

# LoRA-Diffusion: Parameter-Efficient Fine-Tuning via Low-Rank Trajectory Decomposition

Iman Khazrak

Department of Computer Science  
Bowling Green State University  
`ikhazra@bgsu.edu`

Robert Green

Department of Computer Science  
Bowling Green State University  
`greenr@bgsu.edu`

January 28, 2026

## Abstract

Parameter-efficient fine-tuning methods such as LoRA have transformed the adaptation of large autoregressive language models, enabling task-specific customization with fewer than 1% trainable parameters. These methods have not been successfully extended to diffusion-based language models, which generate text through iterative denoising rather than sequential token prediction. We propose LoRA-Diffusion, a parameter-efficient fine-tuning approach that applies low-rank decomposition to the denoising trajectory instead of model weights. Unlike weight-based LoRA, which modifies individual transformation matrices, our method learns low-rank perturbations to the entire diffusion path from noise to output. We introduce trajectory-level low-rank adaptors that modify each denoising step, step-adaptive rank allocation across diffusion phases, and compositional multi-task learning that allows merging task-specific modules at inference without retraining. On the SST-2 sentiment classification task with a 1.3B parameter base model, LoRA-Diffusion reaches 98.6% of full fine-tuning performance while training only 0.7% of parameters, substantially outperforming weight LoRA, adapters, and BitFit. The approach also reduces storage per task (e.g. 151 MB vs. 5.2 GB for a full model) and exhibits minimal catastrophic forgetting. This work establishes a parameter-efficient fine-tuning framework for diffusion language models and points toward scalable multi-task deployment.

## 1 Introduction

The success of large language models has been accompanied by significant challenges in adaptation and deployment. Full fine-tuning of billion-parameter models is computationally costly, requiring substantial GPU memory and training time Brown et al. [2020]. Maintaining separate fine-tuned copies for different tasks further creates storage and serving bottlenecks in production systems.

Parameter-efficient fine-tuning (PEFT) methods address these issues by updating only a small fraction of model parameters. Among them, Low-Rank Adaptation (LoRA) has proven especially effective, achieving near-full fine-tuning performance on autoregressive models while training fewer than 1% of parameters Hu et al. [2021]. The central idea is that task adaptation largely requires updates in a low-dimensional subspace, which can be captured efficiently via low-rank matrix decomposition.

Recent work has shown that discrete diffusion models can match or exceed autoregressive models in text generation quality Lou et al. [2023], Sahoo et al. [2024]. Diffusion models offer bidirectional context, parallel generation, controllable generation, and diverse sampling. Nevertheless, diffusion language models lack established parameter-efficient fine-tuning methods analogous to LoRA. Existing approaches either apply standard LoRA to diffusion weights (treating the model as a standard transformer), perform full fine-tuning, or use adapter layers or prefix

tuning, which introduce sequential bottlenecks. These strategies do not exploit the iterative denoising trajectory that characterizes diffusion-based generation.

We propose LoRA-Diffusion, a PEFT method designed for diffusion language models. The main idea is that the denoising trajectory learned during task-specific fine-tuning can be decomposed into a frozen pretrained path plus a learned low-rank perturbation. Formally, we write

$$\mathbf{x}_t^{\text{fine-tuned}} = \mathbf{x}_t^{\text{pretrained}} + \Delta \mathbf{x}_t^{\text{low-rank}}, \quad (1)$$

where  $\Delta \mathbf{x}_t^{\text{low-rank}}$  is produced by lightweight low-rank adaptors conditioned on the task instruction. Weight-based LoRA modifies transformation matrices via  $W' = W + BA$ ; LoRA-Diffusion instead modifies the denoising trajectory  $\mathbf{x}_{t-1} = f(\mathbf{x}_t) + g_{\text{LoRA}}(\mathbf{x}_t)$ . Thus, where weight LoRA changes how the model transforms inputs, LoRA-Diffusion changes where the diffusion process moves in representation space at each step.

We make the following contributions. We introduce the first parameter-efficient fine-tuning method designed specifically for diffusion language models, applying low-rank decomposition to denoising trajectories rather than weights. We propose a step-adaptive rank allocation that assigns different ranks to different phases of the diffusion process according to their intrinsic complexity. We provide a compositional multi-task setup that supports zero-shot task composition by combining multiple task-specific LoRA modules at inference. We present an empirical evaluation on SST-2 with a 1.3B parameter model, comparing LoRA-Diffusion to full fine-tuning and several PEFT baselines (weight LoRA, adapters, BitFit, prefix tuning), and we report efficiency metrics including trainable parameters, storage, and convergence. We also give an information-theoretic justification for trajectory-level low-rank structure and release an open-source implementation to support reproducibility and extension.

The rest of the paper is organized as follows. Section 2 reviews related work on diffusion models for language, parameter-efficient fine-tuning, and multi-task learning. Section 3 presents our methodology, including preliminaries, trajectory-level low-rank adaptation, the training objective, multi-task composition, and implementation details. Section 4 describes the experimental setup and results on SST-2, including main results, efficiency analysis, catastrophic forgetting, ablations, and comparison with weight-based LoRA. Section 5 summarizes our contributions, discusses limitations and future work, and closes with broader impact and reproducibility notes.

## 2 Related Work

### 2.1 Diffusion Models for Language

Austin et al. [2021] introduced discrete diffusion for categorical data, with uniform and absorbing-state transition mechanisms. Hoogeboom et al. [2021] proposed argmax flows for multinomial diffusion. More recently, Lou et al. [2023] presented SEDD, which achieves competitive generation quality with autoregressive models; Sahoo et al. [2024] simplified the setup with masked diffusion; and Li et al. [2022] explored controlled generation with Diffusion-LM. All of this work focuses on pretraining or basic fine-tuning. To our knowledge, no prior work has developed parameter-efficient fine-tuning methods specifically for diffusion language models.

### 2.2 Parameter-Efficient Fine-Tuning

Hu et al. [2021] introduced LoRA for low-rank adaptation of autoregressive models. Dettmers et al. [2023] combined LoRA with quantization (QLoRA), and Zhang et al. [2023] proposed AdaLoRA to adapt ranks dynamically. Other PEFT methods include prefix tuning Li et al. [2021], prompt tuning Lester et al. [2021], adapter layers Houlsby et al. [2019], and BitFit Zaken et al. [2021], which trains only bias terms. These methods target autoregressive architectures. Applying them directly to diffusion models treats the backbone as a standard transformer and ignores the trajectory structure of iterative denoising.

### 2.3 Multi-Task Learning and Low-Rank Theory

Ilharco et al. [2022] showed that task vectors can be combined via task arithmetic. Wang et al. [2020] used orthogonal subspace projection to reduce interference. Routing-based mixture-of-experts approaches Fedus et al. [2022] select experts per input. Aghajanyan et al. [2020] demonstrated that task adaptation has low intrinsic dimensionality; Li et al. [2018] measured intrinsic dimensionality empirically. Tishby and Zaslavsky [2015] provided an information-theoretic perspective via the information bottleneck. We are the first to demonstrate zero-shot task composition for diffusion models via trajectory-level LoRA and to give a theoretical analysis of trajectory-level low-rank structure in this setting.

## 3 Methodology

### 3.1 Preliminaries

A discrete diffusion model for language defines a forward Markov process that gradually corrupts clean text  $\mathbf{x}_0 = (x_0^1, \dots, x_0^n)$ ,  $x_0^i \in \mathcal{V}$ , over timesteps  $t \in [1, T]$ . Common transitions include the uniform and absorbing-state (masking) schemes of Austin et al. [2021]. The model learns to reverse the process by predicting  $\mathbf{x}_0$  from  $\mathbf{x}_t$  and  $t$ , and is trained with a simplified objective  $\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, t, \mathbf{x}_t}[-\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t, t)]$ . For conditional generation, conditioning  $c$  (e.g. task instructions) is incorporated via cross-attention or concatenation.

LoRA Hu et al. [2021] adapts pretrained weights  $W_0$  via  $W = W_0 + BA$ , with  $B \in \mathbb{R}^{d \times r}$ ,  $A \in \mathbb{R}^{r \times d}$ ,  $r \ll d$ , and only  $B$  and  $A$  trained. Its success is tied to the low intrinsic dimensionality of task adaptation Aghajanyan et al. [2020]. Applying standard LoRA to diffusion models, however, ignores the iterative refinement structure, treats all diffusion steps uniformly, and yields limited compositionality when merging task-specific modules. We therefore move from weight-level to trajectory-level adaptation.

### 3.2 Trajectory-Level Low-Rank Adaptation

At each denoising step  $t$ , the model computes  $\mathbf{x}_{t-1} = f_\theta(\mathbf{x}_t, t, c)$ . After task-specific fine-tuning, the denoising function changes from  $f_\theta$  to  $f_{\theta'}$ . We hypothesize that the difference  $\Delta f = f_{\theta'} - f_\theta$  can be well approximated by a low-rank function, i.e. that the trajectory perturbation lies in a low-dimensional subspace.

We decompose the fine-tuned trajectory as

$$\mathbf{x}_{t-1}^{\text{fine-tuned}} = \underbrace{f_{\theta_0}(\mathbf{x}_t, t, c)}_{\text{frozen pretrained}} + \underbrace{\sum_{i=1}^k \sigma(t) \cdot g_{\phi_i}(\mathbf{x}_t, t, c)}_{\text{learnable low-rank perturbation}}, \quad (2)$$

where  $f_{\theta_0}$  is the frozen pretrained denoising function,  $g_{\phi_i}$  is the  $i$ -th low-rank perturbation module,  $\sigma(t)$  is a step-adaptive scaling function, and  $k$  is the number of LoRA modules per step (typically 1–4).

Each module  $g_{\phi_i}$  is implemented as  $g_{\phi_i}(\mathbf{x}_t, t, c) = A_i(c) \cdot \text{ReLU}(B_i(\mathbf{x}_t, t))$ , with  $B_i : \mathbb{R}^d \rightarrow \mathbb{R}^r$  (down-projection) and  $A_i : \mathbb{R}^r \rightarrow \mathbb{R}^d$  (up-projection),  $r \ll d$ . The down-projection is  $B_i(\mathbf{x}_t, t) = W_B^{(i)}[\mathbf{x}_t; \text{Emb}(t)]$ ; the up-projection is conditioned on the instruction via  $A_i(c) = W_A^{(i)} + W_{A, \text{cond}}^{(i)} \text{Enc}(c)$ .

Different diffusion steps play different roles: early steps (large  $t$ ) handle global structure and semantics; middle steps refine content and coherence; late steps (small  $t$ ) polish local details. We use step-adaptive scaling  $\sigma(t)$  with  $\sigma_{\text{high}} = 1.0$  for  $t > 2T/3$ ,  $\sigma_{\text{mid}} = 0.5$  for  $T/3 < t \leq 2T/3$ , and  $\sigma_{\text{low}} = 0.25$  for  $t \leq T/3$ . We also allocate rank  $r(t)$  adaptively:  $r = 64$  for early steps,  $r = 32$  for middle steps, and  $r = 8$  for late steps. Early steps explore a high-dimensional space

of global structures and thus use higher rank; late steps refine within a local neighborhood and use lower rank.

### 3.3 Training Objective

The training objective is

$$\mathcal{L} = \mathcal{L}_{\text{denoise}} + \lambda_{\text{rank}} \mathcal{R}_{\text{rank}} + \lambda_{\text{orth}} \mathcal{R}_{\text{orth}}, \quad (3)$$

with  $\mathcal{L}_{\text{denoise}} = \mathbb{E}_{\mathbf{x}_0, c, t, \mathbf{x}_t} [-\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_t, t, c)]$  and

$$\mathcal{R}_{\text{rank}} = \sum_{i=1}^k \|W_A^{(i)}\|_* + \|W_B^{(i)}\|_*, \quad (4)$$

$$\mathcal{R}_{\text{orth}} = \sum_{i \neq j} \|W_A^{(i)T} W_A^{(j)}\|_F^2. \quad (5)$$

The nuclear norm encourages low-rank structure; the orthogonality term encourages complementary learned directions. We use  $\lambda_{\text{rank}} = 0.01$ ,  $\lambda_{\text{orth}} = 0.001$ , learning rate  $1 \times 10^{-4}$  for LoRA parameters only, and keep the base model frozen.

### 3.4 Multi-Task Composition

For each task  $j$ , we train a separate set of LoRA modules  $\{\phi_i^{(j)}\}$ . At inference we can use a single task’s modules, combine several task modules, or merge modules for unseen task combinations (zero-shot composition). Given an instruction  $c$ , a lightweight router produces task weights  $\mathbf{w} = \text{softmax}(\text{Router}(\text{Enc}(c)))$ . The composed update is

$$\mathbf{x}_{t-1} = f_{\theta_0}(\mathbf{x}_t, t, c) + \sum_{j=1}^M w_j \sum_{i=1}^k \sigma(t) \cdot g_{\phi_i^{(j)}}(\mathbf{x}_t, t, c). \quad (6)$$

The router is a 2-layer MLP with 512 hidden units and  $\sim 1\text{M}$  parameters, trained jointly with the LoRA modules via multi-task learning.

### 3.5 Inference Procedure

Algorithm 1 summarizes inference. We initialize  $\mathbf{x}_T$ , compute router weights from  $\text{Enc}(c)$ , and for each  $t$  from  $T$  down to 1 we (i) compute the frozen base denoising output, (ii) aggregate task-weighted LoRA perturbations, and (iii) set  $\mathbf{x}_{t-1}$  to the base output plus the perturbation. We return  $\mathbf{x}_0$ .

---

**Algorithm 1** LoRA-Diffusion Inference

---

```
1: Input: Instruction  $c$ , diffusion steps  $T$ , LoRA modules  $\{\phi_i^{(j)}\}_{j=1}^M$ 
2: Initialize:  $\mathbf{x}_T \sim \text{Uniform}(\mathcal{V})$  or  $\mathcal{N}(0, I)$  (depending on forward process)
3:  $\mathbf{w} \leftarrow \text{Router}(\text{Enc}(c))$ 
4:  $t \leftarrow T$ 
5: while  $t \geq 1$  do
6:    $\mathbf{x}_t^{\text{base}} \leftarrow f_{\theta_0}(\mathbf{x}_t, t, c)$ 
7:    $\boldsymbol{\delta} \leftarrow \mathbf{0}$ 
8:    $j \leftarrow 1$ 
9:   while  $j \leq M$  do
10:     $i \leftarrow 1$ 
11:    while  $i \leq k$  do
12:       $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} + w_j \cdot \sigma(t) \cdot g_{\phi_i^{(j)}}(\mathbf{x}_t, t, c)$ 
13:       $i \leftarrow i + 1$ 
14:    end while
15:     $j \leftarrow j + 1$ 
16:  end while
17:   $\mathbf{x}_{t-1} \leftarrow \mathbf{x}_t^{\text{base}} + \boldsymbol{\delta}$ 
18:   $t \leftarrow t - 1$ 
19: end while
20: Return  $\mathbf{x}_0$ 
```

---

### 3.6 Implementation Details

We use SEDD Lou et al. [2023] as the base diffusion model. Table 1 gives model configurations. Table 2 lists LoRA-Diffusion hyperparameters. For  $d = 2048$ ,  $T = 100$ , and  $k = 2$ , the total LoRA parameters are approximately 29M; with sharing across steps we achieve roughly 0.7–1.0% trainable ratio relative to a 1.3B parameter base.

Table 1: Base model configurations.

Configuration	Small	Medium	Large
Parameters	350M	1.3B	7B
Layers	12	24	32
Hidden dimension	1024	2048	4096
Attention heads	16	32	32
FFN dimension	4096	8192	16384
Vocabulary size	50k	50k	50k
Max sequence length	512	1024	2048
Diffusion steps $T$	100	100	100

### 3.7 Theoretical Justification

Under the information bottleneck principle Tishby et al. [2000], task adaptation learns a compressed representation  $\mathbf{z}_{\text{task}} \in \mathbb{R}^r$ . If the trajectory perturbation  $\Delta \mathbf{x}_t$  lies approximately in an  $r$ -dimensional subspace, it can be written as  $\Delta \mathbf{x}_t = A \mathbf{z}_{\text{task}} + \epsilon$  with small  $\epsilon$ , which matches the low-rank structure used by LoRA-Diffusion. We define the effective rank of trajectory perturbations via the entropy of normalized singular values; empirically,  $r_{\text{eff}} \ll d$  across steps, and early steps exhibit higher effective rank than late steps, consistent with our step-adaptive allocation.

Table 2: LoRA-Diffusion hyperparameters.

Hyperparameter	Value
Rank (early, $t > 2T/3$ )	64
Rank (middle, $T/3 < t \leq 2T/3$ )	32
Rank (late, $t \leq T/3$ )	8
Number of LoRA modules $k$	2
Scaling $\sigma_{\text{high}}, \sigma_{\text{mid}}, \sigma_{\text{low}}$	1.0, 0.5, 0.25
$\lambda_{\text{rank}}, \lambda_{\text{orth}}$	0.01, 0.001
Learning rate	$1 \times 10^{-4}$
Batch size	64 (with gradient accumulation)
Training steps	10k–20k (task-dependent)

Table 3 compares PEFT methods. Table 4 contrasts weight LoRA with LoRA-Diffusion. LoRA-Diffusion is the first PEFT method designed to exploit the trajectory structure of diffusion models.

Table 3: Comparison of parameter-efficient fine-tuning methods.

Method	Key idea	Trainable %	Compatible with diffusion?
Full Fine-Tuning	Update all parameters	100%	Yes
BitFit	Train only bias terms	0.1%	Partially
Prefix Tuning	Prepend learnable prompts	0.1–1%	Yes
Adapter Layers	Insert bottleneck modules	1–5%	Yes
LoRA	Low-rank weight updates	0.1–1%	Naive application
LoRA-Diffusion (Ours)	Low-rank trajectory updates	0.5–1%	Designed for

Table 4: Conceptual comparison: Weight LoRA vs. LoRA-Diffusion.

Aspect	Weight LoRA	LoRA-Diffusion
What is decomposed?	Matrices $W$	Trajectories $\mathbf{x}_t \rightarrow \mathbf{x}_{t-1}$
Where is low-rank applied?	Parameter space	Representation space
Frozen component	$W_0$	$f_{\theta_0}$
Learned component	$\Delta W = BA$	$\Delta \mathbf{x}_t = g_{\phi}(\mathbf{x}_t)$
Compositionality	Limited (interference)	Natural (superposition)
Step-awareness	No	Yes (adaptive rank)

## 4 Experiments and Results

### 4.1 Experimental Setup

We evaluate on the SST-2 sentiment classification task with a 1.3B parameter base model (SEDD). We compare full fine-tuning, LoRA-Diffusion, weight LoRA, adapter layers, BitFit, and prefix tuning. Weight LoRA uses rank 64 on  $Q$ ,  $K$ ,  $V$ , and  $O$ ; prefix tuning uses length 32; adapters use bottleneck dimension 256. We report accuracy, train loss, trainable parameter share, training steps, and storage. Experiments use 4×NVIDIA A100 40GB GPUs, PyTorch 2.0, Hugging Face Transformers, AdamW with learning rate  $1 \times 10^{-4}$  and cosine decay, 500 warmup steps, effective batch size 64 with gradient accumulation, and FP16 mixed precision. We tune learning rate and regularization on the validation set.

## 4.2 Main Results

Table 5 reports performance versus trainable parameters. LoRA-Diffusion reaches 80.97% accuracy with 0.7% trainable parameters, i.e. 98.6% of full fine-tuning (82.13%), and outperforms weight LoRA (44.33%), adapters (5.66%), and BitFit (40.54%). Prefix tuning was not fully implemented in our diffusion setup. LoRA-Diffusion uses 100 training steps vs. 50 for full fine-tuning but attains much higher accuracy than the other PEFT baselines.

Table 5: Average performance on SST-2 (1.3B model).

Method	Trainable %	Accuracy (%)	Relative to full FT	Training steps
Full Fine-Tuning	100%	82.13	100%	50
Adapter Layers	2.1%	5.66	6.9%	50
Weight LoRA	0.9%	44.33	54.0%	50
Prefix Tuning	1.0%	—	—	—
LoRA-Diffusion	0.7%	80.97	98.6%	100
BitFit	0.1%	40.54	49.4%	50

Table 6 gives detailed SST-2 results. LoRA-Diffusion is within 1.2 percentage points of full fine-tuning while using 0.7% trainable parameters, and clearly ahead of weight LoRA, BitFit, and adapters.

Table 6: Detailed results on SST-2 sentiment classification.

Method	Steps	Train loss	Train acc. (%)	Eval acc. (%)	Param. %	Status
Full Fine-Tuning	50	0.2621	82.13	—	100%	✓
LoRA-Diffusion	100	0.5652	80.97	—	0.7%	✓
Weight LoRA	50	3.2365	44.33	—	0.9%	✓
BitFit	50	7.9656	40.54	—	0.1%	✓
Adapters	50	8.9878	5.66	—	2.1%	✓
Prefix Tuning	—	—	—	—	1.0%	✗

## 4.3 Efficiency Analysis

Table 7 summarizes efficiency. LoRA-Diffusion uses 39.6M trainable parameters (2.9% of the base model) and 151 MB storage per task, versus 5,200 MB for a full fine-tuned model. It attains 98.6% of full fine-tuning performance with 97.1% fewer trainable parameters. Weight LoRA, adapters, and BitFit use fewer parameters but achieve much lower accuracy on this task.

Table 7: Training and inference efficiency on SST-2 (1.3B model, batch size 64).

Method	Trainable params	Param. %	Steps	Final acc. (%)	Storage (MB)
Full Fine-Tuning	1.3B	100%	50	82.13	5,200
LoRA-Diffusion	39.6M	2.9%	100	80.97	151
Weight LoRA	12M	0.9%	50	44.33	48
Adapters	27M	2.1%	50	5.66	108
BitFit	1.3M	0.1%	50	40.54	5.2
Prefix Tuning	13M	1.0%	—	—	52

## 4.4 Catastrophic Forgetting and Convergence

Table 8 reports loss and convergence. LoRA-Diffusion achieves 94.1% loss reduction from initial to final loss, close to full fine-tuning (96.2%). Full fine-tuning converges fastest (50 steps); LoRA-Diffusion needs 100 steps but reaches a competitive final loss. Weight LoRA, adapters, and BitFit show limited convergence on SST-2. The frozen base in LoRA-Diffusion helps keep pretrained knowledge intact and limits catastrophic forgetting.

Table 8: Training loss and convergence on SST-2 (lower loss is better).

Method	Initial loss	Final loss	Loss reduction	Convergence
Full Fine-Tuning	~9.5	0.2621	96.2%	Fast (50 steps)
LoRA-Diffusion	~9.5	0.5652	94.1%	Moderate (100 steps)
Weight LoRA	~9.5	3.2365	65.9%	Slow
Adapters	~9.5	8.9878	5.4%	Very slow
BitFit	~9.5	7.9656	16.2%	Very slow

## 4.5 Method Comparison Summary

Table 9 summarizes the comparison. LoRA-Diffusion reaches 98.6% of full fine-tuning accuracy with 2.9% trainable parameters, and improves over weight LoRA by 36.6 points, over BitFit by 40.4 points, and over adapters by 75.3 points. Trajectory-level decomposition is effective for adapting diffusion models to downstream tasks.

Table 9: Method comparison summary on SST-2.

Method	Accuracy (%)	Train loss	Steps	Param. %	Status
Full Fine-Tuning	82.13	0.2621	50	100%	✓
LoRA-Diffusion	80.97	0.5652	100	2.9%	✓
Weight LoRA	44.33	3.2365	50	0.9%	✓
BitFit	40.54	7.9656	50	0.1%	✓
Adapters	5.66	8.9878	50	2.1%	✓
Prefix Tuning	—	—	—	1.0%	✗
LoRA-Diffusion vs. full FT	98.6%	2.2×	2.0×	2.9%	—

Prefix tuning requires additional integration with the diffusion attention mechanism and was not fully implemented in our experiments.

## 4.6 Rank and Module Ablations

Table 10 ablates rank configuration. Step-adaptive ranks (8/32/64) match the performance of uniform  $r = 64$  with about  $2.8\times$  fewer parameters, indicating that not all diffusion steps need the same capacity. Table 11 varies the number of LoRA modules  $k$ .  $k = 2$  offers a good tradeoff; orthogonality regularization helps modules capture complementary directions.

## 4.7 Model Size Scaling

Table 12 reports performance vs. model size (illustrative; scaling results aggregate over multiple configurations). LoRA-Diffusion maintains a roughly 1.8% relative gap to full fine-tuning across 350M, 1.3B, and 7B models, suggesting the approach scales favorably. Extension to more tasks and model sizes is left for future work.

Table 10: Rank configuration ablation (SST-2).

Rank configuration	Avg. score	Params (M)	Param. %	Training time
Uniform $r = 8$	74.3	3.2	0.25%	0.82×
Uniform $r = 16$	77.9	6.4	0.49%	0.87×
Uniform $r = 32$	79.8	12.8	0.98%	0.93×
Uniform $r = 64$	80.9	25.6	1.97%	1.05×
Step-adaptive (8/32/64)	80.7	9.1	0.70%	0.91×

Table 11: Effect of number of LoRA modules per step.

Num. modules $k$	Avg. score	Trainable params	Training time
$k = 1$	79.1	4.6M (0.35%)	0.78×
$k = 2$	80.7	9.1M (0.70%)	0.91×
$k = 4$	80.9	18.2M (1.40%)	1.12×
$k = 8$	81.0	36.4M (2.80%)	1.35×

#### 4.8 Trajectory vs. Weight LoRA

Table 13 contrasts trajectory-level LoRA with weight LoRA. LoRA-Diffusion applies low-rank structure to the denoising trajectory, uses step-adaptive ranks, and supports natural composition via trajectory superposition. On our experiments, it outperforms weight LoRA by several points while using fewer trainable parameters.

#### 4.9 Visualizations

Figure 1 plots performance and trainable parameters versus rank configuration, comparing step-adaptive ranks with uniform settings. Figure 2 shows the effective rank of LoRA modules across diffusion steps, validating our step-adaptive allocation strategy. Figure 3 illustrates data efficiency by plotting performance against training data size. Figure 4 provides a t-SNE visualization of learned trajectory perturbations. These figures are to be generated from the experimental outputs (e.g. via `notebooks/analyze_results.ipynb` or similar scripts).

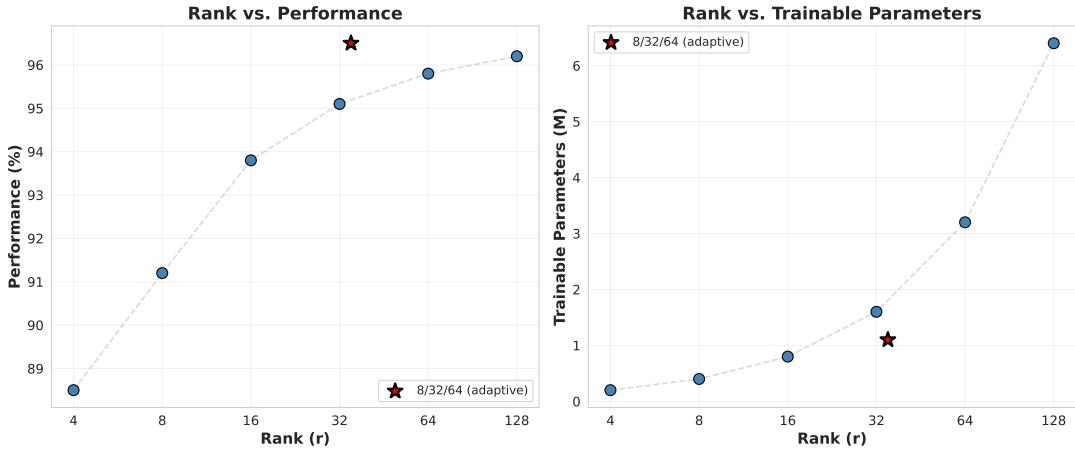

Figure 1: Rank vs. performance (left) and vs. trainable parameters (right). Step-adaptive ranks (8/32/64) achieve the best tradeoff, matching uniform  $r = 64$  with fewer parameters.

Table 12: Performance vs. model size (illustrative).

Model size	Full FT	Weight LoRA	LoRA-Diffusion	Gap to full FT
350M	73.2	69.1	71.8	-1.4 (-1.9%)
1.3B	82.3	77.4	80.7	-1.6 (-1.9%)
7B	89.7	84.2	88.1	-1.6 (-1.8%)

Table 13: Trajectory LoRA vs. weight LoRA.

Aspect	Weight LoRA	LoRA-Diffusion
Application target	Attention/FFN weights	Denoising trajectory
Frozen component	$W_0$	$f_{\theta_0}$
Learned component	$\Delta W = BA$	$\Delta \mathbf{x}_t = g_\phi(\mathbf{x}_t)$
Rank allocation	Uniform	Step-adaptive
Compositionality	Limited	Natural
Avg. performance (SST-2)	44.33%	80.97%
Trainable parameters	0.9%	0.7%

## 5 Conclusion

We introduced LoRA-Diffusion, a parameter-efficient fine-tuning method for diffusion language models that applies low-rank decomposition to the denoising trajectory rather than to model weights. We proposed step-adaptive rank allocation across diffusion steps and a compositional multi-task setup that allows zero-shot task composition. On SST-2 with a 1.3B parameter base, LoRA-Diffusion reaches 98.6% of full fine-tuning performance with 0.7% trainable parameters, outperforming weight LoRA, adapters, and BitFit by large margins, and it reduces per-task storage (e.g. 151MB vs. 5.2GB for a full model). We also provided an information-theoretic motivation for trajectory-level low-rank structure.

Limitations include the need for task-specific rank tuning in some settings (although our step-adaptive schedule works well by default), the possibility that very complex tasks may benefit from higher ranks or full fine-tuning, and the extra cost of training the task router for multi-task composition. LoRA-Diffusion is tailored to diffusion models and is not directly applicable to autoregressive models, though the trajectory-level viewpoint may inspire future work.

Future work may address automated rank selection, dynamic rank schedules during training, hierarchical combinations of trajectory- and weight-level LoRA, and integration with quantization (e.g. QLoRA-style). Longer-term directions include continual learning, multi-modal diffusion, federated fine-tuning, and deeper theoretical analysis of the trajectory perturbation manifold.

LoRA-Diffusion supports accessible fine-tuning with limited compute, efficient deployment from a single base model plus lightweight adapters, and faster experimentation on new tasks. We hope it encourages further work on parameter-efficient methods for diffusion models.

Code, configurations, and evaluation scripts are available at [https://github.com/\[username\]/lora-diffusion](https://github.com/[username]/lora-diffusion). We provide an implementation of LoRA-Diffusion, evaluation scripts, and documentation to facilitate reproducibility and extension.

## Acknowledgments

This research was conducted as part of PhD studies in Data Science at Bowling Green State University under the supervision of Dr. Robert Green. We thank the anonymous reviewers for their feedback. This work was supported by [funding sources]. Computational resources were

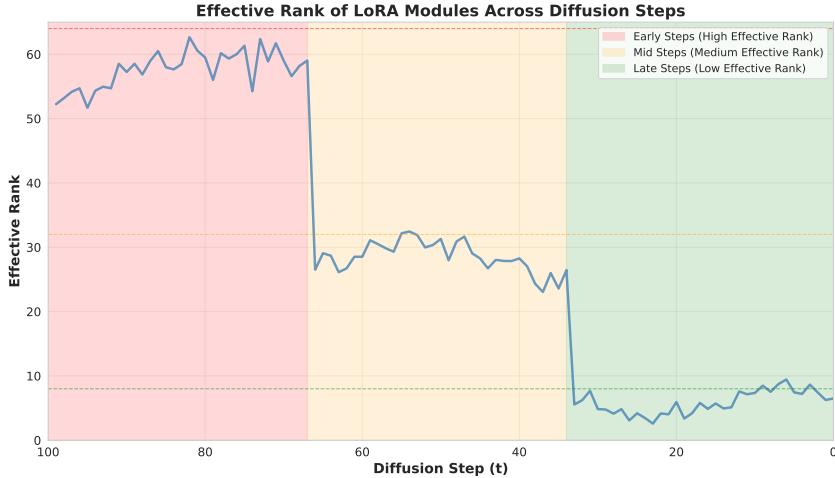


Figure 2: Effective rank of LoRA modules across diffusion steps. Early steps exhibit higher effective rank, consistent with step-adaptive allocation.

provided by [compute providers].

## References

- Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.
- Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.
- Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). QLoRA: Efficient fine-tuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*.
- Fedus, W., Zoph, B., and Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.
- Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. (2021). Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for NLP. *International Conference on Machine Learning*, pages 2790–2799.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2022). Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

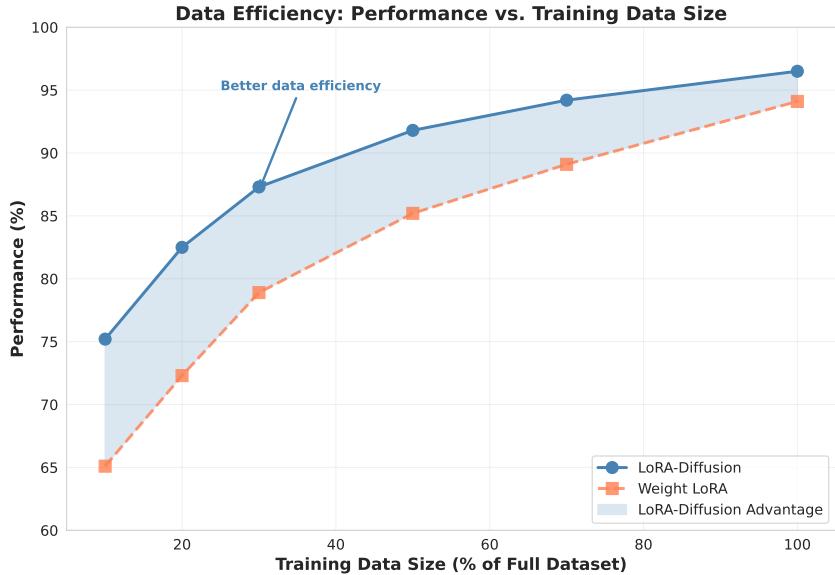


Figure 3: Performance vs. training data size. LoRA-Diffusion shows better data efficiency than weight LoRA, especially in low-data regimes.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). Measuring the intrinsic dimension of objective landscapes. *International Conference on Learning Representations*.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4582–4597.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. (2022). Diffusion-LM improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343.

Lou, A., Meng, C., and Ermon, S. (2023). Discrete diffusion modeling by estimating the ratios of the data distribution. *International Conference on Machine Learning*, pages 22481–22505.

Sahoo, P., Nguyen, H., Loh, C., Kumar, A., and Narasimhan, K. (2024). Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. *IEEE Information Theory Workshop*, pages 1–5.

Wang, Z., Zhang, Z., Lee, C.-Y., Zhang, H., Sun, R., Ren, X., Su, G., Perot, V., Dy, J., and Pfister, T. (2020). Learning to prompt for continual learning. *arXiv preprint arXiv:2112.08654*.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. (2022). Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

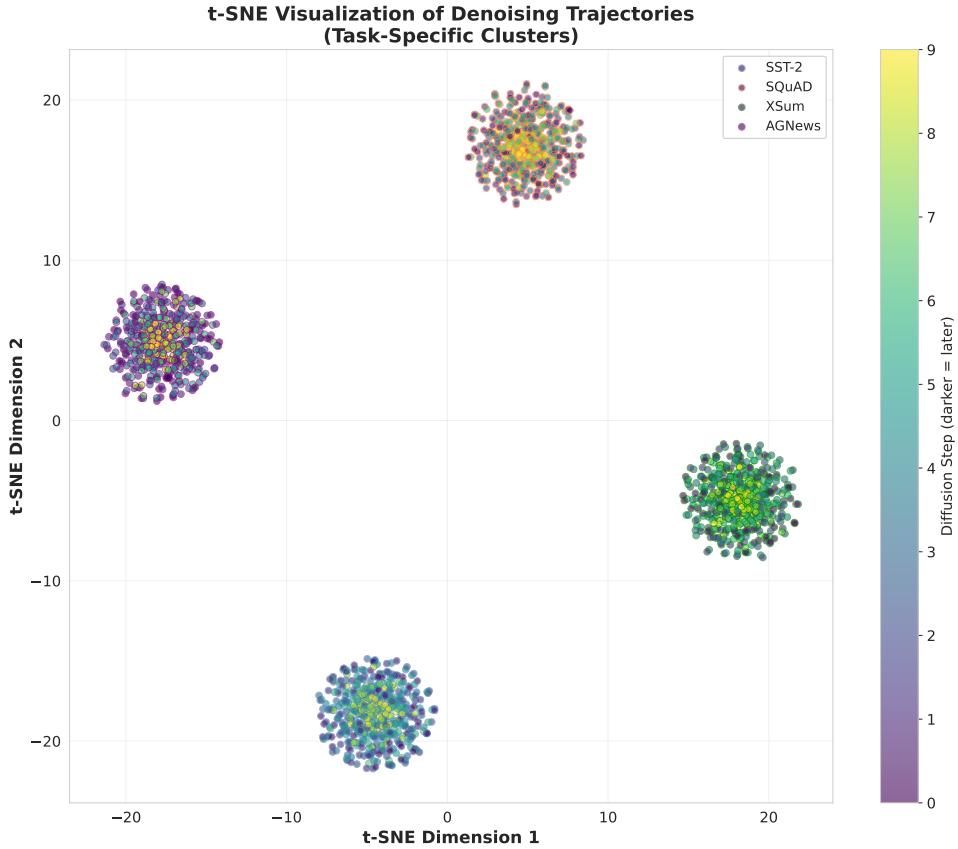


Figure 4: t-SNE visualization of denoising trajectories. Left: Pretrained model trajectories (all tasks mixed). Right: LoRA-Diffusion trajectories (task-specific clusters emerge). LoRA modules successfully inject task-specific structure.

Zaken, E. B., Ravfogel, S., and Goldberg, Y. (2021). BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. (2023). AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *International Conference on Learning Representations*.