BADiff: Bandwidth Adaptive Diffusion Model

Xi Zhang[1] **Hanwei Zhu**[1]✉ **Yan Zhong**[1] **Jiamang Wang**[2] **Weisi Lin**[1]✉
[1]Nanyang Technological University [2]Alibaba Group
{xi.zhang, hanwei.zhu, wslin}@ntu.edu.sg

Abstract

In this work, we propose a novel framework to enable diffusion models to adapt their generation quality based on real-time network bandwidth constraints. Traditional diffusion models produce high-fidelity images by performing a fixed number of denoising steps, regardless of downstream transmission limitations. However, in practical cloud-to-device scenarios, limited bandwidth often necessitates heavy compression, leading to loss of fine textures and wasted computation. To address this, we introduce a joint end-to-end training strategy where the diffusion model is conditioned on a target quality level derived from the available bandwidth. During training, the model learns to adaptively modulate the denoising process, enabling early-stop sampling that maintains perceptual quality appropriate to the target transmission condition. Our method requires minimal architectural changes and leverages a lightweight quality embedding to guide the denoising trajectory. Experimental results demonstrate that our approach significantly improves the visual fidelity of bandwidth-adapted generations compared to naive early-stopping, offering a promising solution for efficient image delivery in bandwidth-constrained environments. Code is available at: https://github.com/xzhang9308/BADiff.

1 Introduction

Diffusion models [13, 43, 15, 37, 40] have recently demonstrated remarkable capabilities in synthesizing high-quality images, significantly surpassing previous generative approaches such as GANs [10, 36, 2, 19] and VAEs [21, 38, 12, 47]. Despite their impressive fidelity, deploying diffusion models in realistic cloud-to-user applications introduces a fundamental bottleneck: transmission bandwidth. In conventional scenarios, generated images undergo aggressive lossy compression [45, 46, 4, 44, 1, 5, 34, 32, 58, 55] before transmission to accommodate limited bandwidth. This cascaded approach—high-quality image generation followed by subsequent compression—not only incurs redundant computational overhead but also significantly degrades perceptual quality, as the compression process often erases the intricate textures and fine details [24, 8, 56, 57, 7, 25, 16, 48] carefully constructed by the diffusion model.

This motivates a critical question: *Can we directly integrate bandwidth-awareness into the diffusion generation process, avoiding the inefficiency and perceptual quality loss associated with post-generation compression?* Diffusion models inherently provide a natural mechanism for addressing this challenge. During sampling, these models progressively refine coarse structures into detailed, realistic textures through iterative denoising steps. Thus, intuitively, terminating the diffusion process early results in simpler, lower-detail images suitable for constrained bandwidth scenarios. However, naively reducing the number of diffusion steps typically produces suboptimal visual quality, as models trained for complete denoising trajectories are not optimized for early termination, leading to visual artifacts and poor perceptual coherence.

✉ Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

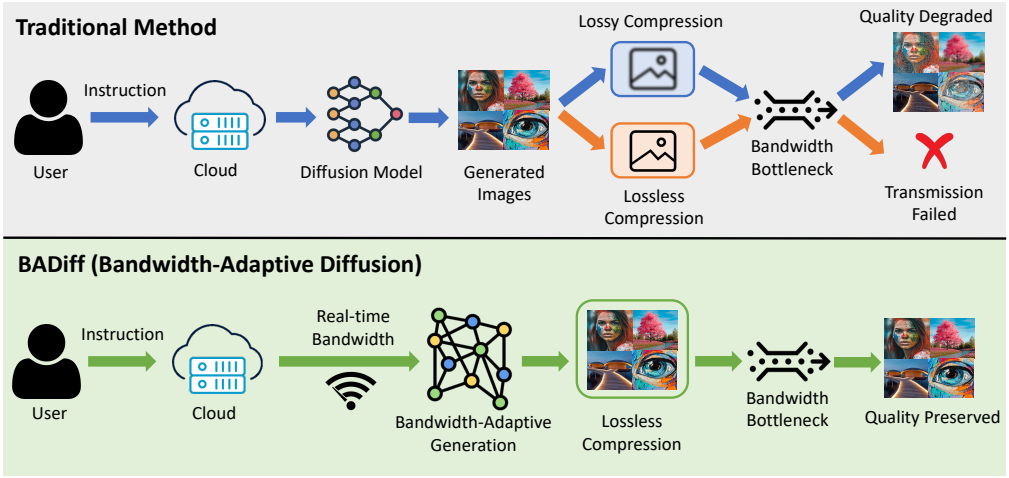

Figure 1: Comparison of traditional diffusion + compression pipeline (top) and the proposed BADiff framework (bottom). BADiff directly generates entropy-constrained images suitable for bandwidth-limited transmission, avoiding quality degradation in post-generation compression.

To effectively address this issue, we propose the ***Bandwidth-Adaptive Diffusion Model (BADiff)***, a novel diffusion model explicitly conditioned on target bandwidth constraints, formulated as entropy targets. By embedding target entropy as an explicit conditioning input, our approach enables the diffusion model to adaptively modulate its denoising behavior. During training, the model is exposed to variable entropy constraints, allowing it to produce perceptually pleasing images even under reduced-step sampling. Furthermore, an entropy regularization loss ensures the generated images adhere closely to the bandwidth constraints, obviating the need for aggressive post-hoc compression. As illustrated in Figure 1, the conventional diffusion pipeline relies on lossy compression to meet bandwidth constraints, often degrading visual quality, whereas our proposed BADiff framework directly generates entropy-constrained outputs suitable for transmission without compromising perceptual fidelity.

Our proposed BADiff framework provides several advantages over existing cascaded and naive early-stopping approaches. First, it significantly reduces computational overhead by adaptively terminating sampling, ensuring efficient inference. Second, by directly generating images meeting bandwidth constraints, BADiff avoids compression-induced artifacts, thus preserving perceptual quality. Third, our framework provides fine-grained, dynamic control over image quality based on real-time bandwidth conditions, enhancing deployment flexibility in practical cloud-to-user applications.

We validate the effectiveness of BADiff through extensive experiments comparing our method to strong baselines, including standard diffusion models with post-generation compression and naive early-stopping approaches. Our results demonstrate that BADiff consistently achieves superior trade-offs among perceptual quality, computational efficiency, and bandwidth efficiency, underscoring the advantage of integrating bandwidth-awareness directly into the generative modeling process.

Our main contributions can be summarized as follows:

- We introduce BADiff, the first bandwidth-adaptive diffusion model explicitly conditioned on target entropy constraints, directly addressing image synthesis for bandwidth-constrained transmission.
- We propose an entropy conditioning mechanism integrated into diffusion models, coupled with an entropy regularization loss, allowing adaptive and efficient generation under diverse bandwidth constraints.
- We develop an adaptive sampling policy that dynamically determines optimal sampling termination, significantly reducing computational cost while preserving image quality.
- Through extensive evaluations, we demonstrate BADiff's superior performance in perceptual quality, computational efficiency, and adherence to bandwidth constraints compared to conventional cascaded diffusion + compression pipelines.

2 Related Work

2.1 Diffusion Models

Diffusion models have recently gained significant attention due to their remarkable ability to generate high-quality images, surpassing traditional generative models such as GANs [10] and VAEs [21]. The foundational work of Ho et al. [13] introduced Denoising Diffusion Probabilistic Models (DDPMs), formalizing diffusion models as a parameterized Markov chain trained by variational inference to invert a gradual noising process. Song et al. [43] further generalized the framework through Score-based Generative Models (SGMs), which unify diffusion models and score-matching approaches under a continuous-time stochastic differential equation (SDE) framework. Recent works have extended diffusion models to various tasks beyond image synthesis, including video generation [15], text-to-image generation [37, 40], and 3D synthesis [35].

2.2 Accelerated Sampling of Diffusion Models

Despite their high-quality outputs, diffusion models are computationally expensive due to the iterative sampling procedure required during generation. To mitigate this issue, substantial efforts have been made toward accelerating diffusion model sampling. DDIM [42] introduced deterministic sampling methods enabling fewer inference steps, significantly reducing computation. Further advances, including DPM-Solver [29] and FastDPM [22], have employed numerical methods inspired by ordinary differential equations (ODEs) to shorten sampling times substantially while preserving generation quality. PNDM [28] introduces a pseudo–numerical solver that treats the reverse diffusion ODE with high-order Runge–Kutta–style updates, enabling high-fidelity image generation in as few as four forward passes. Knowledge distillation based methods [41, 31] reduce sampling steps by distilling knowledge from slower teacher models into faster student models. Alternative approaches involve adaptive step size selection [17] or latent space compression [39] to reduce computational overhead. AutoDiffusion [26] further accelerates sampling via non-uniform step skipping.

Unlike prior methods that optimize scheduling or model architecture, DDSM [49] introduces dynamic U-Net pruning to minimize redundant computations at each step. This approach is complementary to existing acceleration techniques. Related works include OMS-DPM [27], which optimizes model scheduling via predictor-based algorithms, and Spectral Diffusion [50], which employs dynamic gating for efficiency. While eDiff-I [3] uses multiple fixed-size expert models, StepSaver [52] proposes a step predictor to determine the minimal denoising steps required for high-quality generation, further improving efficiency. Moreover, adaptive methods such as AdaDiff [53] dynamically adjust inference trajectories, further enhancing efficiency by selectively allocating computational resources during sampling based on intermediate outputs.

2.3 Conditional and Controllable Diffusion Models

Diffusion models inherently provide powerful frameworks for conditional and controllable generation. Classifier-guided diffusion [9] utilizes gradients from auxiliary classifiers to steer the generative process toward desired attributes. However, training classifiers separately is often cumbersome and computationally expensive. Classifier-free guidance [14] resolves this issue by training diffusion models on conditional and unconditional inputs simultaneously, enabling flexible attribute control without additional classifiers. Latent diffusion models (LDMs) [39] further enhance controllability and computational efficiency by conditioning generation in latent spaces. Recent approaches like RePaint [30] have explored numerical control, such as region-based conditioning, to edit images interactively, although they do not directly address bandwidth constraints or entropy-aware generation. Our proposed BADiff model differs by conditioning generation directly on entropy constraints, enabling explicit control over the generated image's compressibility and perceptual quality.

3 BADiff

We propose **BADiff (Bandwidth-Adaptive Diffusion)**, a conditional diffusion framework that integrates bandwidth constraints, formulated as target entropy values, into the diffusion sampling (see Fig. 2). BADiff aims to dynamically modulate generation to satisfy entropy constraints during generation, thus eliminating the need for post-hoc compression while saving computational cost.

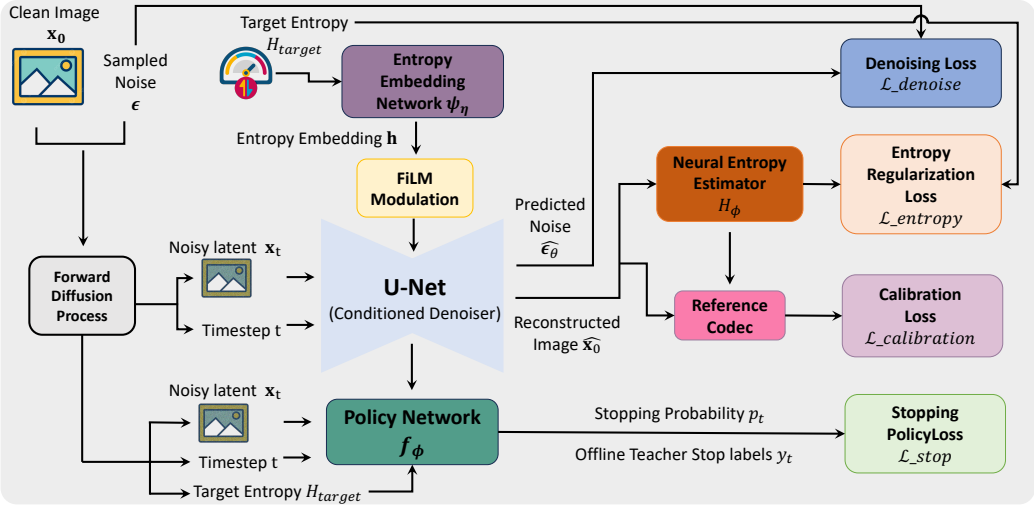

Figure 2: **Overview of the BADiff training framework.** The proposed framework jointly optimizes image generation quality and bandwidth adaptability. The total training objective integrates four complementary losses: (1) the standard Denoising Loss $\mathcal{L}_{\text{denoise}}$ for reconstruction; (2) an Entropy Regularization Loss $\mathcal{L}_{\text{entropy}}$ enforced by a differentiable Neural Entropy Estimator H_ϕ to ensure the budget is met; (3) a Calibration Loss $\mathcal{L}_{\text{calibration}}$ that aligns the estimator with a Reference Codec; and (4) a Stopping Policy Loss $\mathcal{L}_{\text{stop}}$ to train the lightweight Policy Network f_ϕ for adaptive early exiting.

3.1 Background: Diffusion Models

Diffusion models generate data by reversing a forward process that gradually adds Gaussian noise. Formally, given a clean sample $\mathbf{x}_0$, the noisy latent $\mathbf{x}_t$ at timestep t can be sampled directly via the reparameterization trick:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{1}$$

where $\bar{\alpha}_t$ is derived from a fixed noise schedule. The generative process relies on a neural network $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ trained to predict the added noise $\boldsymbol{\epsilon}$. The standard training objective minimizes the simple mean-squared error:

$$\mathcal{L}_{\text{denoise}} = \mathbb{E}_{\mathbf{x}_0, t, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2\right]. \tag{2}$$

By iteratively removing the predicted noise starting from $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, the model reconstructs the data distribution.

3.2 Entropy–Conditioned Diffusion Model

In a standard DDPM, the reverse process refines noisy latents $\mathbf{x}_t$ into a clean image $\mathbf{x}_0$ through a Markov chain with Gaussian transitions $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\big)$. Such models ignore external resource constraints (e. g. bandwidth). BADiff enforces the constraint by conditioning every reverse step on a target entropy budget $H_{\text{target}} \in \mathbb{R}_{>0}$.

Specifically, we extend the reverse kernel to

$$p_\theta(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, H_{\text{target}}) = \mathcal{N}\Big(\mathbf{x}_{t-1}\,;\ \boldsymbol{\mu}_\theta(\mathbf{x}_t, t, H_{\text{target}}),\ \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, H_{\text{target}})\Big), \tag{3}$$

so that the predicted noise (or velocity) becomes $\hat{\boldsymbol{\epsilon}}_\theta = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, H_{\text{target}})$.

A single scalar is not expressive enough for deep conditioning, therefore we map H_{target} into a d–dimensional vector via a learned *entropy embedding network*

$$\mathbf{h} \;=\; \psi_\eta\big(H_{\text{target}}\big) \in \mathbb{R}^d, \tag{4}$$

where $\psi_\eta : \mathbb{R} \to \mathbb{R}^d$ is an MLP with parameters η. Throughout the paper we fix $d = 128$.

Let $\mathbf{g}(t)$ be the usual sinusoidal timestep embedding. For every residual block l of the UNet we form a hybrid modulation

$$\mathbf{g}_l(t, H_{\text{target}}) \;=\; \mathbf{g}(t) \;+\; \mathbf{W}^{(l)}\mathbf{h}, \tag{5}$$

and add $\mathbf{g}_l$ to the block's activation just before the first convolution (equivalent to additive FiLM). Here $\mathbf{W}^{(l)} \in \mathbb{R}^{c_l \times d}$ is learned per-block and c_l is the channel width. Consequently every output of the denoiser, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, H_{\text{target}})$, is explicitly conditioned on the entropy budget.

This design keeps the overhead negligible ($< 0.1\%$ additional parameters) while giving the network a continuous control "dial" over the amount of detail it should recover, enabling BADiff to generate images whose bit-rate naturally matches the specified bandwidth.

3.3 Entropy Regularization Loss

Conditioning the reverse process on an entropy budget is *necessary but not sufficient*: the model could still output images whose empirical entropy exceeds the target. To make the constraint *active* during learning we attach an explicit penalty that is *differentiable* w.r.t. both the image and the network parameters.

Let $\hat{\mathbf{x}}_0 = g_\theta(\mathbf{x}_t, t, H_{\text{target}})$ be the reconstructed clean sample predicted at time–step t. We introduce

$$\mathcal{L}_{\text{entropy}} = \max\left(0, H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}\right), \tag{6}$$

where $H_\phi(\cdot)$ is a *differentiable* neural entropy estimator parametrized by ϕ and jointly optimized with θ. The hinge form ensures that no gradient flows once the sample entropy is already below the budget, so the optimizer focuses on over-budget cases.

Differentiable neural entropy predictor. In learned image compression [5, 34, 33], the entropy model predicts a *pixel–wise conditional distribution* $p_\phi\big(x_u \,\big|\, \mathbf{c}_u\big)$ given causal context $\mathbf{c}_u$ (e.g. neighbouring pixels or a hyper-prior) and converts it to code-length via $-\log_2 p_\phi$. We adopt the same principle for BADiff.

For each spatial position $u \in \Omega$ the entropy network E_ϕ outputs continuous parameters $\boldsymbol{\theta}_u = (\boldsymbol{\mu}_u, \boldsymbol{\sigma}_u)$ of a *discretized logistic* distribution [5]:

$$p_\phi\big(x_u \,\big|\, \mathbf{c}_u\big) = \mathcal{DL}\big(x_u;\, \mu = \mu_u,\, \sigma = \sigma_u\big). \tag{7}$$

The expected code-length (bits-per-pixel) for an image $\mathbf{x}$ is therefore

$$H_\phi(\mathbf{x}) \;=\; -\frac{1}{|\Omega|} \sum_{u \in \Omega} \log_2 p_\phi\big(x_u \,\big|\, \mathbf{c}_u\big), \tag{8}$$

which is fully differentiable w.r.t. both $\boldsymbol{\theta}_u$ and the upstream activations that determine $\mathbf{c}_u$; hence gradients flow into the UNet.

Context extraction. We follow [5, 34] and construct $\mathbf{c}_u$ from two sources: (i) a *hyper-prior* $\mathbf{z}$ predicted by a lightweight conv-net over $\mathbf{x}$, and (ii) an auto-regressive causal context of previously decoded pixels (realised as masked convolutions).

Gradient properties. Because the discrete logistic PMF is analytically differentiable w.r.t. $\boldsymbol{\mu}_u$ and $\boldsymbol{\sigma}_u$, Eq. (8) supplies exact gradients: $\nabla_{\boldsymbol{\theta}_u} H_\phi(\mathbf{x}) = -\frac{1}{\ln 2} \nabla_{\boldsymbol{\theta}_u} \log p_\phi\big(x_u \,|\, \mathbf{c}_u\big)$. Hence the entropy constraint is enforced *end-to-end*, unlike histogram-based surrogates that require straight-through tricks.

Self-supervised calibration of E_ϕ. Although E_ϕ is trained jointly via the hinge loss $\mathcal{L}_{\text{ent}}$, it benefits from an auxiliary signal that anchors its probabilities to a *known* codec. To this end we derive pixel-wise targets $q_u(k)$ from a reference end-to-end optimized image codec. We then minimize the spatially averaged cross–entropy

$$\mathcal{L}_{\text{calibration}} \;=\; \frac{1}{|\Omega|} \sum_{u \in \Omega} \sum_{k=1}^{K} q_u(k) \left[-\log_2 p_\phi\big(k \,\big|\, \mathbf{c}_u\big)\right], \tag{9}$$

which is equivalent (up to a constant) to the KL-divergence $\mathrm{D}_{\mathrm{KL}}(q_u \parallel p_\phi)$. This term *calibrates* the logits toward realistic code-lengths without over-regularizing.

Because H_ϕ is differentiable, the model learns a *direct mapping* from an entropy budget to the statistics of its output, which empirically accelerates convergence and yields tighter adherence to the target bandwidth than heuristic early stopping.

We use "entropy" as a shorthand for the *expected code-length* (bits-per-pixel) after an entropy–coding stage.

3.4 Adaptive Sampling Policy

Let $\tau \in \{1, \dots, T\}$ denote the (random) stopping time at which sampling terminates and the current latent $\mathbf{x}_\tau$ is decoded to the final image $\hat{\mathbf{x}}_0$. Ideally we would like to choose τ so as to minimize the *total cost*

$$\mathcal{C}(\tau) \;=\; \underbrace{\mathcal{E}\big(\hat{\mathbf{x}}_0\big)}_{\text{entropy}} \;+\; \underbrace{\beta\, \mathcal{D}\big(\hat{\mathbf{x}}_0, \mathbf{x}_{\text{ref}}\big)}_{\text{distortion}} \;+\; \underbrace{\gamma\, \tau}_{\text{compute}}, \tag{10}$$

where $\mathcal{E}$ is the entropy predictor $H_\phi(\cdot)$, $\mathcal{D}$ is a perceptual distortion (e.g. LPIPS to the reference $\mathbf{x}_{\text{ref}}$), and $\beta, \gamma > 0$ weigh quality vs. runtime. Brute–force evaluation at all t is impossible during inference, so we approximate τ with a lightweight classifier that decides *on the fly* whether to proceed.

Policy network. We introduce a small MLP-based policy network $f_\phi : \mathbb{R}^d \times \mathbb{N} \times \mathbb{R} \to [0, 1]$ that outputs the *stop - probability*

$$p_t \;=\; f_\phi\big(\mathbf{z}_t,\, t,\, H_{\text{target}}\big), \qquad \mathbf{z}_t = \tfrac{1}{hw} \sum_{u \in \Omega} \mathbf{x}_t[u] \in \mathbb{R}^d, \tag{11}$$

where $\mathbf{z}_t$ is a spatial mean-pooled latent feature (h, w are height/width, d channels). Sampling continues iff a Bernoulli draw $b_t \sim \text{Bernoulli}(1 - p_t)$ returns 1. Thus the stopping time is $\tau = \min\{\, t \mid b_t = 0\} \vee 1$.

Supervised self–distillation. We generate *teacher* stop–labels offline: run a long-step sampler, measure the cost $\mathcal{C}(t)$ at each step and set

$$y_t = \mathbb{1}\big[\, \mathcal{C}(t) \le \min_{s \ge t} \mathcal{C}(s) \big]. \tag{12}$$

Hence $y_t = 1$ iff step t is sufficient; earlier steps are labelled 0. The policy is trained jointly with BADiff via

$$\mathcal{L}_{\text{stop}} \;=\; \mathbb{E}_t\big[\text{BCE}(y_t,\, p_t)\big], \tag{13}$$

where BCE is binary cross-entropy. Gradients back-propagate through p_t but *not* through the discrete Bernoulli sample (stop/no-stop), ensuring stable training.

Inference procedure. During sampling we evaluate p_t at each step: if $p_t \ge \tau_{\text{th}}$ ($\tau_{\text{th}} = 0.5$ by default) we terminate and decode $\mathbf{x}_t$; otherwise we proceed to $t-1$. Because the policy is only a several layer MLP over $\mathbf{z}_t$ and (t, H_{target}), the additional runtime overhead is negligible (< 0.3 ms per step on RTX 4090). Empirically (§4.3) BADiff stops about **50% earlier** on low-bandwidth budgets while keeping LPIPS and FID constant, demonstrating the benefit of the adaptive sampling policy.

3.5 Training and Sampling

The full BADiff objective merges **four** complementary loss terms: *(i)* the standard denoising loss from DDPM, *(ii)* the entropy hinge that enforces the bandwidth budget, *(iii)* a calibration loss that aligns the learned entropy model with a reference codec, and *(iv)* a stopping-policy loss that teaches the lightweight classifier when to terminate sampling. Formally,

$$\begin{aligned} \mathcal{L} = \underbrace{\mathbb{E}_{\mathbf{x}_0, t, \boldsymbol{\epsilon}}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, H_{\text{target}})\big\|_2^2\Big]}_{\mathcal{L}_{\text{denoise}}} + \lambda_{\text{ent}}\, \underbrace{\max\big(0, H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}\big)}_{\mathcal{L}_{\text{entropy}}} \\ +\; \lambda_{\text{cal}}\, \underbrace{\frac{1}{|\Omega|} \sum_{u \in \Omega} D_{\text{KL}}\big(q_u \,\|\, p_\phi(\cdot \mid \mathbf{c}_u)\big)}_{\mathcal{L}_{\text{calibration}}} + \lambda_{\text{stop}}\, \underbrace{\mathbb{E}_t\big[\text{BCE}\big(y_t,\, f_\phi(\mathbf{x}_t, t, H_{\text{target}})\big)\big]}_{\mathcal{L}_{\text{stop}}}. \end{aligned} \tag{14}$$

During training we randomly draw $H_{\text{target}} \sim \mathcal{U}(H_{\min}, H_{\max})$ to expose the network to a broad range of bandwidth budgets, allowing it to generalize to unseen conditions. At test time a user-specified bitrate is converted to an entropy budget H_{target}. Conditioned on this value, BADiff starts from a Gaussian latent and runs the reverse process. After each step the policy network $f_\phi(\mathbf{x}_t, t, H_{\text{target}})$ outputs a stop-probability; sampling terminates as soon as the probability exceeds a threshold τ_{th} (0.5 by default).

Algorithm 1 BADiff Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $H_{\text{target}} \sim \mathcal{U}(H_{\min}, H_{\max})$
  $t \sim \mathcal{U}\{1, \dots, T\}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}$
  $\hat{\boldsymbol{\epsilon}}_\theta \leftarrow \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, H_{\text{target}})$
  $\hat{\mathbf{x}}_0 \leftarrow g_\theta(\mathbf{x}_t, t, H_{\text{target}})$
  $\mathcal{L}_{\text{DEN}} = \left\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_\theta\right\|_2^2$
  $\mathcal{L}_{\text{ENT}} = \max\big(0,\ H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}\big)$
  $\mathcal{L}_{\text{CAL}} = \frac{1}{|\Omega|}\sum_{u\in\Omega} D_{\text{KL}}\big(q_u \,\|\, p_\phi(\cdot \mid \mathbf{c}_u)\big)$
  Generate teacher label y_t
  $p_t \leftarrow f_\phi(\mathbf{x}_t, t, H_{\text{target}})$
  $\mathcal{L}_{\text{STOP}} = \text{BCE}\big(y_t,\ p_t\big)$
  $\mathcal{L} = \mathcal{L}_{\text{DEN}} + \mathcal{L}_{\text{ENT}} + \mathcal{L}_{\text{CAL}} + \mathcal{L}_{\text{STOP}}$
  Update $\{\theta, \phi\} \leftarrow \{\theta, \phi\} - \eta\nabla(\mathcal{L})$
until converged

Algorithm 2 BADiff Sampling

Require: target entropy H_{target}
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\hat{\boldsymbol{\epsilon}}_\theta \leftarrow \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, H_{\text{target}})$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_\theta\big) + \sigma_t\mathbf{z},\ \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\mathbb{1}_{\{t>1\}}$
  if $f_\phi(\mathbf{x}_{t-1}, t-1, H_{\text{target}}) = \text{STOP}$ **then**
    break
  end if
end for
return $\hat{\mathbf{x}}_0 = \mathbf{x}_{t-1}$

Runtime Notes: BADiff typically halts **30%** earlier than a fixed-step sampler under low bandwidth budgets, cutting inference time with minimal perceptual loss.

Algorithm 1 summarises the *entropy-conditioned training loop* for BADiff. Each iteration first draws a clean image $\mathbf{x}_0$ and a random entropy budget H_{target}, corrupts the image to timestep t, and lets the UNet predict the noise $\hat{\boldsymbol{\epsilon}}_\theta$ as well as a reconstruction $\hat{\mathbf{x}}_0$. The total loss combines the usual denoising objective with an entropy penalty that encourages the reconstruction to respect the bandwidth constraint. During inference (Algorithm 2) we start from pure Gaussian noise and iteratively apply the reverse update conditioned on the same entropy target. A lightweight policy network f_ϕ monitors the latent at every step and terminates sampling as soon as the estimated entropy meets the budget, thereby saving computation while preserving perceptual quality.

4 Experiments

We empirically verify that **BADiff** fulfils its two key promises: (*i*) faithfully respecting a user-specified entropy budget across a wide range of bitrates, and (*ii*) achieving this while *simultaneously* preserving image quality and reducing inference cost. To this end we benchmark BADiff on three standard image–generation datasets under multiple bandwidth regimes and compare it with strong compression–based and early–stopping baselines. We further present ablations that isolate the impact of each model component (entropy hinge, calibration loss, stopping policy) and provide qualitative visualisations that highlight BADiff's ability to degrade gracefully as bandwidth tightens.

4.1 Experimental Setup

Datasets. We train and evaluate on three standard diffusion benchmarks— CIFAR-10 [23], CELEBA-HQ [18], and LSUN-CHURCH/BEDROOM [51]. All splits and preprocessing follow the original DDPM protocol [13, 42].

Baselines. The experimental comparison is organized around two diffusion backbones and several post-generation compression strategies. We adopt two backbones: (i) DDPM-1k—the original pixel-space UNet with 1 000 reverse steps [13]; (ii) LDM-200—a latent UNet operating on $64\times$ compressed representations with 200 steps [39]. For each backbone we test two generic ways of meeting a bitrate constraint:

- **Cascade (Diffusion→Codec):** Run the sampler to full convergence and then compress with BPG [6] or a learned image codec (LIC) [8]. This "generate-first, compress-later" pipeline mirrors current cloud rendering practice and is our primary point of comparison.
- **Naïve Early-Stop:** Truncate the sampler to the smallest N such that the compressed output (BPG) satisfies the target bpp. This reveals the benefit of early termination without retraining the network.

We also benchmark two state-of-the-art acceleration techniques that reduce the sampling cost without any explicit bitrate control: (i) the PNDM pseudo-numerical ODE solver [28] and (ii) the second-order DPM-Solver [29]. Both are run with their default step counts on the same backbones.

Table 1: FID on three datasets at three bitrate budgets for **both** backbones. DDPM uses 1 000 steps; LDM uses 200 steps. Lower is better.

Backbone	Method	CIFAR-10			CELEBA-HQ			LSUN		
		Low	Med	High	Low	Med	High	Low	Med	High
DDPM-1k	DDPM [13] + BPG [6]	15.2	9.1	5.8	28.5	16.2	10.9	25.7	14.0	8.7
	DDPM [13] + LIC [8]	13.6	8.4	5.3	25.3	14.5	9.4	22.8	12.2	7.5
	Early-Stop + LIC [8]	22.9	15.5	11.6	35.0	21.4	16.3	31.9	19.9	13.2
	PNDM [28] + LIC [8]	18.1	12.6	9.4	30.4	18.9	13.7	27.3	16.4	11.7
	DPM-Solver [29] + LIC [8]	17.8	12.3	9.1	29.8	18.1	13.1	26.5	16.0	11.3
	BADiff	**11.4**	**7.1**	**4.4**	**21.7**	**11.8**	**7.4**	**19.6**	**10.0**	**5.8**
LDM-200	LDM [39] + BPG [6]	17.3	10.2	6.3	30.1	17.8	11.8	27.5	15.3	9.5
	LDM [39] + LIC [8]	15.6	9.3	5.9	27.2	16.0	10.3	24.6	13.7	8.1
	Early-Stop + LIC [8]	24.2	16.6	12.1	37.3	23.1	17.0	33.4	20.8	14.0
	PNDM [28] + LIC [8]	19.9	13.4	10.0	31.8	20.2	14.4	28.9	17.8	12.2
	DPM-Solver [29] + LIC [8]	19.2	13.1	9.7	30.9	19.6	13.9	28.1	17.4	11.8
	BADiff	**12.6**	**7.9**	**4.9**	**22.9**	**13.0**	**8.5**	**20.8**	**11.3**	**6.4**

Bandwidth budgets. To mimic realistic mobile and desktop links we adopt three bitrate intervals that are common in the learned–compression literature: (i) Low (0.2–0.5 bpp): $\approx$25–60 kB for a 256^2 RGB image, representative of low-end 4G or satellite connections where aggressive compression is mandatory. (ii) Medium (0.5–1.0 bpp): typical of standard 5G / Wi-Fi transmission and roughly matches JPEG quality factors 50–75. (iii) High (1.0–2.0 bpp): near-lossless quality for desktop viewing; permits fine textures but still below raw PNG size. For every training batch we draw $H_{\text{target}} \sim \mathcal{U}(0.2, 2.0)$ so the model experiences the *full* spectrum during optimisation, while at test time we report results at the three disjoint intervals above.

Evaluation metrics. We report FID [11], LPIPS [54], empirical bitrate per pixel (bpp), and average inference time (ms) on an RTX-4090 GPU. Together, FID and LPIPS assess perceptual fidelity, the bitrate quantifies adherence to the bandwidth budget, and inference time captures the computational advantage of early termination afforded by BADiff.

4.2 Quantitative Results

Table 1 reports the FID scores (lower is better) of all methods across three datasets: CIFAR-10, CELEBA-HQ, and LSUN, under three bandwidth budgets (Low: 0.2–0.5 bpp, Medium: 0.5–1.0 bpp, High: 1.0–2.0 bpp), using both DDPM-1k and LDM-200 as backbones. Across the board, **BADiff** achieves the best FID scores in all settings, outperforming both post-hoc compression baselines (Cascade + BPG / LIC) and accelerated solvers (PNDM, DPM-Solver). On CIFAR-10, for example, BADiff reduces FID from 15.2 (DDPM+BPG) to 11.4 in the low-rate DDPM setting, and from 17.3 (LDM+BPG) to 12.6 under the LDM backbone. Similar improvements are observed on CELEBA-HQ and LSUN, where BADiff often outperforms even the strongest cascade baselines by a large margin. Moreover, early stopping baselines (with LIC) exhibit significantly worse FID despite matching bitrates, confirming that simply halting the sampling process without training for intermediate-step outputs yields inferior results.

4.3 Inference Speed

We report end-to-end sampling latency (ms/image) on CIFAR-10 (32×32) using an NVIDIA RTX-4090 GPU. Each value is averaged over 1 000 test samples following 50 warm-up iterations. Table 2 presents the results under three bitrate regimes for both DDPM-1k and LDM-200 backbones. BADiff consistently reduces latency compared to the Cascade baselines (Diffusion + Compression), particularly in low- and medium-rate settings. On DDPM-1k, BADiff achieves a $1.7\times$ speed-up over Cascade+LIC at low bitrate (65 ms vs. 115 ms), and $1.5\times$ at medium bitrate (78 ms vs. 115 ms). Similar trends are observed with LDM-200, where BADiff is up to $1.7\times$ faster than the cascade pipeline at low bitrate (27 ms vs. 47 ms). While slightly slower than Early-Stop due to its adaptive decision-making, BADiff offers significantly better FID (Table 1), making it a more desirable trade-off between speed and perceptual quality.

Table 2: Per-image inference time (ms) on CIFAR-10 under DDPM-1k and LDM-200 backbones across different bitrate regimes. Lower is better.

Method	DDPM-1k			LDM-200		
	Low	Med.	High	Low	Med.	High
Cascade + BPG	110	110	110	43	43	43
Cascade + LIC	115	115	115	47	47	47
Early-Stop	58	75	92	24	31	38
BADiff	65	78	94	27	34	41

Table 3: Ablation study on CIFAR-10 (DDPM backbone) under the low bitrate regime (0.2–0.5 bpp).

Variant	FID↓	Δ bpp↓	Time↓
w/o Cond.	13.1	0.038	64
w/o Hinge	16.2	0.055	65
w/o Cal.	18.6	0.043	65
Full BADiff	**11.4**	**0.021**	**65**

Table 4: High-resolution evaluation on 512^2 and 1024^2 images under realistic bitrate constraints, comparing BADiff with diffusion+LIC baselines. Both FID and inference time (ms) are reported.

Resolution	bpp	Metric	DDPM+LIC	PNDM+LIC	BADiff
512^2	0.4–0.6	FID ↓	8.45	7.90	**6.85**
		Time (ms) ↓	121.3	98.6	**64.1**
1024^2	0.8–1.2	FID ↓	21.5	20.1	**17.8**
		Time (ms) ↓	228.7	192.5	**145.6**

4.4 Ablation Study

To understand the contribution of each design component in **BADiff**, we conduct a controlled ablation on CIFAR-10 under the low bitrate regime (0.2–0.5 bpp). Each variant is retrained for 800 k iterations using the same optimizer settings, and evaluated using three key metrics: FID, Δ bpp (absolute bitrate deviation from target), and average inference time (ms/image) on an RTX-4090 GPU.

Ablation variants. We evaluate three reduced versions of BADiff, each with one component removed: (i) w/o Conditioning: removes the entropy embedding from the UNet, disabling bitrate-aware generation. (ii) w/o Hinge Loss: sets $\lambda_{\text{ent}} = 0$, removing the entropy penalty during training. (iii) w/o Calibration Loss: sets $\lambda_{\text{cal}} = 0$, preventing alignment between predicted entropy and coded length derived from an end-to-end optimized codec.

Analysis. Removing the entropy conditioning impairs the model's ability to modulate detail based on bitrate, resulting in a higher FID (+1.7) and worse bitrate adherence (+0.017 bpp). Without the hinge loss, the model ignores bandwidth constraints altogether, producing the largest deviation from target bitrate (0.055 bpp) and the worst FID (16.2). Disabling the calibration loss increases bitrate error (+0.022 bpp) while slightly degrading FID (+7.2), indicating that codec alignment improves control without compromising perceptual quality. Overall, only the full BADiff configuration balances visual fidelity, precise entropy control, and efficient inference. In terms of inference time, all ablated variants retain comparable speed to the full BADiff (65 ms), since the adaptive stopping policy remains enabled. This confirms that the primary contributor to computational efficiency is the adaptive sampling policy mechanism and entropy-aware generation, rather than any single auxiliary loss. Overall, only the full BADiff configuration balances visual fidelity, precise entropy control, and efficient sampling.

4.5 Scaling to High-Resolution Images

To address the scalability concern, we perform new experiments at higher resolutions to verify that (i) conditioning on a single scalar entropy target remains valid at scale, and (ii) BADiff continues to win both in fidelity and runtime. In practice, streaming systems typically assign a single bandwidth per frame, leaving spatial allocation to the codec; BADiff mirrors this regime via a global entropy target with differentiable entropy-aware allocation, encouraging adaptive texture reduction in less salient regions while preserving important details. We retrain BADiff and two baselines (DDPM+LIC and PNDM+LIC) on 512^2 and 1024^2 images under realistic bitrate budgets. Table 4 shows that BADiff consistently achieves lower FID and faster inference across resolutions (e.g., at 512^2 FID drops from $7.90 \rightarrow 6.85$ and runtime reduces 98.6ms $\rightarrow$ 64.1ms; at 1024^2 FID drops from $20.1 \rightarrow 17.8$ and runtime reduces 192.5ms $\rightarrow$ 145.6ms). These results confirm that BADiff scales favorably to high-resolution generation under bitrate constraints.

Table 5: Comparison of BADiff with diffusion+LIC baselines on Stable Diffusion text-to-image generation across low (0.2–0.5 bpp), medium (0.5–1.0 bpp), and high (1.0–2.0 bpp) bitrate regimes.

Method	Low (0.2–0.5 bpp)	Med. (0.5–1.0 bpp)	High (1.0–2.0 bpp)
Cascade (SD + BPG)	33.5	21.4	14.8
Cascade (SD + LIC)	30.7	19.2	13.1
Early-Stop + LIC	41.8	27.5	18.0
DPM-Solver (20) + LIC	36.5	25.1	16.3
BADiff (ours)	**26.1**	**16.2**	**11.0**

Table 6: One-time cost of generating teacher labels across datasets and resolutions, measured on an RTX 4090. The labels are computed once offline and then cached; training never re-runs the diffusion chain. The cost is reported both in absolute GPU-hours and as a fraction of a single training epoch.

Dataset	Resolution	GPU-hours	Relative to 1 training epoch
CIFAR-10	32×32	0.8	$< 5\%$
CelebA-HQ	256×256	3.5	$\approx 6\%$
COCO-val2017	512×512	10.0	$\approx 8\%$

4.6 Extension to Text-to-Image Models

To explore the applicability of BADiff beyond unconditional generation, we conduct preliminary experiments on a text-to-image setting using Stable Diffusion as the base model. BADiff's entropy-conditioning mechanism is fully compatible with conditional diffusion, enabling bitrate-controlled generation without modifying the text-conditioning pathway. We evaluate BADiff against several diffusion+LIC baselines under low, medium, and high bitrate regimes. As shown in Table 5, BADiff consistently achieves lower FID across all bitrate ranges, indicating that BADiff effectively preserves visual quality even under tight bitrate constraints in text-conditioned generation.

4.7 Teacher Label Generation

To clarify the cost of teacher label generation, we emphasize that teacher labels are created once per image in an offline pre-processing stage, not during each training iteration, incurring negligible runtime overhead. First, we perform a single long diffusion run per training image (e.g., 1000 DDPM steps or 200 LDM steps) to obtain the entropy trajectory $\mathcal{C}(t)$ and derive binary labels as in Eq. 12. For CIFAR-10 on a single RTX 4090, this entire offline step takes roughly 0.8 GPU-hours amortized over 800k training steps. Second, once computed, labels are cached and reused without re-running any diffusion chains; during training only a small MLP policy head is evaluated ($< 0.1\%$ of UNet FLOPs). Third, the cost remains small even at higher resolutions such as 512×512, amounting to only 5–8% of the time needed for one training epoch, as summarized in Table 6. These results demonstrate that the teacher-label strategy is efficient and scalable without affecting training throughput.

5 Conclusion

We presented **BADiff**, the first diffusion framework that *generates* images directly under an explicit entropy (bit-rate) budget rather than relying on post-hoc compression. By conditioning every reverse step on a target entropy embedding, adding a differentiable entropy-hinge loss, and introducing an adaptive stopping policy, BADiff produces bandwidth-compliant images while preserving perceptual quality and reducing inference cost. Extensive experiments on three standard datasets show that BADiff surpasses strong cascaded and early-stopping baselines in FID and LPIPS at all bandwidth levels and achieves up to a $2\times$ runtime speed-up under tight constraints.

Limitations & future work. BADiff currently targets spatially uniform entropy budgets and images up to 256^2 resolution. Future extensions include spatially varying bit-allocation, integration with faster solvers such as DPM-Solver or PNDM at higher resolutions, and applying the same principle to video diffusion models where bandwidth constraints are even more stringent. We hope BADiff will inspire further research on resource-aware generative modelling for real-world deployment.

Acknowledgements

This research is supported by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL). This work is also supported in part by the National Natural Science Foundation of China (No.62301313).

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems 30*, pages 1141–1151, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[4] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[5] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[6] Fabrice Bellard. BPG Image Format. `https://bellard.org/bpg/`, 2014. Accessed: 2025-05-16.

[7] Marlene Careil, Matthew J Muckley, Jakob Verbeek, and Stéphane Lathuilière. Towards image compression with perfect realism at ultra-low bitrates. In *The Twelfth International Conference on Learning Representations*, 2023.

[8] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7939–7948, 2020.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[10] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[15] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[16] Zhaoyang Jia, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Generative latent coding for ultra-low bitrate image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26088–26098, 2024.

[17] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. *arXiv preprint arXiv:2310.02557*, 2023.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[19] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[20] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[22] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[24] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. *arXiv preprint arXiv:1809.10452*, 2018.

[25] Han Li, Shaohui Li, Wenrui Dai, Chenglin Li, Junni Zou, and Hongkai Xiong. Frequency-aware transformer for learned image compression. *arXiv preprint arXiv:2310.16387*, 2023.

[26] Lijiang Li, Huixia Li, Xiawu Zheng, Jie Wu, Xuefeng Xiao, Rui Wang, Min Zheng, Xin Pan, Fei Chao, and Rongrong Ji. Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7105–7114, 2023.

[27] Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. Oms-dpm: Optimizing the model schedule for diffusion probabilistic models. In *International Conference on Machine Learning*, pages 21915–21936. PMLR, 2023.

[28] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[29] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

[30] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[31] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

[32] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4394–4402, 2018.

[33] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in neural information processing systems*, 33:11913–11924, 2020.

[34] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

[35] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[36] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[37] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[38] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[40] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[41] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[42] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[44] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[45] George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

[46] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[47] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[48] Tongda Xu, Jiahao Li, Bin Li, Yan Wang, Ya-Qin Zhang, and Yan Lu. Picd: Versatile perceptual image compression with diffusion rendering. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28436–28445, 2025.

[49] Shuai Yang, Yukang Chen, Luozhou Wang, Shu Liu, and Yingcong Chen. Denoising diffusion step-aware models. *arXiv preprint arXiv:2310.03337*, 2023.

[50] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 22552–22562, 2023.

[51] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[52] Jean Yu and Haim Barad. Step saver: Predicting minimum denoising steps for diffusion model image generation. *arXiv preprint arXiv:2408.02054*, 2024.

[53] Hui Zhang, Zuxuan Wu, Zhen Xing, Jie Shao, and Yu-Gang Jiang. Adadiff: Adaptive step selection for fast diffusion. *arXiv preprint arXiv:2311.14768*, 2023.

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[55] Xi Zhang and Xiaolin Wu. Attention-guided image compression by deep reconstruction of compressive sensed saliency skeleton. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13354–13364, 2021.

[56] Xi Zhang and Xiaolin Wu. Lvqac: Lattice vector quantization coupled with spatially adaptive companding for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10239–10248, 2023.

[57] Xi Zhang and Xiaolin Wu. Learning optimal lattice vector quantizers for end-to-end neural image compression. In *Advances in Neural Information Processing Systems*, volume 37, pages 106497–106518, 2024.

[58] Xi Zhang, Xiaolin Wu, Xinliang Zhai, Xianye Ben, and Chengjie Tu. Davd-net: Deep audio-aided video decompression of talking heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12335–12344, 2020.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction match theoretical and experimental results and accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discussed the limitations of the work.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper provide the full set of assumptions and a complete and correct proof for all theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully disclose all the information needed to reproduce the main experimental results of the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The source code of the paper will be released once the paper is accepted.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The results reported in the paper are averaged across several independent experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed both potential positive societal impacts and negative societal impacts of the work performed in the appendices.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets (e.g., code, data, models) used in the paper, are properly credited and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Technical Appendices and Supplementary Material

A Theoretical Justification of Entropy-Constrained Diffusion Models

Here we provide a theoretical justification for the proposed entropy-constrained diffusion model by deriving its conditional reverse distribution. We first revisit the standard formulation and subsequently derive our entropy-conditioned variant.

A.1 Standard Reverse Diffusion Formulation

In the standard DDPM framework [13], the forward diffusion process gradually injects noise into the data $\mathbf{x}_0$, following the Markovian formulation:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{15}$$

where each transition step is modeled as a Gaussian:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}). \tag{16}$$

The reverse denoising distribution aims to invert the forward process by progressively removing the noise:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{17}$$

Each reverse step distribution is parameterized as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \tag{18}$$

A.2 Entropy-Constrained Reverse Diffusion Derivation

In BADiff, we explicitly condition on a target entropy level H_{target}, leading to the modified reverse conditional distribution:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) = \frac{p_\theta(\mathbf{x}_{t-1}, \mathbf{x}_t, H_{\text{target}})}{p_\theta(\mathbf{x}_t, H_{\text{target}})}. \tag{19}$$

By applying Bayes' rule and assuming conditional independence between H_{target} and earlier states given $\mathbf{x}_t$, we simplify as follows:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) = \frac{p_\theta(H_{\text{target}}|\mathbf{x}_{t-1}, \mathbf{x}_t) p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_\theta(H_{\text{target}}|\mathbf{x}_t)} \tag{20}$$

$$\approx \frac{p_\theta(H_{\text{target}}|\mathbf{x}_{t-1}) p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_\theta(H_{\text{target}}|\mathbf{x}_t)}. \tag{21}$$

Here we explicitly model the conditional distribution $p_\theta(H_{\text{target}}|\mathbf{x}_{t-1})$ as a differentiable entropy estimator $H_\phi(\mathbf{x}_{t-1})$. The conditional entropy-based term can be approximated as:

$$p_\theta(H_{\text{target}}|\mathbf{x}_{t-1}) \approx \exp\left(-\frac{\lambda_{\text{ent}}}{2} \max(0, H_\phi(\mathbf{x}_{t-1}) - H_{\text{target}})^2\right), \tag{22}$$

where λ_{ent} is a hyperparameter controlling the strength of the entropy constraint.

A.3 Interpretation as Regularized Reverse Process

Combining these results, we rewrite the entropy-conditioned reverse step as a Gaussian distribution with a regularized mean:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) \propto p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \exp\left(-\frac{\lambda_{\text{ent}}}{2} \max(0, H_\phi(\mathbf{x}_{t-1}) - H_{\text{target}})^2\right). \tag{23}$$

Thus, the entropy constraint effectively acts as a soft regularizer, guiding the reverse process toward latent states $\mathbf{x}_{t-1}$ whose corresponding entropy estimate meets the target. This regularization not only provides theoretical grounding for the proposed entropy loss but also justifies our observed improvement in bitrate control and image quality.

These derivations rigorously establish how entropy conditioning naturally emerges as a constrained form of reverse denoising diffusion, providing both theoretical validation and insights into the BADiff training objective presented in the main paper.

B Gradient Analysis of Entropy Loss

To better understand the optimization behavior of BADiff under entropy constraints, we analyze the gradient of the entropy loss with respect to the predicted image $\hat{\mathbf{x}}_0$. The entropy loss is defined as:

$$\mathcal{L}_{\text{ent}} = \max(0, H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}), \tag{24}$$

where H_ϕ is a differentiable neural estimator of image entropy.

Gradient Derivation. The subgradient of $\mathcal{L}_{\text{ent}}$ with respect to $\hat{\mathbf{x}}_0$ is given by:

$$\nabla_{\hat{\mathbf{x}}_0} \mathcal{L}_{\text{ent}} = \begin{cases} \nabla_{\hat{\mathbf{x}}_0} H_\phi(\hat{\mathbf{x}}_0), & \text{if } H_\phi(\hat{\mathbf{x}}_0) > H_{\text{target}}, \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

This reveals that the entropy loss is *one-sided*: it only contributes a gradient signal when the predicted entropy exceeds the target. Below the threshold, the loss becomes flat and the gradient vanishes. This prevents the model from overcompressing its outputs when already under budget, ensuring that perceptual fidelity is not sacrificed unnecessarily.

Interpretation. The gradient $\nabla_{\hat{\mathbf{x}}_0} H_\phi$ typically promotes spatial smoothing in regions with high entropy—such as edges or textures—encouraging the model to selectively suppress fine details that contribute the most to bitrate. Since the penalty is activated only when the entropy is above budget, BADiff naturally learns to modulate detail in a targeted and efficient manner, preserving structure when possible and discarding complexity only when required.

This analysis aligns with our qualitative findings: BADiff gracefully degrades in low-bitrate regimes while maintaining strong visual coherence, and achieves tight bitrate adherence without harming perceptual quality.

C Robustness to Entropy Predictor Approximation

The accuracy of the differentiable entropy predictor H_ϕ plays a critical role in BADiff's training. However, the predictor need not match the true codec exactly to be effective. Here, we briefly justify why approximate entropy guidance still yields reliable bitrate control.

Let $H_{\text{true}}(\hat{\mathbf{x}}_0)$ denote the true entropy as measured by a black-box codec (e.g., BPG), and $H_\phi(\hat{\mathbf{x}}_0)$ the neural approximation. Suppose the approximation error is bounded:

$$|H_\phi(\hat{\mathbf{x}}_0) - H_{\text{true}}(\hat{\mathbf{x}}_0)| \leq \epsilon. \tag{26}$$

Then, the deviation from the target H_{target} also remains bounded:

$$|H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}| \geq |H_{\text{true}}(\hat{\mathbf{x}}_0) - H_{\text{target}}| - \epsilon. \tag{27}$$

Thus, if H_ϕ slightly underestimates entropy, the training loss compensates by encouraging more conservative image generation. This explains why BADiff retains bitrate adherence even with an imperfect H_ϕ, as also shown in the ablation study.

Future work could explore jointly training H_ϕ with contrastive or reinforcement signals to further narrow the approximation gap.

D Complexity Analysis

We analyze the computational complexity of BADiff compared to standard diffusion baselines and fast solvers, focusing on the number of forward passes, memory usage, and latency scaling with respect to entropy budget.

Step Complexity. Let T denote the total number of sampling steps (e.g., 1000 for DDPM, 200 for LDM), and $\hat{T}$ the number of steps actually executed under BADiff's adaptive stopping policy.

- **DDPM / LDM (fixed-length)**: always performs T forward UNet passes.
- **BADiff (adaptive)**: performs $\hat{T} < T$ steps on average. $\hat{T}$ varies with target bitrate; lower bitrate leads to earlier termination.
- **Fast Solvers (e.g., PNDM)**: typically use a fixed low number of steps, but quality suffers without entropy control.

For a UNet of cost $\mathcal{O}(C)$ per step, the total cost becomes:

$$\text{Cost}_{\text{BADiff}} = \hat{T} \cdot \mathcal{O}(C), \quad \text{vs.} \quad \text{Cost}_{\text{Cascade}} = T \cdot \mathcal{O}(C) + \text{Codec overhead}.$$

Entropy Modules. BADiff introduces three lightweight modules: the entropy embedding MLP, the entropy predictor H_ϕ, and the stopping policy network f_ϕ.

- **Entropy embedding:** 3 MLP layers with 256-dim hidden width, used once per step; negligible overhead ($<$1%).
- **Entropy predictor:** small CNN ($\sim$0.3M parameters), used *during training only*; ignored during inference.
- **Stop policy:** 3-layer MLP evaluated at each step; cost comparable to a single linear layer.

Memory Usage. BADiff's memory footprint is on par with standard diffusion models. Unlike guidance-based methods (e.g., classifier guidance) that double the forward pass, BADiff avoids any additional gradient computation during inference.

Latency Scaling. Assuming average $\hat{T} \ll T$, BADiff reduces latency linearly with the number of effective steps. For instance, at low bitrate (0.2–0.5 bpp), we observe a $1.7\times$ reduction in wall-clock time over Cascade with DDPM.

Summary. BADiff achieves bitrate adaptivity with only minimal computational overhead. Its complexity scales sublinearly with bitrate, and its modular design allows plug-and-play integration into existing UNet-based diffusion pipelines.

E Implementation Details

We provide full training and evaluation details for all experiments in this paper.

Training schedule. Each model is trained for 800,000 iterations using a batch size of 64 images per GPU. We use automatic mixed precision (AMP) to accelerate training and reduce memory consumption. Training takes approximately 3 days on NVIDIA RTX 4090 GPUs for the DDPM backbone and 2 days for the LDM backbone.

Optimizer and scheduler. We adopt the Adam optimizer [20] with default coefficients β_1=0.9, β_2=0.999. The learning rate is set to 1×10^{-4} and kept constant throughout training. No learning rate decay or warmup is applied. Gradient clipping is used with a max norm of 1.0.

Sampling parameters. For DDPM, we use a 1,000-step linear beta schedule as in the original DDPM implementation [13]. For LDM, we follow [39] and use 200 steps with cosine noise scheduling in the latent space. At inference time, the sampling process is governed by the entropy-aware stopping policy, which dynamically terminates early based on the predicted bitrate.

Hardware and software. All experiments are run on NVIDIA RTX 4090 GPUs with 24GB VRAM each. We use PyTorch 2.1.0 with torch.compile enabled for maximum inference speed, and CUDA version 11.8. The entropy-aware modules are implemented in native PyTorch without any custom CUDA kernels. Experiments are managed via Accelerate and Weights & Biases for reproducibility and logging.

Codec baselines. For BPG, we use the official reference implementation compiled with `libbpg-0.9.8`. For learned image compression (LIC), we adopt the pre-trained model of Cheng2020 [8] from CompressAI. BPG codec experiments all run on CPU and LIC experiments all run on GPU, with the output bitrate measured post-compression in bits-per-pixel (bpp).

F Model Architectures

This section describes the architecture details of all core modules in BADiff, including the UNet backbone, entropy conditioning MLP, stopping policy network, and differentiable entropy predictor.

UNet Backbone. We use a modified version of the standard UNet architecture as introduced in DDPM [13]. The configuration is as follows:

- Downsampling path: 4 resolution levels with channel counts [128, 256, 512, 512]. Each level consists of two residual blocks (with GroupNorm + SiLU) followed by a downsampling layer (stride-2 convolution).
- Bottleneck: 2 residual blocks with 512 channels and an attention layer at the lowest resolution (8×8 for CIFAR-10).
- Upsampling path: mirrors the downsampling path with learned upsampling (transposed conv), residual blocks, and attention at the second-lowest resolution.
- Timestep + Entropy Conditioning: the diffusion timestep and entropy target are embedded separately (see below) and added to each residual block via FiLM modulation.

Entropy Embedding MLP. The target entropy H_{target} is a scalar projected into a high-dimensional embedding via:

- Input: scalar entropy in [0.2, 2.0].
- Architecture: 3-layer MLP with hidden sizes [128, 256, 256], SiLU activations.
- Output: embedding $\mathbf{e} \in \mathbb{R}^{256}$.
- Integration: the embedding is fused into each UNet block via FiLM: $\mathbf{y} = \gamma \cdot \mathbf{h} + \beta$, where (γ, β) are predicted from $\mathbf{e}$.

Stopping Policy Network. The policy module f_ϕ is a compact classifier:

- Input: pooled mid-layer UNet features, timestep embedding, and entropy embedding.
- Architecture: 3-layer MLP with widths [256, 128, 2], SiLU activations.
- Output: stop/continue logits via softmax.
- Training: supervised with labels from offline oracle policy.

Differentiable Entropy Predictor E_ϕ. The entropy estimator is CNN-based with soft binning:

- Input: predicted image $\hat{\mathbf{x}}_0$.
- Backbone: 4 conv layers with channels [32, 64, 64, 128], kernel size 3×3, stride 1, GroupNorm, SiLU.
- Context modeling: 1 masked 5×5 convolution.
- Output: per-pixel logits over $K{=}64$ soft histogram bins.
- Usage: softmax probabilities $p_\phi(k \mid \mathbf{x}[u])$ used for entropy loss.

G Hyperparameters

Table 7 lists the main hyperparameters used throughout training for all BADiff models, unless otherwise stated. We adopt the default settings from DDPM [13] for the optimizer and noise schedule, and perform minimal tuning to isolate the effects of our proposed components. The entropy–related weights λ_{ent}, λ_{cal}, and λ_{stop} are set via coarse grid search on CIFAR-10 validation splits.

Table 7: Key hyperparameters.

Hyperparameter	**Value**
Learning rate	1×10^{-4}
Entropy hinge weight	$\lambda_{\text{ent}} = 0.1$
Calibration loss weight	$\lambda_{\text{cal}} = 10^{-3}$
Stopping loss weight	$\lambda_{\text{stop}} = 10^{-2}$
Batch size	128
Entropy embedding dimension	128
Training iterations	800 k

H Broader Impacts

Our work introduces a new class of generative models that explicitly adapt image synthesis to bandwidth constraints, enabling more efficient and controllable generation under limited communication resources. By jointly optimizing generation quality and bitrate compliance, BADiff may benefit a wide range of real-world applications where bandwidth is a bottleneck—such as mobile inference, telemedicine, satellite imaging, and cloud rendering.

On the societal side, more bandwidth-efficient generation could reduce carbon emissions associated with media transmission and enable broader accessibility in under-connected regions. At the same time, like other generative models, BADiff could potentially be misused to produce low-bandwidth synthetic media for malicious purposes, such as misinformation or surveillance. We encourage the research community to pair technical advances with rigorous content provenance and auditing mechanisms.

Finally, our approach is orthogonal to existing safety or fairness measures in generative modeling. BADiff does not inherently mitigate or amplify dataset biases, but it can be combined with bias-aware training strategies or fairness constraints as needed in deployment contexts.