e_k]\|^2}_{\text{Commitment Loss}}$$

- $z_e(x)$: encoder output
- e_k: nearest codebook embedding
- $\text{sg}[\cdot]$: stop-gradient operator

Why VQ-VAE? Key Advantages

- **Sharper Reconstructions**
  - Vanilla VAE samples multiple latents:

$$z_1, z_2 \sim q(z \mid x)$$

  - Decoder must map different z values to the same x
  - Averaging effect leads to blurry reconstructions
  - VQ-VAE maps x to a single discrete code e_k
  - No averaging $\Rightarrow$ sharper images

- **No KL Term & No Posterior Collapse**
  - No prior matching via KL divergence
  - Discrete bottleneck enforces latent usage
  - Encoder cannot ignore the latent representation

VQ-VAE: Generative Process

- Encoder maps input image to discrete latent codes:

$$x \xrightarrow{\text{Encoder}} z_e(x) \xrightarrow{\text{VQ}} e_k$$

- A **secondary autoregressive model** (e.g., PixelCNN) is trained on the discrete latents:

$$p(e) = \prod_i p(e_i \mid e_{<i})$$

- Sampling pipeline:

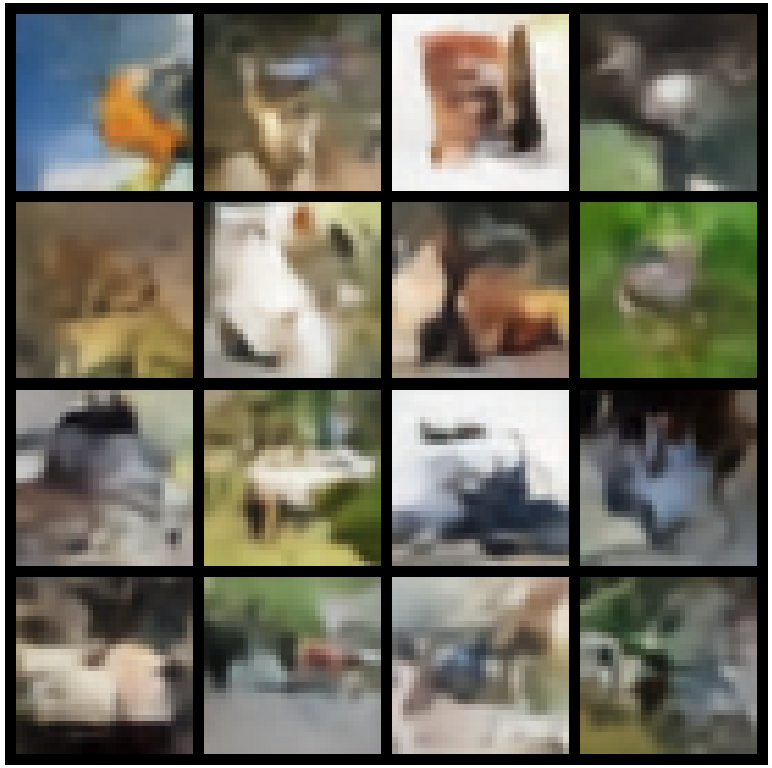
$$e \sim \text{PixelCNN} \;\rightarrow\; \hat{x} = \text{Decoder}(e)$$

- **Disadvantage: PixelCNN Bottleneck**
  - Requires training an additional complex model
  - Autoregressive sampling is **slow and sequential**
  - Sampling speed limited by PixelCNN, not the decoder

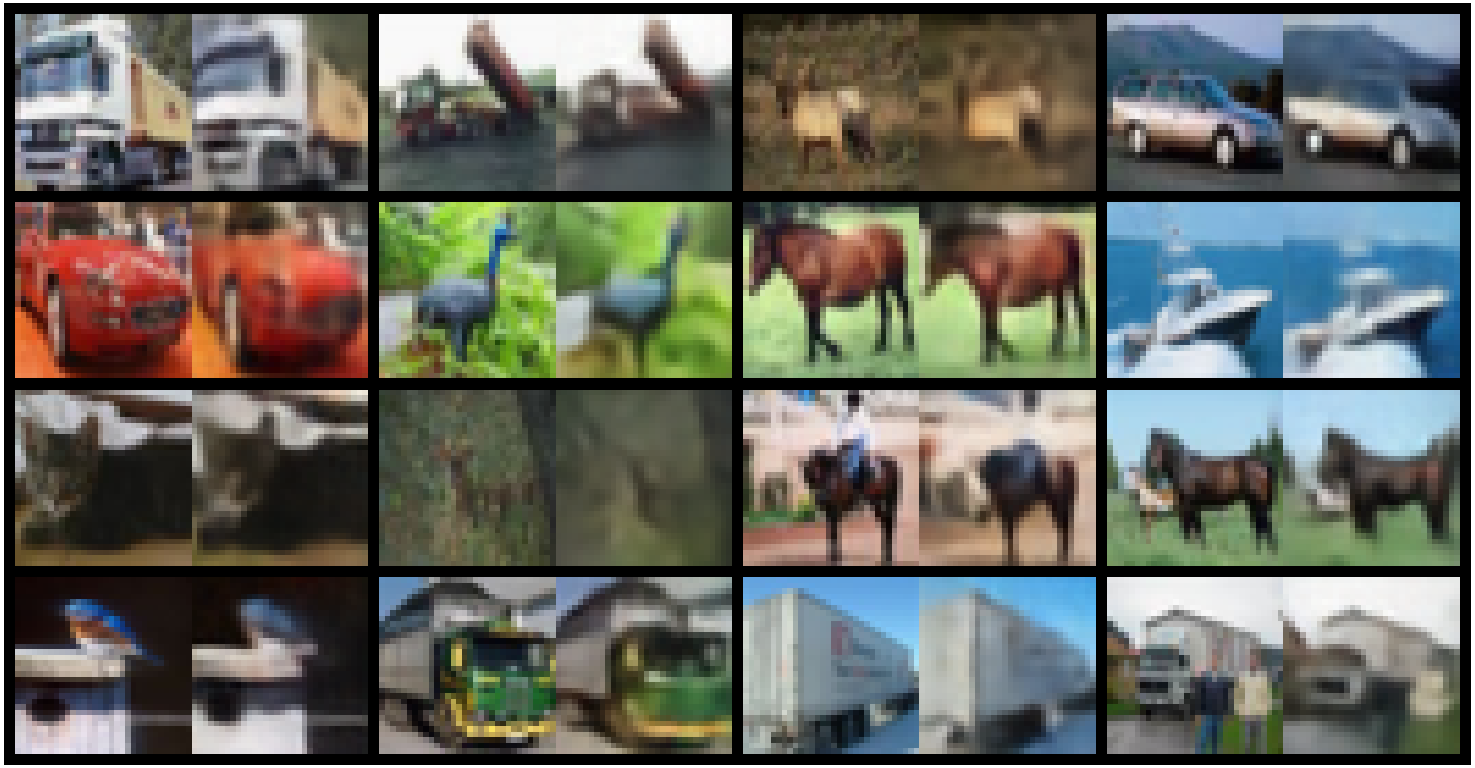
VQ-VAE: Quantitative Results on CIFAR-10

Metric	Value
FID $\downarrow$	84.45
IS $\uparrow$	4.448
Mutual Information $\uparrow$	5.36
Reconstruction Loss $\downarrow$	10.72

Generated Samples (VQ-VAE)



Reconstruction Samples (VQ-VAE)



Summary: VAE-Based Models on CIFAR-10

Model	FID ↓	IS ↑	MI ↑	Recon. Loss ↓
ConvVAE ($\beta = 0.1$)	129	3.66	5.53	31.66
MMD-VAE	116	4.105	5.54	7.54
VQ-VAE	84.45	4.448	5.36	10.72

Progressive improvement in sample quality is observed from ConvVAE to VQ-VAE.

NCSN

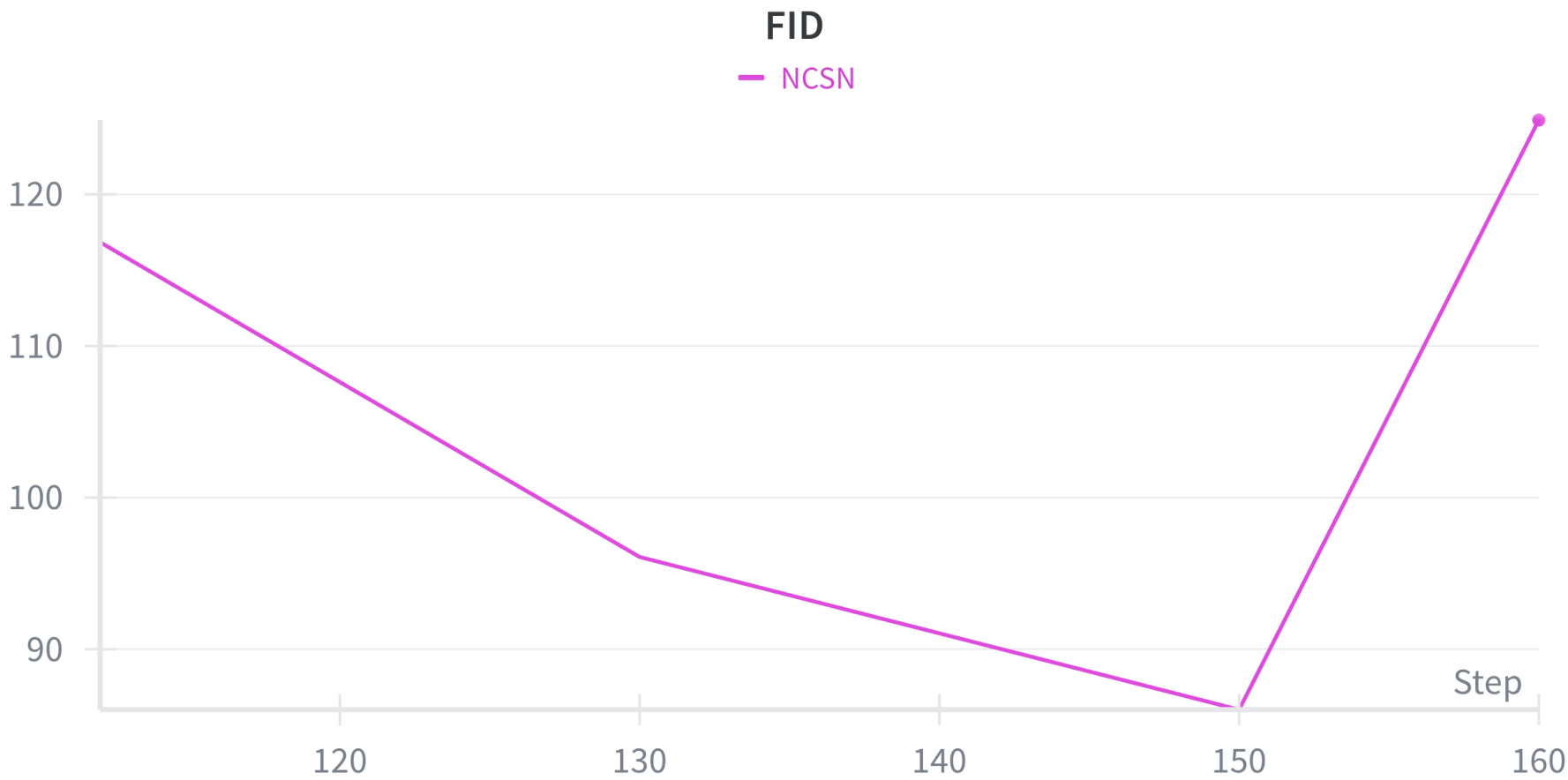
- Noise Conditional Score Network
- Learns gradients of the data distribution
- Sampling via Langevin dynamics
- Unet-type network

NCSN: Quantitative Results on CIFAR-10

Metric	Value
FID $\downarrow$	86.43
IS $\uparrow$	4.12

NCSN achieves competitive sample quality without an explicit latent representation.

NCSN: FID Over Training Epochs



DDPM Architecture and Training Setup

- **Backbone: U-Net**
  - Residual blocks with Group Normalization
  - Self-attention at 16×16 resolution
  - Sinusoidal position embeddings for timestep encoding
  - Approximately **37M parameters** (CIFAR-10 configuration)
- **Diffusion Process (Following the DDPM Paper)**
  - Number of timesteps: $T = 1000$
  - Linear noise schedule:

$$\beta_1 = 10^{-4} \rightarrow \beta_T = 0.02$$

  - Dropout rate: 0.1
  - Exponential Moving Average (EMA) of parameters with decay 0.9999

DDPM Training Algorithm

1. Sample a clean image: $x_0 \sim p_{\text{data}}(x)$
2. Sample timestep: $t \sim \mathcal{U}\{1, \ldots, T\}$
3. Sample noise: $\varepsilon \sim \mathcal{N}(0, I)$
4. Corrupt the image:

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

5. Predict noise: $\varepsilon_\theta(x_t, t)$
6. Compute loss:

$$\mathcal{L} = \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2$$

7. Update model parameters θ via gradient descent

DDPM Sampling Algorithm (Reverse Diffusion)

1. Initialize with pure noise:

$$x_T \sim \mathcal{N}(0, I)$$

2. For $t = T, \ldots, 1$:
   - Predict noise:

$$\varepsilon_\theta(x_t, t)$$

   - Compute the mean:

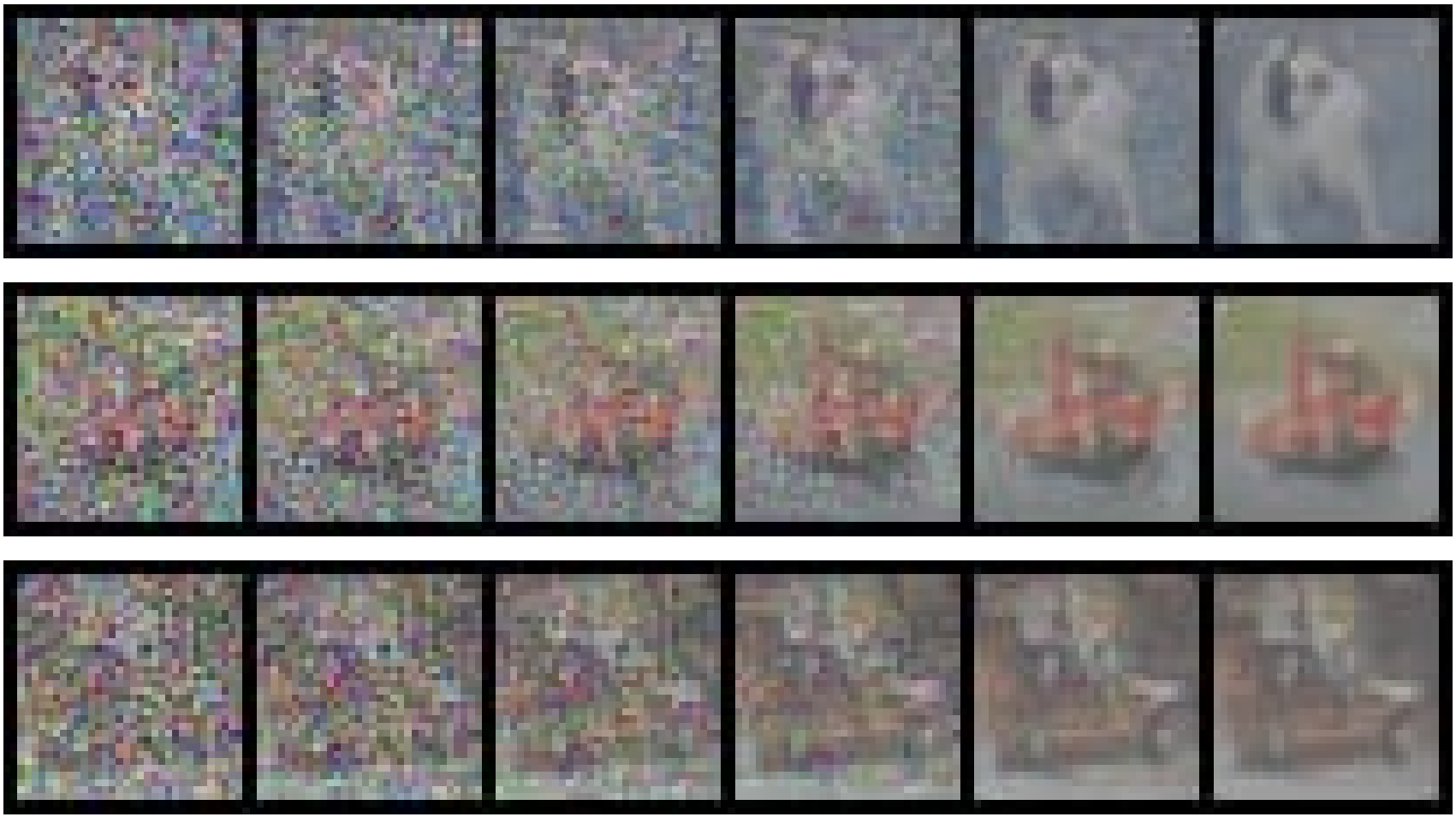
$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right)$$

   - Sample:

$$x_{t-1} = \mu_\theta + \sqrt{\tilde{\beta}_t}\, z, \quad z \sim \begin{cases} \mathcal{N}(0, I), & t > 1 \\ 0, & t = 1 \end{cases}$$

3. Return generated sample: x_0

DDPM Denoising Process



DDPM: Model Components

- **Noise Schedule**
  - `get_beta_schedule()`: constructs β_t (linear or cosine)
- **Timestep Encoding**
  - `SinusoidalPositionEmbeddings`
  - Encodes diffusion timestep t (Transformer-style)
- **Residual Block**
  - Group Normalization
  - Time embedding injection
  - Dropout
- **AttentionBlock**
  - Multi-head self-attention
- **U-Net Architecture**
  - Downsampling path with attention layers
  - Middle blocks at lowest resolution
  - Upsampling path with skip connections

DDPM: Main API

- **DDPM**
  - Main diffusion model wrapper
- `q_sample()`
  - Forward diffusion process
  - Adds noise to clean images:

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

- `p_losses()`
  - Computes training objective
  - MSE between true and predicted noise
- `p_sample()`
  - One-step reverse diffusion (denoising)
  - Computes x_{t-1} from x_t
- `sample()`
  - Full generative procedure
  - Iteratively denoises from $x_T \rightarrow x_0$

Generated Samples (DDPM)

Conclusion

- Compared four generative paradigms on CIFAR-10
- Trade-off between sample quality, diversity, and training cost
- Future work: hybrid models, larger datasets

Thank You

Questions?