# DiffusionBlocks: Blockwise Training for Generative Models via Score-Based Diffusion

**Makoto Shing** [1] **Takuya Akiba** [1]

## Abstract

Training large neural networks with end-to-end backpropagation creates significant memory bottlenecks, limiting accessibility to state-of-the-art AI research. We propose *DiffusionBlocks*, a novel training framework that interprets neural network blocks as performing denoising operations in a continuous-time diffusion process. By partitioning the network into independently trainable blocks and optimizing noise level assignments based on equal cumulative probability mass, our approach achieves significant memory efficiency while maintaining competitive performance compared to traditional backpropagation in generative tasks. Experiments on image generation and language modeling tasks demonstrate memory reduction proportional to the number of blocks while achieving superior performance. DiffusionBlocks provides a promising pathway for democratizing access to large-scale neural network training with limited computational resources.

## 1. Introduction

As neural networks grow following established scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), they become increasingly inaccessible to much of the research community. Training models with hundreds of billions of parameters requires computational resources available only to select institutions, threatening to concentrate AI advancement within well-resourced organizations.

The fundamental bottleneck lies in end-to-end backpropagation (Rumelhart et al., 1986; He et al., 2016), which requires storing intermediate activations across the entire network, resulting in prohibitive memory demands for large models. This memory bottleneck is particularly critical for generative AI applications, where large-scale models are essential for high-quality generation.

Previous layerwise training approaches (Hinton, 2022; Bengio et al., 2006; Nøkland & Eidnes, 2019; Belilovsky et al., 2019; Siddiqui et al., 2024) have underperformed compared to end-to-end backpropagation, primarily because they lack principled mechanisms to coordinate information flow between independently trained layers and struggle to balance parameter allocation effectively. Moreover, these approaches have been predominantly evaluated on image classification tasks, with limited exploration of generative modeling applications.

Meanwhile, diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021) have revolutionized generative modeling through their mathematically principled approach to distribution transformation. Recent advances in network conditioning (Karras et al., 2022) and sampling efficiency (Lu et al., 2022; 2023; Zhao et al., 2023) have established diffusion models as state-of-the-art across multiple domains.

We propose *DiffusionBlocks*, a framework that reconceptualizes neural network training by interpreting network blocks as implementing discretized steps of a continuous-time reverse diffusion process. Our key innovation is a principled mapping between network blocks and noise-level ranges based on equal cumulative probability mass, ensuring each block confronts an equally challenging learning problem. This approach enables independent block training without requiring gradient communication between blocks. Through experiments on image generation and language modeling tasks, we demonstrate that DiffusionBlocks reduces memory requirements proportionally to the number of blocks while achieving competitive or superior performance. Our primary contributions are:

- A diffusion-inspired blockwise training framework achieving true block independence in continuous time, where each block can be trained without requiring gradients from other blocks.
- An equi-probability partitioning strategy that optimally allocates learning difficulty across blocks based on

[1]Sakana AI. Correspondence to: Makoto Shing <mksh-ing@sakana.ai>, Takuya Akiba <takiba@sakana.ai>.

*Proceedings of the $42^{nd}$ International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. 

cumulative probability mass, ensuring balanced parameter utilization.

- Comprehensive empirical validation demonstrating $B$-fold memory reduction (with $B$ blocks) and improved performance on both image generation and language modeling tasks.

## 2. Preliminaries

### 2.1. Score-Based Diffusion Models

Let $\mathbf{z}_0 \in \mathbb{R}^d \sim p_{\text{data}}$ denote a clean data sample. Following the Variance-Exploding (VE) formulation (Song et al., 2021; Karras et al., 2022), we perturb $\mathbf{z}_0$ with Gaussian noise whose standard deviation $\sigma(t)$ increases monotonically with the (continuous) time variable $t \in [0, 1]$:

$$\mathbf{z}_t = \mathbf{z}_0 + \sigma(t)\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{1}$$

This gives $\mathbf{z}_t \sim \mathcal{N}(\mathbf{z}_0, \sigma(t)^2\mathbf{I}) = p_t(\mathbf{z}_t|\mathbf{z}_0)$ with marginal distribution $p_t(\mathbf{z}_t) = \int p_{\text{data}}(\mathbf{z}_0)p_t(\mathbf{z}_t|\mathbf{z}_0)\mathrm{d}\mathbf{z}_0$.

The continuous-time formulation of this process is described by a stochastic differential equation (SDE):

$$\mathrm{d}\mathbf{z}_t = \sqrt{\frac{\mathrm{d}\sigma(t)^2}{\mathrm{d}t}}\mathrm{d}\mathbf{w}, \quad t \in [0, 1] \tag{2}$$

where $\mathbf{w}$ is a standard Wiener process.

For generating samples, we employ the Probability Flow ODE (PF-ODE), which shares the same marginal distributions as the SDE but follows deterministic trajectories:

$$\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{z}} \log p_t(\mathbf{z}_t) \tag{3}$$

where $\dot{\sigma}(t) = \frac{\mathrm{d}\sigma(t)}{\mathrm{d}t}$ and $\nabla_{\mathbf{z}} \log p_t(\mathbf{z}_t)$ is the score of the density $p_t(\mathbf{z}_t)$. Following Karras et al. (2022), we can eliminate the abstract time variable by parameterizing directly in terms of noise levels. Setting $\sigma(t) = t$, the PF-ODE simplifies to:

$$\frac{\mathrm{d}\mathbf{z}_\sigma}{\mathrm{d}\sigma} = -\sigma\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma). \tag{4}$$

To estimate this score function, we parameterize it using a neural network. We leverage the relation $\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma) \approx \frac{\mathbf{z}_0 - \mathbf{z}_\sigma}{\sigma^2}$ (Robbins, 1992) to approximate the score in terms of a denoiser $D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma)$ that predicts the clean data:

$$\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma) \approx \frac{D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma) - \mathbf{z}_\sigma}{\sigma^2} \tag{5}$$

The denoiser is trained using a weighted L2 loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}, p_\sigma, \mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(\sigma)\|D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma) - \mathbf{z}_0\|_2^2\right] \tag{6}$$

where $w(\sigma)$ is a weighting function and $p_\sigma$ is the distribution from which noise levels are sampled during training.

### 2.2. Neural Network Block Structure

Consider a deep neural network with $L$ layers, parameterized by $\boldsymbol{\theta} = (\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L)$. Traditional end-to-end training processes the input $\mathbf{x} \in \mathcal{X}$ through the network to produce an output $\hat{\mathbf{y}} \in \mathcal{Y}$ as follows:

$$\mathbf{z}^{(0)} = f_0(\mathbf{x}; \boldsymbol{\theta}_0) \quad \text{(input embedding)} \tag{7}$$

$$\mathbf{z}^{(l)} = f_l(\mathbf{z}^{(l-1)}; \boldsymbol{\theta}_l), \quad l \in [L] \tag{8}$$

$$\hat{\mathbf{y}} = f_{L+1}(\mathbf{z}^{(L)}; \boldsymbol{\theta}_{L+1}) \quad \text{(output projection)} \tag{9}$$

A loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ is computed between the predicted output $\hat{\mathbf{y}}$ and target $\mathbf{y}$. Backpropagation calculates gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ by propagating error signals backward through the entire network, requiring storage of all intermediate activations $\{\mathbf{z}^{(l)}\}_{l=0}^{L}$. This memory requirement scales with network depth and batch size, creating a bottleneck for large-scale models.

When partitioning a network into blocks, we group consecutive layers together to form $B$ blocks, where each block $i \in [B]$ consists of multiple layers and is parameterized by $\boldsymbol{\theta}_i$. In traditional blockwise approaches, defining appropriate training objectives for each block remains challenging, as these blocks must coordinate to accomplish the overall task without end-to-end supervision.

### 2.3. Residual Connections as Euler Steps of the Reverse Diffusion Process

The connection between residual networks and continuous-time ODEs has been established in prior work (Haber & Ruthotto, 2017; Chen et al., 2018), where residual updates $\mathbf{z}^{(l)} = \mathbf{z}^{(l-1)} + g_{\boldsymbol{\theta}_l}(\mathbf{z}^{(l-1)})$ are shown to correspond to Euler discretizations of ODEs. We extend this perspective to our blockwise diffusion framework.

In diffusion models, the forward process adds noise progressively, while the reverse process removes it to generate data. This reverse process can be formulated either as a stochastic differential equation (SDE) or its deterministic counterpart, PF-ODE (Eq. (3)). While both formulations share the same marginal distributions, we focus on the PF-ODE due to its deterministic nature, which aligns naturally with the deterministic forward pass of neural networks.

Applying Euler discretization to Eq. (4) with noise levels $\sigma_0 > \sigma_1 > \cdots > \sigma_N$ yields:

$$\mathbf{z}_{\sigma_l} = \mathbf{z}_{\sigma_{l-1}} - \Delta\sigma_l \cdot \sigma_{l-1}\nabla_{\mathbf{z}} \log p_{\sigma_{l-1}}(\mathbf{z}_{\sigma_{l-1}}) \tag{10}$$

$$= \mathbf{z}_{\sigma_{l-1}} + \underbrace{\frac{\Delta\sigma_l}{\sigma_{l-1}}\left(\mathbf{z}_{\sigma_{l-1}} - D_{\boldsymbol{\theta}}(\mathbf{z}_{\sigma_{l-1}}, \sigma_{l-1})\right)}_{=:g_{\boldsymbol{\theta}_l}(\mathbf{z}_{\sigma_{l-1}})}, \tag{11}$$

where $\Delta\sigma_l = \sigma_{l-1} - \sigma_l > 0$ and we used the score approximation from Eq. (5).

This reveals that each denoising step naturally takes the form of a residual update $\mathbf{z}_{\sigma_l} = \mathbf{z}_{\sigma_{l-1}} + g_{\boldsymbol{\theta}_l}(\mathbf{z}_{\sigma_{l-1}})$, matching the structure of modern neural architectures with skip connections. This mathematical correspondence explains why skip connections are essential for our framework: they naturally implement the Euler discretization of the reverse diffusion process. Architectures with residual connections—such as ResNets (He et al., 2016), U-Nets (Ronneberger et al., 2015), and transformer blocks with residual paths (Vaswani et al., 2017)—are therefore ideally suited for our approach. Architectures without skip connections would require implicit ODE solvers, which are computationally more complex and less compatible with our blockwise training approach. Therefore, we restrict our framework to architectures with explicit residual connections, ensuring compatibility between the network structure and the underlying continuous-time diffusion process.

## 3. Method

We now present *DiffusionBlocks*, our approach for training neural networks without end-to-end backpropagation. Our key insight is interpreting neural networks as implementing discretized steps of a continuous-time score-based diffusion process. This perspective enables training individual blocks independently while maintaining network-wide coherence through a shared mathematical framework .

### 3.1. Diffusion-Based Blockwise Training Framework

Traditional neural networks transform input $\mathbf{x}$ through hidden layers to output $\hat{\mathbf{y}}$. We reconceptualize this as a reverse diffusion process: the input corresponds to noise ($\mathbf{z}_{\sigma_{\max}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$), and the output to clean data ($\mathbf{z}_0 \sim p_{\text{data}}$). Each network block then performs partial denoising within a specific noise range.

Given a neural network with $L$ layers, we partition it into $B$ blocks, where each block contains one or more consecutive layers. Instead of training the entire network end-to-end, each block is assigned responsibility for a specific range of noise levels in the diffusion process. Specifically, Block $i$ handles the noise level range $[\sigma_i, \sigma_{i+1}]$, where $i \in \{0, 1, ..., B-1\}$ and $\sigma_0 = \sigma_{\max}$ and $\sigma_B = \sigma_{\min}$ (typically set to a small positive value or zero).

During training, for a block $i$ handling noise level range $[\sigma_i, \sigma_{i+1}]$, we train the corresponding denoiser $D_{\boldsymbol{\theta}_i}(\mathbf{z}_\sigma, \sigma, \mathbf{x})$ to predict the clean target:

$$\mathcal{L}(\boldsymbol{\theta}_i) = \mathbb{E}_{p_{\text{data}}, p_\sigma^{(i)}, \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ w(\sigma) \| D_{\boldsymbol{\theta}_i}(\mathbf{z}_\sigma, \sigma, \mathbf{x}) - \mathbf{y} \|_2^2 \right] \tag{12}$$

where $p_\sigma^{(i)}$ is the distribution of noise levels specifically for block $i$, defined by restricting the global noise distribution to the range $[\sigma_i, \sigma_{i+1}]$. For tasks like language modeling, we replace the $L_2$ loss with cross-entropy after appropriate normalization.

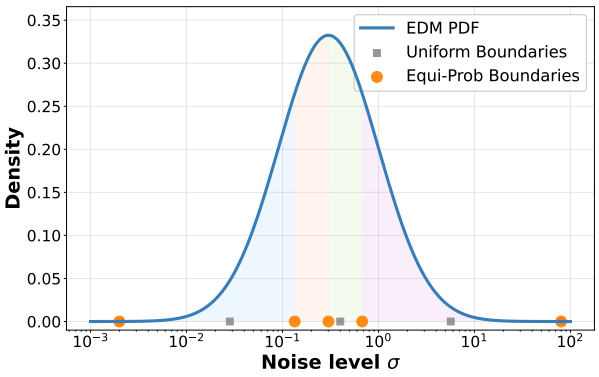


*Figure 1.* **Block partitioning strategies for noise level assignment.** Colored regions represent individual blocks under our equi-probability partitioning, where each block handles equal cumulative probability mass from the EDM log-normal distribution (blue curve). Orange circles show our equi-probability boundaries that concentrate in the challenging intermediate noise region, while gray squares show uniform boundaries (equal intervals in log-space) for comparison. This strategy ensures balanced learning difficulty across blocks.

Each block-specific denoiser includes input embedding layers, neural network blocks, and output embedding components, making blocks truly independent.

This block independence is the key to our memory efficiency—during training, we only need to store activations for a single block rather than the entire network. Specifically, our approach requires storage of activations for $L/B$ layers instead of all $L$ layers needed by end-to-end backpropagation, resulting in approximately $B$-fold memory reduction during training.

### 3.2. Equi-Probability Block Partitioning

A critical innovation in our approach is how we partition the noise levels among blocks. Following Karras et al. (2022), we recognize that different noise levels present varying degrees of difficulty for the denoising task. The intermediate noise range tends to be most challenging and impactful for learning, while very low or high noise levels are comparatively simpler. To optimize parameter utilization, we partition the range of noise levels $[\sigma_{\min}, \sigma_{\max}]$ into $B$ blocks such that each block handles an equal amount of cumulative probability under the noise distribution:

$$\sigma_i = \exp\left(P_{\text{mean}} + P_{\text{std}} \cdot \Phi^{-1}(p_i)\right) \tag{13}$$

where $p_i = \text{CDF}_{\min} + \frac{i}{B} \cdot (\text{CDF}_{\max} - \text{CDF}_{\min})$ represents the target cumulative probability for block $i$, $\Phi^{-1}$ is the inverse CDF of the standard normal distribution, and $\text{CDF}_{\min}$ and $\text{CDF}_{\max}$ are the CDFs corresponding to $\sigma_{\min}$

and $\sigma_{\max}$ respectively. This partitioning ensures that each block handles an equal amount of cumulative probability mass:

$$\int_{\sigma_i}^{\sigma_{i+1}} p_\sigma(\sigma) d\sigma = \frac{1}{B}. \tag{14}$$

Figure 1 illustrates how our approach allocates block boundaries to ensure equal cumulative probability across the noise level distribution. This strategy ensures that each block contributes equally to the overall learning task, optimizing parameter utilization. In contrast, naive uniform partitioning (e.g., dividing $[\sigma_{\min}, \sigma_{\max}]$ into equal intervals) would allocate too many parameters to easy regions while underserving challenging noise levels.

### 3.3. Controlled Block Overlap

To mitigate potential discontinuities between blocks, we introduce a controlled overlap between adjacent noise level ranges. For a block $i$ responsible for noise range $[\sigma_i, \sigma_{i+1}]$, we expand the training range to:

$$[\sigma_i/\alpha, \sigma_{i+1} \cdot \alpha], \tag{15}$$

where $\alpha := (\sigma_{i+1}/\sigma_i)^\gamma$ and $\gamma$ is the overlap coefficient. This controlled overlap ensures smoother transitions during inference by allowing each block to learn from samples slightly outside its primary range of responsibility. In all our experiments, we use $\gamma = 0.1$, which provides an effective balance between block independence and transition smoothness.

### 3.4. Implementation Details

Our implementation follows the EDM framework (Karras et al., 2022) including the preconditioning strategy. Detailed training and inference algorithms are provided in Appendix C.

## 4. Experiments

We evaluate DiffusionBlocks on image generation and language modeling tasks, demonstrating superior or comparable performance to end-to-end backpropagation while training with significantly reduced memory requirements. We also analyze key components of our framework.

### 4.1. Image Generation

**Experimental Setup.** We evaluate our method on CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009) at 256×256 resolution using Diffusion Transformer (DiT) architectures (Peebles & Xie, 2023). We use DiT-S with 12 layers and DiT-L with 24 layers, and partition them into 4 blocks. All models are trained with classifier-free guidance (Ho & Salimans, 2022), dropping labels with probability 0.1. For ImageNet, we follow Peebles & Xie (2023) compressing images using a pre-trained VAE. Detailed hyperparameters and implementation specifics are provided in Appendix D.1.

*Table 1.* **Image generation results comparing FID scores (lower is better).** DiffusionBlocks achieves superior quality while training each block independently.

| Method | CIFAR-10 | ImageNet-256 |
|---|---|---|
| End-to-End BackProp | 41.87 | 16.62 |
| **Ours** | **41.39** | **15.55** |

**Results.** Table 1 compares our approach against end-to-end backpropagation, showing that DiffusionBlocks achieves better FID scores on both datasets. By training only one block at a time and optimizing each block independently, our approach reduces memory requirements during training by a factor of $B$ ($B = 4$ in our experiments)—backpropagation needs to be performed only through the active block rather than the entire network. Figure 2 shows examples of generated images from our model on the CIFAR-10 dataset.

Additionally, a significant advantage of our approach is faster inference: while the baseline model requires forwarding through all layers for each diffusion step, our method only needs to use the relevant block. This results in approximately 3× faster generation time.

### 4.2. Language Modeling

**Experimental Setup.** For language modeling, we use The One Billion Words Benchmark (LM1B) (Chelba et al., 2014) with a Llama-style architecture (Touvron et al., 2023) comprising 12 transformer layers partitioned into 4 blocks. We implement specialized attention mechanisms (Arriola et al., 2025) to handle autoregressive dependencies while maintaining diffusion-based denoising capabilities.

We evaluate models using MAUVE score (Pillutla et al., 2021), following the conditional generation protocol established by SEDD (Lou et al., 2024). Detailed hyperparameters and implementation specifics are provided in Appendix D.2.

**Results.** Table 2 shows that our method achieves superior MAUVE scores compared to end-to-end backpropagation, despite only requiring backpropagation through one block at a time during training. This demonstrates that our blockwise training approach can effectively learn high-quality text generation while maintaining significant memory efficiency.

*Table 2.* **Language modeling results comparing MAUVE scores (higher is better).** Our method achieves superior performance compared to end-to-end backpropagation.

| Method | MAUVE (↑) |
|---|---|
| End-to-End BackProp | 0.595 |
| **Ours** | **0.779** |

*Table 3.* **Effect of block partitioning strategy on CIFAR-10.** Our equi-probability partitioning outperforms uniform partitioning by allocating blocks based on learning difficulty.

| Partitioning Strategy | FID (↓) |
|---|---|
| Uniform | 68.06 |
| Equi-Probability | **45.50** |

### 4.3. Ablation Studies

We perform ablation studies on CIFAR-10 to analyze the importance of key components in our framework. All experiments use the same network architecture and hyperparameters unless otherwise specified.

**Block Partitioning Strategy.** We compare our equi-probability partitioning strategy against uniform partitioning across noise levels. We disabled the block overlap in Section 3.3 to isolate the effectiveness of our partitioning strategy. As shown in Table 3, our approach outperforms uniform partitioning, achieving an FID of 45.50 compared to 68.06. While this improvement is meaningful, the difference highlights that both strategies can achieve reasonable performance, with our equi-probability approach providing a consistent advantage. This supports our hypothesis that allocating block capacity based on the intrinsic difficulty of denoising at different noise levels (as visualized in Figure 1) contributes to more effective parameter utilization. The uniform strategy, while functional, appears to be less optimal as it allocates equal capacity across all noise regions rather than concentrating resources where learning is most challenging.

**Effect of Block Overlap.** To evaluate the importance of controlled overlap between blocks, we varied the overlap coefficient $\gamma$ from 0 (no overlap) to 0.2 (substantial overlap). Table 4 demonstrates that controlled overlap significantly improves performance compared to strict block boundaries. Without overlap ($\gamma = 0$), FID degrades to 45.50 due to discontinuities between independently trained blocks. Performance improves as we introduce modest overlap, reaching optimal results at $\gamma = 0.1$ (FID 41.39). However, excessive overlap ($\gamma \geq 0.15$) begins to degrade performance, with $\gamma = 0.2$ producing significantly worse results (FID 56.69), likely due to conflicting learning objectives when blocks have substantial overlap in their training regions. These results confirm that $\gamma = 0.1$ provides an effective balance between maintaining block independence and ensuring smooth transitions during inference.

*Table 4.* **Effect of block overlap on CIFAR-10.** Controlled overlap between adjacent blocks significantly improves performance, with $\gamma = 0.1$ providing the optimal balance between block independence and transition smoothness.

| Overlap Coefficient $\gamma$ | FID (↓) |
|---|---|
| $\gamma = 0.00$ | 45.50 |
| $\gamma = 0.05$ | 42.98 |
| $\gamma = 0.10$ | **41.39** |
| $\gamma = 0.15$ | 42.84 |
| $\gamma = 0.20$ | 56.69 |

*Table 5.* **Effect of block count on CIFAR-10.** Fewer blocks achieve better FID but require more layers per diffusion step (L/S), creating a trade-off between quality and efficiency. Note that L/S=$L/B$, where $L$ is the total number of layers (12) and $B$ is the number of blocks.

| Number of Blocks | FID (↓) | L/S (↓) | Relative Speed |
|---|---|---|---|
| $B = 1$ (End-to-End BackProp) | 41.87 | 12 | 1.0× |
| $B = 2$ | 38.58 | 6 | 2.0× |
| $B = 3$ | 41.39 | 4 | 3.0× |
| $B = 4$ | 41.39 | 3 | 4.0× |
| $B = 6$ | 53.74 | 2 | 6.0× |

**Effect of Block Count.** We investigate how performance varies with different numbers of blocks while keeping the total network depth constant (12 layers). Table 5 reveals a clear trade-off between FID score and computational efficiency. Using fewer blocks yields better FID scores due to larger block capacity—$B = 2$ achieves the best FID (38.58) but requires processing 6 layers per forward pass. As the number of blocks increases, inference becomes more efficient: $B = 4$ processes only 3 layers per step (4× faster than end-to-end) while maintaining reasonable FID (41.39), and $B = 6$ achieves 6× speedup at the cost of degraded performance (FID 53.74). The results suggest that $B = 3$ or $B = 4$ provide good balance points, offering substantial efficiency gains while preserving competitive generation quality. Beyond $B = 6$, individual blocks become too small (2 layers each) to perform effective denoising, leading to significant quality degradation. This analysis enables practitioners to choose the appropriate block count based on their specific quality requirements and computational constraints.

## 5. Related Work

**Diffusion Models and Score-Based Generation.** Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) and score-based generative models (Song & Ermon, 2019; 2020; Song et al., 2021) have emerged as powerful frameworks for generative modeling. These models define processes that gradually transform simple distributions into complex ones through sequences of denoising steps. Recent advances in network conditioning (Karras et al., 2022), sampling efficiency (Lu et al., 2022; 2023; Zhao et al., 2023), and architectural improvements (Rombach et al., 2022; Peebles & Xie, 2023) have established diffusion models as state-of-the-art across various generative tasks. Our work leverages these mathematical foundations for neural network training, interpreting layer transformations through the lens of continuous-time diffusion processes.

**Layer/Block-wise Training Methods.** Various approaches have been proposed to train neural networks without end-to-end backpropagation. *Synthetic Gradients* (Jaderberg et al., 2017) enables decoupled neural interfaces by predicting gradients locally, while biologically-motivated methods include *Feedback Alignment* (Lillicrap et al., 2016), the *Forward-Forward algorithm* (Hinton, 2022), and *Target Propagation* (Lee et al., 2015). Additional approaches include local learning methods (Nøkland & Eidnes, 2019; Belilovsky et al., 2019), greedy layer-wise pretraining (Bengio et al., 2006), and *Blockwise Self-Supervised Learning* (Siddiqui et al., 2024). However, these methods face two fundamental limitations: they lack principled theoretical foundations for coordinating information flow between independently trained components, and have demonstrated limited effectiveness on generative modeling tasks where maintaining coherent probabilistic modeling across components remains challenging. DiffusionBlocks addresses both limitations through the mathematical rigor of continuous-time diffusion theory, where each block's denoising objective naturally aligns with the global generative goal.

**Memory-Efficient Implicit Depth Models.** *Neural ODEs* (Chen et al., 2018) parameterize network dynamics as continuous-time differential equations, using the adjoint sensitivity method to achieve constant memory backpropagation through time. *Deep Equilibrium Models* (DEQs) (Bai et al., 2019) represent another memory-efficient paradigm, directly solving for fixed points of implicit layers using root-finding and implicit differentiation, effectively creating infinite-depth networks with constant memory. While both approaches achieve memory efficiency through implicit computation, they fundamentally differ from our method: Neural ODEs still require end-to-end backpropagation through a single monolithic network, and DEQs focus on equilibrium computation rather than generative modeling. In contrast, DiffusionBlocks achieves true block independence by partitioning the continuous-time diffusion process into disjoint noise-level ranges, enabling genuinely parallel block training without any inter-block gradient flow.

**Connection to Concurrent Work.** Most closely related to our work is the concurrent *NoProp* framework (Li et al., 2025), which also interprets neural network training through diffusion principles. NoProp's discrete-time formulation (*NoProp-DT*) treats each network layer as a discrete denoising step, achieving memory-efficient training for classification tasks. However, their continuous-time variant (*NoProp-CT*) fundamentally differs from true blockwise training: it employs a single network $\hat{u}_\theta(z_t, x, t)$ that must handle all noise levels $t \in [0, 1]$, requiring end-to-end backpropagation through the entire architecture. This approach more closely resembles Neural ODEs (Chen et al., 2018) than blockwise methods.

Our framework achieves genuine blockwise independence in continuous time by partitioning the noise range $[\sigma_{\min}, \sigma_{\max}]$ into $B$ intervals, with each block $D_{\theta_i}$ independently responsible for its assigned range $[\sigma_i, \sigma_{i+1}]$. This enables $B$-fold memory reduction during training while maintaining the mathematical rigor of continuous-time diffusion. Furthermore, our equi-probability partitioning based on cumulative distribution mass ensures optimal parameter utilization across blocks—a principled approach absent in NoProp's fixed layer-to-timestep mapping. Notably, while NoProp focuses primarily on classification tasks and evaluates against diffusion-inspired baselines, we demonstrate superior performance on generative modeling tasks—image generation and language modeling—where our framework naturally excels, directly comparing against conventional end-to-end backpropagation on established architectures.

## 6. Conclusion

We introduced *DiffusionBlocks*, a novel framework that enables independent neural network block training by interpreting blocks as denoising operations at specific noise levels in a continuous-time diffusion process. Our approach achieves blockwise independence, optimal equi-probability partitioning, and $B$-fold memory reduction with competitive or superior performance on generative modeling tasks.

Experiments on image generation and language modeling demonstrate that DiffusionBlocks outperforms end-to-end backpropagation while requiring only $1/B$ the memory during training. In our experiments with $B = 4$ blocks, this translates to a $4\times$ memory reduction with superior performance, offering a principled pathway for democratizing large-scale neural network training with limited computational resources.

## References


Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2023.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Arriola, M., Sahoo, S. S., Gokaslan, A., Yang, Z., Qi, Z., Han, J., Chiu, J. T., and Kuleshov, V. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to ImageNet. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 583–593. PMLR, 09–15 Jun 2019.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In Schölkopf, B., Platt, J., and Hoffman, T. (eds.), *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *International Conference on Machine Learning*, pp. 41–48. Association for Computing Machinery, 2009.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2014.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, dec 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hinton, G. The Forward-Forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. An empirical analysis of compute-optimal large language model training. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022.

Jaderberg, M., Czarnecki, W., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In Precup, D. and Teh, Y. W. (eds.), *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1627–1635. PMLR, 06–11 Aug 2017.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A. (eds.), *Machine Learning and Knowledge Discovery in*

*Databases*, pp. 498–515, Cham, 2015. Springer International Publishing.

Li, Q., Teh, Y. W., and Pascanu, R. NoProp: Training neural networks without back-propagation or forward-propagation. *arXiv preprint arXiv:2503.24322*, 2025.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7(1):13276, Nov 2016.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Lou, A., Meng, C., and Ermon, S. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.1683*, 2024.

Lu, C., Zhou, Y., Bao, F., Chen, J., LI, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 5775–5787. Curran Associates, Inc., 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2023.

Neyshabur, B. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

Nøkland, A. and Eidnes, L. H. Training neural networks with local error signals. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4839–4850. PMLR, 09–15 Jun 2019.

Parmar, G., Zhang, R., and Zhu, J.-Y. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11410–11420, 2022.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Pillutla, K., Swayamdipta, S., Zellers, R., Thickstun, J., Welleck, S., Choi, Y., and Harchaoui, Z. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4816–4828. Curran Associates, Inc., 2021.

Robbins, H. E. *An Empirical Bayes Approach to Statistics*, pp. 388–394. Springer New York, 1992.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F. (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241. Springer International Publishing, 2015.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323:533–536, 1986.

Shi, Y., De Bortoli, V., Campbell, A., and Doucet, A. Diffusion schrödinger bridge matching. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 62183–62223. Curran Associates, Inc., 2023.

Siddiqui, S., Krueger, D., LeCun, Y., and Deny, S. Blockwise self-supervised learning at scale. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In Bach, F. and Blei, D. (eds.), *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12438–12448. Curran Associates, Inc., 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 49842–49869. Curran Associates, Inc., 2023.

# A. Limitations and Future Directions

**Limitations.** Our approach has two fundamental limitations. First, our framework requires architectures with explicit residual connections, as these naturally implement the Euler discretization of the reverse diffusion process (Section 2.3). This restricts applicability to ResNet-style, U-Net, and Transformer architectures, excluding feedforward networks and other non-residual designs. While most modern architectures incorporate residual connections, this constraint prevents exploration of alternative architectural paradigms. Second, for autoregressive tasks like language modeling, our approach requires multiple diffusion steps per token, resulting in $O(KM)$ forward passes for generating $K$ tokens with $M$ diffusion steps. This multiplicative overhead compared to standard $O(K)$ autoregressive generation may be prohibitive for real-time applications, despite quality improvements through test-time scaling.

**Future Directions.** Several promising directions emerge from this work. First, while we focus on the VE formulation, our DiffusionBlocks framework is fundamentally applicable to other diffusion formulations, including Variance Preserving (VP) (Song et al., 2021), flow matching (Lipman et al., 2023; Liu et al., 2023), stochastic interpolants (Albergo & Vanden-Eijnden, 2023; Albergo et al., 2023), and bridge matching (Shi et al., 2023).

Second, theoretical analysis of why DiffusionBlocks outperforms end-to-end backpropagation warrants investigation. We hypothesize this improvement stems from two mechanisms. First, our framework may induce structured constraints on the optimization process through the diffusion formulation, potentially providing a form of implicit regularization (Neyshabur, 2017) that guides parameter updates toward more principled solutions. Second, our equi-probability partitioning explicitly allocates computational resources based on the empirical difficulty distribution of denoising tasks (Bengio et al., 2009), which may lead to more efficient parameter utilization compared to the implicit allocation mechanisms in end-to-end training.

Third, adapting recent advances in fast diffusion sampling (Lu et al., 2023; Zhao et al., 2023) could significantly reduce inference costs for language modeling while preserving test-time scaling benefits.

Fourth, exploring block-parallel generation techniques (Arriola et al., 2025) could enable simultaneous token generation, addressing sequential bottlenecks in autoregressive tasks.

Finally, investigating optimal architecture designs for different noise level ranges and extending our framework to multimodal tasks with efficient architectures like Mixture-of-Experts could broaden its applicability and impact.

# B. Mathematical Background

## B.1. Variance Exploding Diffusion Models

In the Variance Exploding (VE) formulation (Ho et al., 2020; Song et al., 2021), a perturbed sample at noise level $\sigma$ is defined as:

$$\mathbf{z}_\sigma = \mathbf{z}_0 + \sigma\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{16}$$

where $\mathbf{z}_\sigma \sim \mathcal{N}(\mathbf{z}_0, \sigma^2\mathbf{I}) =: p_\sigma(\mathbf{z}_\sigma|\mathbf{z}_0)$ with marginal distribution $p_\sigma(\mathbf{z}_\sigma) = \int p_{\text{data}}(\mathbf{z}_0)p_\sigma(\mathbf{z}_\sigma \mid \mathbf{z}_0)\mathrm{d}\mathbf{z}_0$.

Following Karras et al. (2022), we parameterize the continuous diffusion process directly in terms of noise levels $\sigma$, eliminating the abstract time variable. The forward SDE becomes:

$$\mathrm{d}\mathbf{z}_\sigma = \sqrt{2\sigma}\mathrm{d}\mathbf{w}, \tag{17}$$

where $\mathbf{w}$ is a standard Wiener process. The corresponding reverse SDE is:

$$\mathrm{d}\mathbf{z}_\sigma = 2\sigma\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma)\mathrm{d}\sigma + \sqrt{2\sigma}\mathrm{d}\bar{\mathbf{w}}, \tag{18}$$

where $\bar{\mathbf{w}}$ is a standard Wiener process in reverse direction (from high to low noise levels).

## B.2. Probability Flow ODE

The Probability Flow ODE (PF-ODE) is a deterministic process that shares the same marginal distributions as the stochastic diffusion process. In our noise-level parameterization:

$$\frac{\mathrm{d}\mathbf{z}_\sigma}{\mathrm{d}\sigma} = -\sigma\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma). \tag{19}$$

This formulation directly connects noise levels to the dynamics of the diffusion process, making it natural for our blockwise approach where each block handles a specific noise level range.

### B.3. Score Estimation and Denoising Score Matching

Our goal is to sample $\mathbf{z}_0 \sim p_{\text{data}}$ from $\mathbf{z}_{\sigma_{\max}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ by the reverse process. The score $\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma)$ is approximated using a neural network.

We leverage the relation $\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma \mid \mathbf{z}_0) = \frac{\mathbf{z}_0 - \mathbf{z}_\sigma}{\sigma^2}$ (Robbins, 1992) to parameterize the score in terms of a denoiser $D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma)$ that predicts the clean data:

$$\nabla_{\mathbf{z}} \log p_\sigma(\mathbf{z}_\sigma) \approx \frac{D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma) - \mathbf{z}_\sigma}{\sigma^2}. \tag{20}$$

The denoiser is trained using a weighted L2 loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}_0 \sim p_{\text{data}}, \sigma \sim p_\sigma, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[ w(\sigma) \| D_{\boldsymbol{\theta}}(\mathbf{z}_\sigma, \sigma) - \mathbf{z}_0 \|_2^2 \right], \tag{21}$$

where $\mathbf{z}_\sigma = \mathbf{z}_0 + \sigma \boldsymbol{\epsilon}$ and $w(\sigma) = \frac{\sigma^2 + \sigma_{\text{data}}^2}{(\sigma \cdot \sigma_{\text{data}})^2}$ following Karras et al. (2022).

### B.4. Noise Level Scheduling

The distribution of noise levels $p_\sigma$ significantly impacts training efficiency and generation quality. Following Karras et al. (2022), we use a log-normal distribution:

$$\log(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2). \tag{22}$$

This distribution concentrates probability mass in intermediate noise regions, which empirically contribute most to learning quality. Very low noise levels result in trivial denoising tasks, while very high noise levels destroy all meaningful information.

## C. Algorithmic Details

**Algorithm 1** DiffusionBlocks Training

**Require:** Dataset $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^N$, Number of blocks $B$, Noise level range $[\sigma_{\min}, \sigma_{\max}]$, Log-normal parameters $P_{\text{mean}}, P_{\text{std}}$

Compute block boundaries $\{\sigma_0, \sigma_1, \ldots, \sigma_B\}$ using Equation (13)
Initialize parameters $\{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{B-1}\}$
**while** not converged **do**
  Sample $i \sim \text{Uniform}(0, B-1)$ {Select a block randomly}
  Sample $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ {Sample a data point}
  Sample $\sigma$ from $[\sigma_i, \sigma_{i+1}]$ according to $p_\sigma^{(i)}$ {Sample noise level for block $i$}
  Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ {Sample noise}
  $\mathbf{z}_\sigma \leftarrow \mathbf{y} + \sigma \boldsymbol{\epsilon}$ {Create noisy target}
  **if** image generation task **then**
    $\mathcal{L} \leftarrow w(\sigma) \| D_{\boldsymbol{\theta}_i}(\mathbf{z}_\sigma, \sigma, \mathbf{x}) - \mathbf{y} \|_2^2$ {Compute L2 loss}
  **else if** language modeling task **then**
    $\mathcal{L} \leftarrow w(\sigma) \cdot \text{CrossEntropy}(\text{Normalize}(D_{\boldsymbol{\theta}_i}(\mathbf{z}_\sigma, \sigma, \mathbf{x})), \mathbf{y})$ {Compute CE loss}
  **end if**
  Update $\boldsymbol{\theta}_i$ to minimize $\mathcal{L}$ {Update only block $i$ parameters}
**end while**

## D. Experimental Details

### D.1. Image Generation Details

**Model Architecture** For our image generation experiments, we employ Diffusion Transformers (DiT) (Peebles & Xie, 2023). We use DiT/S-2 and DiT/L-2 for CIFAR-10 and ImageNet, respectively. We partition them into 4 blocks for DiffusionBlock.

**Algorithm 2** DiffusionBlocks Inference

**Require:** Input $\mathbf{x}$, Trained block parameters $\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{B-1}\}$, Block boundaries $\{\sigma_0, \sigma_1, \ldots, \sigma_B\}$, Number of inference steps $N$, ODE solver (e.g., Euler or Heun)

Generate discretized noise levels $\{\sigma^{(0)}, \sigma^{(1)}, \ldots, \sigma^{(N)}\}$ {EDM discretization}
Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\mathbf{z}^{(0)} \leftarrow \sigma^{(0)}\boldsymbol{\epsilon}$ {Initialize with noise}
**for** $j = 0$ to $N - 1$ **do**
  $\sigma \leftarrow \sigma^{(j)}$ {Current noise level}
  Determine block index $i$ such that $\sigma \in [\sigma_i, \sigma_{i+1})$ {Find responsible block}
  $\mathbf{z}^{(j+1)} \leftarrow \text{ODESolverStep}(\mathbf{z}^{(j)}, \sigma^{(j)}, \sigma^{(j+1)}, D_{\boldsymbol{\theta}_i}, \mathbf{x})$ {Apply ODE solver with block $i$}
**end for**
**return** $\mathbf{z}^{(N)}$

**Training Settings** For both CIFAR-10 and ImageNet, we train class-conditional models with classifier-free guidance (Ho & Salimans, 2022), randomly dropping labels with 10% probability during training. All models are trained with a batch size of 512 for 100 epochs using the AdamW optimizer with a learning rate of 1e-4. For ImageNet, following the latent diffusion approach (Rombach et al., 2022), we first resize images to 256×256 and compress them using a pre-trained VAE from Stability AI's SDXL [1]. The batch size is increased to 1024, with all other settings remaining the same as CIFAR-10.

**Inference Settings** For inference, we use classifier-free guidance with a scale of 2.0 and Euler sampling. We generate 50,000 samples and evaluate them using FID (specifically clean-FID (Parmar et al., 2022)) against the test sets. To reduce the impact of random variation, we compute FID three times in each experiment and report the minimum following Karras et al. (2022).

**Fair Comparison** To ensure fair comparison, we match the total number of layer forwards between our method and the end-to-end backpropagation baseline. For the baseline with $L$ layers trained for $E$ epochs, each layer is updated $L \times E$ times. For our method with $B$ blocks, we train each block for $E$ epochs, resulting in $(L/B) \times E \times B = L \times E$ layer updates.

### D.2. Language Modeling Details

**Model Architecture** For language modeling, we use a Llama-style architecture with Llama 2 tokenizer (Touvron et al., 2023), choosing a model size comparable to DiT/B with 12 transformer layers, 768 hidden dimensions, and 12 attention heads. The model is partitioned into 4 blocks of 3 layers each.

**Training Settings** We train on The One Billion Words Dataset (LM1B) (Chelba et al., 2014) with a batch size of 256 for 10 epochs using the AdamW optimizer with a learning rate of 3e-4 and weight decay of 0.0. We use a context length of 256 tokens.

**Specialized Attention Masks** A key challenge in autoregressive language modeling with diffusion models is maintaining proper conditioning on clean previous tokens while denoising current tokens. To address this, we implement the Vectorized Training approach from Block Diffusion (Siddiqui et al., 2024), which processes both clean and noisy sequences simultaneously by concatenating them and using specialized attention masks. This approach eliminates the need for multiple forward passes and KV cache storage, making training significantly more efficient.

**Evaluation** Following SEDD (Lou et al., 2024), we evaluate using MAUVE score (Pillutla et al., 2021) on conditional generation. We sample 1000 sequences from the LM1B test set, filtering for sequences with at least 100 tokens. For each sequence, we use the first 50 tokens as prompts and generate 5 samples of 50 tokens each using 50 diffusion steps with Euler sampling. We then compute MAUVE scores between the 5000 generated samples and 1000 reference samples from the test set.

[1] https://huggingface.co/stabilityai/sdxl-vae

# E. Generated Samples

## E.1. CIFAR-10 Class-Conditional Samples

*Figure 2.* Class-conditional samples generated by DiffusionBlocks on CIFAR-10. Each row shows samples from a different class.

## E.2. Language Generation Samples

Table 6 presents an example of text generated by our language model trained on LM1B.

*Table 6.* Text samples generated by our language model trained on LM1B.

| |
|---|
| the United States are fighting for the English team, resuming training on Sunday and that his players will be home from the Champions League final in Moscow on Wednesday. ”The evidence shows that the manufacturing sector is not quite |