

# 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 [2], the forward diffusion process gradually injects noise into the data  $\mathbf{x}_0$ , following the Markovian formulation:

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

where each transition step is modeled as a Gaussian:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1 - \alpha_t)\mathbf{I}). \quad (2)$$

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

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t). \quad (3)$$

Each reverse step distribution is parameterized as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)). \quad (4)$$

### A.2 Entropy-Constrained Reverse Diffusion Derivation

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

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) = \frac{p_\theta(\mathbf{x}_{t-1}, \mathbf{x}_t, H_{\text{target}})}{p_\theta(\mathbf{x}_t, H_{\text{target}})}. \quad (5)$$

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

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) = \frac{p_\theta(H_{\text{target}}|\mathbf{x}_{t-1}, \mathbf{x}_t)p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_\theta(H_{\text{target}}|\mathbf{x}_t)} \quad (6)$$

$$\approx \frac{p_\theta(H_{\text{target}}|\mathbf{x}_{t-1})p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{p_\theta(H_{\text{target}}|\mathbf{x}_t)}. \quad (7)$$

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:

$$p_\theta(H_{\text{target}}|\mathbf{x}_{t-1}) \approx \exp\left(-\frac{\lambda_{\text{ent}}}{2} \max(0, H_\phi(\mathbf{x}_{t-1}) - H_{\text{target}})^2\right), \quad (8)$$

where  $\lambda_{\text{ent}}$  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:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, H_{\text{target}}) \propto p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \exp\left(-\frac{\lambda_{\text{ent}}}{2} \max(0, H_\phi(\mathbf{x}_{t-1}) - H_{\text{target}})^2\right). \quad (9)$$

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:

$$\mathcal{L}_{\text{ent}} = \max(0, H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}), \quad (10)$$

where  $H_\phi$  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:

$$\nabla_{\hat{\mathbf{x}}_0} \mathcal{L}_{\text{ent}} = \begin{cases} \nabla_{\hat{\mathbf{x}}_0} H_\phi(\hat{\mathbf{x}}_0), & \text{if } H_\phi(\hat{\mathbf{x}}_0) > H_{\text{target}}, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

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  $H_\phi$  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:

$$|H_\phi(\hat{\mathbf{x}}_0) - H_{\text{true}}(\hat{\mathbf{x}}_0)| \leq \epsilon. \quad (12)$$

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

$$|H_\phi(\hat{\mathbf{x}}_0) - H_{\text{target}}| \geq |H_{\text{true}}(\hat{\mathbf{x}}_0) - H_{\text{target}}| - \epsilon. \quad (13)$$

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

Future work could explore jointly training  $H_\phi$  with contrastive or reinforcement signals to further narrow the approximation gap.

## 60 D Complexity Analysis

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

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

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

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

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

72 **Entropy Modules.** BADiff introduces three lightweight modules: the entropy embedding MLP, the  
73 entropy predictor  $H_\phi$ , and the stopping policy network  $f_\phi$ .

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

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

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

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

## 88 E Implementation Details

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

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

94 **Optimizer and scheduler.** We adopt the Adam optimizer [3] with default coefficients  $\beta_1=0.9$ ,  
95  $\beta_2=0.999$ . The learning rate is set to  $1 \times 10^{-4}$  and kept constant throughout training. No learning  
96 rate decay or warmup is applied. Gradient clipping is used with a max norm of 1.0.

97 **Sampling parameters.** For DDPM, we use a 1,000-step linear beta schedule as in the original  
98 DDPM implementation [2]. For LDM, we follow [4] and use 200 steps with cosine noise scheduling  
99 in the latent space. At inference time, the sampling process is governed by the entropy-aware stopping  
100 policy, which dynamically terminates early based on the predicted bitrate.

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

106 **Codec baselines.** For BPG, we use the official reference implementation compiled with  
 107 libbpg-0.9.8. For learned image compression (LIC), we adopt the pre-trained model of  
 108 Cheng2020 [1] from CompressAI. BPG codec experiments all run on CPU and LIC experiments all  
 109 run on GPU, with the output bitrate measured post-compression in bits-per-pixel (bpp).

## 110 F Model Architectures

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

113 **UNet Backbone.** We use a modified version of the standard UNet architecture as introduced in  
 114 DDPM [2]. The configuration is as follows:

- 115 • Downsampling path: 4 resolution levels with channel counts [128, 256, 512, 512]. Each level  
 116 consists of two residual blocks (with GroupNorm + SiLU) followed by a downsampling layer  
 117 (stride-2 convolution).
- 118 • Bottleneck: 2 residual blocks with 512 channels and an attention layer at the lowest resolution  
 119 ( $8 \times 8$  for CIFAR-10).
- 120 • Upsampling path: mirrors the downsampling path with learned upsampling (transposed conv),  
 121 residual blocks, and attention at the second-lowest resolution.
- 122 • Timestep + Entropy Conditioning: the diffusion timestep and entropy target are embedded sepa-  
 123 rately (see below) and added to each residual block via FiLM modulation.

124 **Entropy Embedding MLP.** The target entropy  $H_{\text{target}}$  is a scalar projected into a high-dimensional  
 125 embedding via:

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

131 **Stopping Policy Network.** The policy module  $f_\phi$  is a compact classifier:

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

136 **Differentiable Entropy Predictor  $E_\phi$ .** The entropy estimator is CNN-based with soft binning:

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

## 143 G Hyperparameters

144 Table 1 lists the main hyperparameters used throughout training for all BADiff models, unless  
 145 otherwise stated. We adopt the default settings from DDPM [2] for the optimizer and noise schedule,  
 146 and perform minimal tuning to isolate the effects of our proposed components. The entropy-related  
 147 weights  $\lambda_{\text{ent}}$ ,  $\lambda_{\text{cal}}$ , and  $\lambda_{\text{stop}}$  are set via coarse grid search on CIFAR-10 validation splits.

Table 1: Key hyperparameters.

Hyperparameter	Value
Learning rate	$1 \times 10^{-4}$
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 Visual Comparisons

We provide detailed visual comparisons of images sampled from CelebA-HQ dataset in this section to qualitatively assess the effectiveness of BADiff in generating high-fidelity images under varying bandwidth constraints. These comparisons complement the quantitative results presented in the main paper and offer intuitive insights into the perceptual improvements introduced by BADiff.

Specifically, we present side-by-side generations from the following baseline methods: DDPM + BPG, DDPM + LIC, PNDM + LIC and BADiff. BADiff produces images with sharper edges, fewer artifacts, and more consistent texture than other baselines. Under **low bitrate** settings, cascaded DDPM + BPG and DDPM + LIC often suffer from blocking artifacts, oversmoothing, or ringing due to aggressive compression. Fast solvers like PNDM generate slightly more structured images, but their perceptual quality degrades when the bitrate constraint is tight.

## I 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.

## References

- [1] 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.
- [2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.
- [3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.
- [4] 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.



Figure 1: Visual comparison of different generation methods under low-bit constraint.