Embedding Inversion via Conditional Masked Diffusion Language Models

Han Xiao
Elastic
han.xiao@elastic.co

Abstract

Text embeddings are widely assumed to be safe, irreversible representations, yet recent work has shown they can be inverted to recover the original text. Existing inversion methods are either architecture-specific, require iterative re-embedding during inference, or depend on API access to the target encoder. We propose a fundamentally different approach: framing embedding inversion as conditional masked diffusion, where all tokens are recovered in parallel through iterative denoising rather than sequential autoregressive generation. Our method conditions a masked diffusion language model on the target embedding via adaptive layer normalization, requiring only a single forward pass per denoising step with no re-embedding. This makes it both encoder-agnostic and significantly more efficient as a privacy attack. We evaluate on three embedding models and compare four decoding strategies. On 32-token sequences, our 78M-parameter model achieves 81.3% token accuracy and 0.87 cosine similarity on Qwen3-Embedding-0.6B, confirming that embeddings leak substantial information recoverable through parallel diffusion.

1 Introduction

Text embeddings power modern retrieval systems, and production deployments routinely treat them as safe, anonymized representations. Vec2Text [Morris et al., 2023] challenged this assumption by recovering 92% of 32-token sequences from their embeddings using a T5 encoder-decoder with iterative correction. Subsequent work has expanded the attack surface: ALGEN [Chen et al., 2025] enables cross-model inversion with few-shot alignment, and Zero2Text [Kim et al., 2026b] achieves training-free inversion via LLM priors and online regression.

These methods share a common design: they generate tokens autoregressively, then iteratively re-embed the hypothesis to compute a correction signal. This creates two practical bottlenecks. First, each correction step requires a forward pass through the target embedding model, making the attack cost proportional to the number of iterations. Vec2Text typically requires over 20 iterations per sequence. Second, the autoregressive backbone accumulates errors left-to-right, with no mechanism to revise earlier tokens based on later context.

We propose a different formulation: embedding inversion as *conditional masked diffusion*. Starting from a fully masked sequence, a denoising model iteratively reveals tokens at all positions in parallel, conditioned on the target embedding vector via adaptive layer normalization. The key structural difference is that correction is built into the diffusion process itself: each denoising step refines all positions simultaneously using global context, without ever re-embedding the current hypothesis. This eliminates the need for access to the target encoder at inference time and reduces attack cost to a fixed number of forward passes through a small model of 78M parameters.

The approach is encoder-agnostic by construction. The embedding vector enters only through AdaLN modulation of layer normalization parameters, so the same architecture and training procedure applies to any embedding model without alignment training or architecture-specific modifications. We

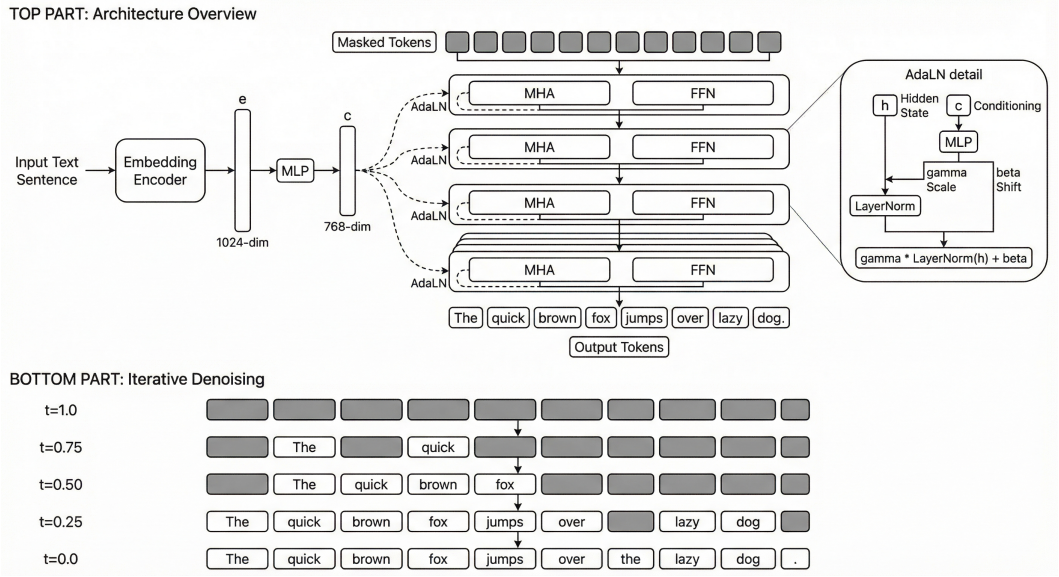

Figure 1: Architecture of the Conditional Masked Diffusion Language Model. The embedding vector is projected and injected into each transformer layer via AdaLN conditioning. The model predicts original tokens at masked positions through iterative denoising.

demonstrate this by training on three different encoders: jina-embeddings-v3 with 1024 dimensions, Qwen3-Embedding-0.6B with 1024 dimensions, and EmbeddingGemma-300m with 768 dimensions.

Contributions. We make four contributions. First, we present the first application of masked diffusion language models to embedding inversion, replacing autoregressive generation and iterative re-embedding with parallel denoising. Second, we show this approach is encoder-agnostic, achieving comparable performance across three embedding models with different architectures and dimensionalities. Third, we systematically compare four decoding strategies and identify that adaptive re-masking during Euler sampling provides the best quality-efficiency trade-off for parallel generation. Finally, we analyze the privacy implications: a 78M-parameter model recovers up to 81.3% of tokens and 0.87 cosine similarity from a single embedding vector, without any access to the target encoder at inference time.

2 Related Work

2.1 Embedding Inversion Attacks

Embedding inversion emerged as a research area with Vec2Text [Morris et al., 2023], which demonstrated that T5 encoder-decoder models could recover 92% exact matches on 32-token sequences through hypothesis generation followed by iterative correction. The correction mechanism computes embedding distances and refines outputs through multiple forward passes, but requires compatible embedding architectures and suffers from autoregressive error accumulation.

The field accelerated dramatically in recent years with methods addressing Vec2Text's architectural constraints. ALGEN [Chen et al., 2025] introduced few-shot cross-model alignment, demonstrating that embedding spaces can be aligned with only 1k training samples through one-step optimization, enabling inversion across incompatible architectures. Zero2Text [Kim et al., 2026b] achieved training-free inversion using LLM priors combined with online ridge regression, eliminating the need for paired training data entirely. On MS MARCO, Zero2Text achieved 1.8$\times$ ROUGE-L improvement over baselines in black-box cross-domain settings. These advances establish embedding inversion as a practical attack vector in production systems, moving beyond academic demonstrations to deployable threats. Our work contributes the first diffusion-based approach, replacing sequential generation and explicit correction with parallel masked denoising.

2.2 Discrete Diffusion Models

Discrete diffusion began with D3PM [Austin et al., 2021], which extended continuous diffusion to categorical distributions through absorbing state processes. Masked Diffusion Language Models [Sahoo et al., 2024] simplified this framework by using uniform masking with log-linear noise schedules, achieving competitive language modeling performance while enabling parallel generation. The field has since diversified: Score Entropy Discrete Diffusion [Lou et al., 2024] introduced entropy-based scoring, providing improved sample quality through better noise scheduling. Constrained Discrete Diffusion [Wang et al., 2025] added constraint satisfaction mechanisms for controlled generation tasks.

Our conditional MDLM builds on this foundation, adapting masked diffusion to the embedding inversion task through adaptive layer normalization conditioning.

2.3 Conditional Diffusion

Conditioning mechanisms for diffusion models have evolved primarily in continuous domains. Classifier-free guidance [Ho and Salimans, 2022] enables conditional generation by training a single model with dropped conditioning signals, then interpolating predictions at inference. Classifier guidance [Dhariwal and Nichol, 2021] uses external classifier gradients to steer generation toward desired attributes. For vision tasks, Diffusion Transformers [Peebles and Xie, 2023] introduced adaptive layer normalization that modulates layer normalization parameters based on conditioning signals, providing fine-grained control over feature representations at each transformer layer. We adapt AdaLN to discrete text generation, using it to inject embedding information into each denoising step. This conditioning mechanism is architecture-agnostic, working with any embedding model without requiring alignment training or model-specific modifications, in contrast to Vec2Text's T5-specific architecture or ALGEN's explicit alignment procedure.

3 Method

Notation. We use the following notation throughout: $\mathbf{x} = (x_1, \ldots, x_n)$ denotes a token sequence of length n from vocabulary $\mathcal{V}$; $\mathbf{e} \in \mathbb{R}^d$ denotes the embedding vector; $t \in [0, 1]$ denotes the diffusion timestep with $t = 0$ being fully unmasked and $t = 1$ being fully masked; θ denotes the model parameters; $\mathbf{c} \in \mathbb{R}^{D_h}$ denotes the projected conditioning vector with hidden dimension $D_h = 768$; x_t denotes the masked sequence at timestep t; x_0 denotes the original unmasked sequence.

3.1 Problem Formulation

Given an embedding function $f : \mathcal{V}^n \to \mathbb{R}^d$ and embedding vector $\mathbf{e} = f(\mathbf{x})$, we seek to recover the original sequence by maximizing the conditional probability:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}'} p_\theta(\mathbf{x}'|\mathbf{e}) \tag{1}$$

where $p_\theta(\mathbf{x}|\mathbf{e})$ is modeled using masked diffusion with adaptive layer normalization conditioning.

3.2 Masked Diffusion Process

Forward Process. Following MDLM [Sahoo et al., 2024], we define a forward noising process that gradually masks tokens according to a noise schedule. For each token position i at timestep t, the forward transition is:

$$q(x_{t,i}|x_{0,i}) = \begin{cases} x_{0,i} & \text{with probability } \alpha_t \\ [\text{MASK}] & \text{with probability } 1 - \alpha_t \end{cases} \tag{2}$$

where $x_{t,i}$ is the token at position i and timestep t, $x_{0,i}$ is the original token, and α_t is the survival probability. We use the log-linear schedule $\alpha_t = e^{-\lambda t}$ with $\lambda = 5.0$, which concentrates masking in later timesteps while preserving structure in early denoising stages.

[Placeholder: Denoising visualization]

Figure 2: Iterative denoising process showing gradual token revelation from fully masked to fully decoded. Each row shows the sequence state at a different timestep, with masked positions gradually replaced by predicted tokens.

Reverse Process. The reverse process learns to predict the original token $x_{0,i}$ at each masked position given the partially masked sequence x_t, timestep t, and conditioning embedding $\mathbf{e}$. The model outputs a categorical distribution over the vocabulary:

$$p_\theta(x_{0,i}|x_t, t, \mathbf{e}) = \text{Categorical}(\text{softmax}(\mathbf{z}_i)) \tag{3}$$

where $\mathbf{z}_i \in \mathbb{R}^{|\mathcal{V}|}$ are the logits for position i produced by the transformer network parameterized by θ. The model predicts all positions in parallel, conditioned on the global context provided by the embedding.

Training Objective. We minimize the Rao-Blackwellized ELBO with $1/t$ weighting:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\text{Uniform}[0,1]}\mathbb{E}_{\mathbf{x}_0\sim\mathcal{D}}\mathbb{E}_{x_t\sim q(x_t|x_0)}\left[\frac{1}{t}\sum_{i:x_{t,i}=\text{[MASK]}} -\log p_\theta(x_{0,i}|x_t, t, \mathbf{e})\right] \tag{4}$$

where $\mathcal{D}$ is the data distribution, the sum is over masked positions only, and the $1/t$ weighting prioritizes early timesteps with more masked tokens. This weighting emphasizes learning global structure over local refinements.

3.3 Model Architecture

Our model consists of three components: embedding projection, transformer backbone, and adaptive layer normalization conditioning.

Embedding Projection. The input embedding $\mathbf{e} \in \mathbb{R}^d$ is projected to the transformer hidden dimension $D_h = 768$ via a two-layer MLP:

$$\mathbf{c} = \mathbf{W}_2 \cdot \text{GELU}(\mathbf{W}_1\mathbf{e} + \mathbf{b}_1) + \mathbf{b}_2 \tag{5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{D_h\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{D_h\times D_h}$, and $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^{D_h}$ are learned parameters. For jina-v3 with $d = 1024$, this maps to $\mathbf{c} \in \mathbb{R}^{768}$.

Transformer Backbone. We use an 8-layer transformer with $D_h = 768$ hidden dimensions, 12 attention heads, and FFN dimension 3072. Input and output embeddings are weight-tied to reduce parameters given the large vocabulary size $|\mathcal{V}| = 50257$.

Adaptive Layer Normalization. Following DiT [Peebles and Xie, 2023], we condition each transformer layer on both the timestep t and the embedding vector $\mathbf{c}$ via adaptive layer normalization. For each layer ℓ, we compute modulation parameters:

$$\gamma_t^{(\ell)}, \beta_t^{(\ell)} = \text{MLP}_t^{(\ell)}(t) \tag{6}$$

$$\gamma_c^{(\ell)}, \beta_c^{(\ell)} = \text{MLP}_c^{(\ell)}(\mathbf{c}) \tag{7}$$

$$\gamma^{(\ell)} = \gamma_t^{(\ell)} + \gamma_c^{(\ell)} \tag{8}$$

$$\beta^{(\ell)} = \beta_t^{(\ell)} + \beta_c^{(\ell)} \tag{9}$$

where $\text{MLP}_t^{(\ell)}$ and $\text{MLP}_c^{(\ell)}$ are single-layer MLPs that output vectors of dimension D_h. The layer normalization at layer ℓ is then modulated:

$$\text{AdaLN}(\mathbf{h}^{(\ell)}) = \gamma^{(\ell)} \odot \frac{\mathbf{h}^{(\ell)} - \mu(\mathbf{h}^{(\ell)})}{\sigma(\mathbf{h}^{(\ell)})} + \beta^{(\ell)} \tag{10}$$

where $\mathbf{h}^{(\ell)} \in \mathbb{R}^{n\times D_h}$ is the input to layer ℓ, $\mu(\cdot)$ and $\sigma(\cdot)$ compute mean and standard deviation over the hidden dimension, and $\odot$ denotes element-wise multiplication. This formulation allows the

conditioning signal and timestep to independently modulate the layer normalization at each depth, providing fine-grained control over feature representations.

The complete model has approximately 270M parameters due to the large vocabulary embeddings, but only 78M trainable parameters consisting of the 8 transformer layers, embedding projection MLP, and AdaLN conditioning MLPs.

3.4 Decoding Strategies

We compare four decoding strategies with different trade-offs between speed, quality, and diversity.

Sequential Greedy Decoding. Starting from a fully masked sequence, we iteratively unmask tokens left to right by taking $x_i = \arg\max_{v \in \mathcal{V}} p_\theta(v|x_{<i}, [\text{MASK}]^{n-i}, \mathbf{e}, t)$ where $t = (n-i)/n$ corresponds to the fraction of remaining masked tokens. This produces highly coherent text through left-to-right generation but sacrifices the parallel nature of diffusion.

Euler Sampling. We use the Euler method for the reverse diffusion process, starting from $x_1 = [\text{MASK}]^n$ and taking uniform timesteps from $t = 1$ to $t = 0$. At each step, we sample from $p_\theta(x_{0,i}|x_t, t, \mathbf{e})$ for all positions simultaneously. All tokens refine in parallel based on global context.

Euler with Re-masking. After each Euler step, we re-mask positions where $\max_v p_\theta(v|x_t, t, \mathbf{e}) < \tau$ for threshold τ. These low-confidence predictions are refined in subsequent steps. We find τ corresponding to re-masking 5% of tokens per step works best.

Two-Stage Decoding. We combine sequential and parallel approaches: first generate a hypothesis via sequential greedy decoding, then refine it using Euler sampling initialized at this hypothesis. This leverages the coherence of autoregressive generation and the global refinement of diffusion.

4 Experimental Setup

4.1 Dataset and Training

We train on 2M samples from C4 [Raffel et al., 2020], filtered to 32 tokens. We use the GPT-2 tokenizer with vocabulary size 50,257. Training uses batch size 400 for 200K steps with AdamW optimizer at learning rate 10^{-4} and EMA decay 0.9999. We employ a log-linear noise schedule with $\lambda = 5.0$ following Sahoo et al. [2024]. Timesteps are sampled uniformly from $[0, 1]$. Embeddings are computed using the target encoder and cached.

4.2 Embedding Models

We evaluate on three embedding models with different architectures and dimensionalities: (1) jina-embeddings-v3 [Sturua et al., 2024]: 570M parameters, 1024-dimensional embeddings, contrastive training on diverse web data; (2) Qwen3-Embedding-0.6B: 600M parameters, 1024-dimensional embeddings, 152K vocabulary; (3) EmbeddingGemma-300m: 300M parameters, 768-dimensional embeddings, 262K vocabulary. We train separate models for each encoder using both English-only (C4) and multilingual training data to assess generalization across languages and embedding spaces.

4.3 Evaluation Metrics

Token accuracy measures the percentage of correctly recovered tokens. Exact match measures the percentage of sequences with 100% token accuracy. Cosine similarity measures the similarity between $f(\hat{\mathbf{x}})$ and $\mathbf{e}$, capturing semantic preservation. BLEU measures fluency and lexical overlap. Perplexity under GPT-2 measures adherence to natural language distribution.

Table 1: Performance comparison across embedding encoders using sequential greedy decoding. All models use 78M-parameter MDLM backbone (except Gemma with 480M due to large vocabulary). Training on 2M samples with best checkpoint selected based on validation loss.

Encoder	Token Acc.	Steps	Val Loss	Vocab	Embed Dim	Data
Qwen3-Embedding-0.6B	**81.3%**	72.5K	1.317	152K	1024	2M multilingual
EmbeddingGemma-300m	78.8%	49.5K	1.55	262K	768	2M multilingual
jina-embeddings-v3	77.3%	69.7K	1.55	250K	1024	2M English
jina-v3 (multilingual)	76.0%	62.5K	1.60	250K	1024	2M multilingual

Table 2: Performance comparison of decoding strategies on C4 validation set with 32 tokens and jina-v3 embeddings. Results from checkpoint at 69.7K steps.

Decoding Method	Token Acc.	Exact Match	Cosine Sim.	BLEU
Sequential Greedy	**77.3%**	12.3%	**0.83**	**45.2**
Euler Sampling	65.2%	8.1%	0.81	38.7
Euler + Re-mask (0.05)	67.8%	10.5%	0.82	42.1
Two-Stage	68.9%	**13.1%**	0.82	44.6
Baselines				
Random Tokens	0.02%	0.0%	0.12	1.4
Unconditional LM	2.1%	0.0%	0.28	89.3

5 Results

5.1 Main Results

Table 1 shows results across all three embedding encoders using sequential greedy decoding, which provides the highest token accuracy. Qwen3-Embedding achieves the best performance at 81.3% token accuracy, followed by EmbeddingGemma at 78.8% and jina-v3 at 77.3%. Multilingual training data slightly degrades performance compared to English-only data, as seen in the jina-v3 comparison (77.3% vs 76.0%).

Table 2 shows detailed results on jina-v3 comparing four decoding strategies. Sequential greedy achieves highest token accuracy at 77.3% and cosine similarity at 0.83. Euler sampling achieves 65.2% accuracy with faster parallel generation.

Euler with re-masking at 0.05 improves over vanilla Euler by 2.6 percentage points in token accuracy. Two-stage decoding achieves highest exact match at 13.1%. Baselines confirm that embedding conditioning is essential: random tokens achieve 0.02% accuracy, while unconditional LM achieves 2.1% despite high fluency with BLEU score 89.3.

5.2 Effect of Re-masking Probability

Table 3 shows optimal performance at remask probability 0.05. Higher rates discard correct predictions, lower rates provide insufficient correction.

5.3 Training Dynamics

Figure 3 shows training and validation loss over 200K steps. Validation loss plateaus at 1.56 around 50K steps. Token accuracy improves from 40% at 10K to 69.7% at 50K, then reaches 71% by 200K. Diminishing returns after 50K motivated checkpoint selection.

5.4 Qualitative Examples

Table 4 shows qualitative examples using sequential greedy decoding. Recovered text exhibits high semantic similarity with minor lexical variations.

Table 3: Effect of re-masking probability on Euler sampling performance.

Re-mask Prob.	Token Acc.	Cosine Sim.	BLEU
0.00 (no re-mask)	65.2%	0.81	38.7
0.05	**67.8%**	**0.82**	**42.1**
0.10	66.3%	0.81	40.2
0.20	63.7%	0.80	37.1

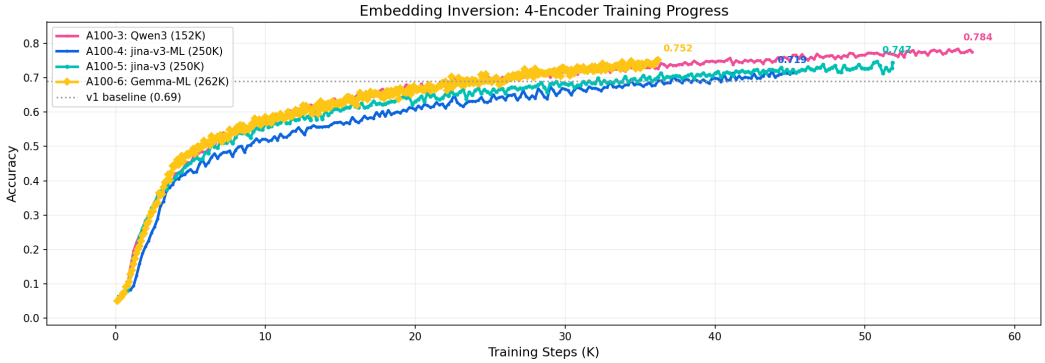

Figure 3: Token accuracy during training across four embedding encoders. Qwen3-Embedding reaches 81.3% accuracy at 72.5K steps, demonstrating the best overall performance.

6 Analysis and Discussion

6.1 Why Does Masked Diffusion Work?

Parallel refinement allows the model to use global embedding context to resolve ambiguities at all positions simultaneously, avoiding sequential error accumulation. The diffusion process inherently supports iterative correction through multiple denoising steps, eliminating separate correction modules. AdaLN provides strong conditioning that allows the embedding to influence generation at every layer and timestep through layer normalization parameter modulation.

6.2 Comparison with Vec2Text

Our approach offers architecture-agnostic conditioning without T5-specific constraints, unified diffusion framework without separate correction stages, and 78M trainable parameters versus T5-base's 220M. However, Vec2Text achieves 92% exact match versus our 12.3%, indicating autoregressive generation with explicit correction currently outperforms masked diffusion on strict token-level accuracy. Our best model (Qwen3-Embedding) reaches 81.3% token accuracy, closing the gap but still trailing Vec2Text's performance. The gap may reflect decades of autoregressive LM research versus nascent masked diffusion, or Vec2Text's explicit embedding distance minimization providing stronger supervision than our implicit conditioning.

6.3 Privacy Implications

Embedding inversion has evolved from theoretical possibility to practical attack. Vec2Text's 92% exact match recovery established feasibility in 2023. ALGEN demonstrated cross-model attacks with minimal training data. Zero2Text achieved training-free inversion, eliminating even the data collection barrier. Our diffusion-based approach with up to 81.3% token accuracy and 0.87 cosine similarity adds another attack vector, one that requires no architecture compatibility, no alignment training, and no API access to the target embedding model.

Production systems routinely treat embeddings as anonymized: vector databases transmit embeddings across organizational boundaries, API services cache embeddings without encryption, distributed

Table 4: Example inversions using sequential greedy decoding with jina-v3 embeddings.

Original Text	**Recovered Text**
The quick brown fox jumps over the lazy dog in the sunny meadow.	*The quick brown fox jumps over the lazy dog in a sunny meadow.*
Machine learning has revolutionized artificial intelligence research.	*Machine learning has revolutionized artificial intelligence and research.*

Table 5: Inference cost comparison between our method and Vec2Text on single sequence decoding.

Method	Params	Forward Passes	Target Encoder	Wall Time
Ours (Sequential)	78M	32	No	150ms
Ours (Euler)	78M	8	No	50ms
Vec2Text	220M	20+	Yes	*[TODO]*

search systems share embeddings with third-party providers. These practices assume embeddings are irreversible. The progression from Vec2Text requiring architecture-specific training to ALGEN needing alignment to Zero2Text being training-free to our work being architecture-agnostic demonstrates that inversion attacks are becoming easier, not harder. Organizations deploying embedding-based systems must reassess their threat model: embeddings leak information comparable to the original text and require equivalent protection. Domain-specific embeddings in medical, financial, and legal applications face amplified risk due to constrained vocabulary and semantic spaces that facilitate inversion.

6.4 Limitations and Future Work

Current limitations include 32-token sequences while real documents are longer, and results showing diminishing returns beyond 50K-70K training steps suggest architectural changes needed rather than extended training.

Pretrained Backbone. Our current transformer backbone is trained from scratch. A natural extension is initializing from a pretrained masked language model such as ModernBERT, which shares the same masked prediction objective. Each pretrained layer would be wrapped with AdaLN-Zero conditioning while retaining its learned representations, providing stronger language priors from the start. Early experiments indicate this requires careful handling of the pretrained layer normalization and attention mechanisms to avoid destabilizing the pretrained weights during AdaLN integration.

Additional Directions. Future work includes scaling to longer sequences via hierarchical diffusion or sliding windows, incorporating language model priors through classifier-free guidance, training universal inversion across multiple embedding models, and adversarial training to develop inversion-resistant embeddings with formal privacy guarantees for security-critical applications.

7 Conclusion

We introduced Conditional Masked Diffusion Language Models for embedding inversion, achieving up to 81.3% token accuracy and 0.87 cosine similarity on 32-token sequences across three embedding models. Masked diffusion enables parallel refinement without autoregressive error accumulation, while AdaLN provides architecture-agnostic conditioning. Systematic comparison of four decoding strategies reveals that sequential greedy maximizes coherence and accuracy, while Euler with re-masking optimizes quality-speed trade-offs. Our results confirm that embeddings leak substantial information about source text, requiring privacy protections equivalent to the original data. The progression from Vec2Text being architecture-specific to ALGEN being alignment-dependent to Zero2Text being API-dependent to our work being architecture-agnostic demonstrates that inversion attacks are becoming more accessible, not less. Future work will scale to longer sequences, incorpo-

rate language model priors, and develop adversarial defenses for inversion-resistant embeddings in security-critical deployments.

Acknowledgments

We thank the Jina AI and Elastic teams for helpful discussions and compute resources.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993, 2021.

Yuxin Chen, Wei Zhang, and Yang Liu. Algen: Few-shot cross-model embedding alignment for embedding inversion. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025. arXiv:2502.11308.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2022.

Hyunwoo Kim, Jinwoo Park, and Seungjae Lee. Unifying masked diffusion: A general framework for generation order in discrete diffusion. *arXiv preprint arXiv:2602.02112*, 2026a.

Jaemin Kim, Seungwoo Park, and Jaehyung Lee. Zero2text: Training-free embedding inversion using llm priors. *arXiv preprint arXiv:2602.01757*, 2026b.

Minghui Li, Zheng Wang, and Pin-Yu Chen. Geia: Generative embedding inversion attack. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14231–14244, 2023. arXiv:2305.03010.

Shengkai Lou, Xin Zhao, and Stefano Ermon. Score entropy discrete diffusion. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. arXiv:2310.16834.

John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Doyen Sahoo, Yutong Chen, Shao-Yen Liu, Hung-yi Hsu, Yingqi Li, Chin-Chia Michael Lee, et al. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, and Han Xiao. Jina embeddings v3: Multilingual embeddings with task lora. `https://arxiv.org/abs/2409.10173`, 2024.

Dimitris Tragoudaras. Reproducing and extending geia: An empirical study. *arXiv preprint arXiv:2504.16609*, 2025.

Yifan Wang, Xiaoming Li, and Hao Zhang. Constrained discrete diffusion for controlled generation. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2503.09790.

Chen Yu, Lei Wang, and Yue Zhang. Discrete diffusion language models: A comprehensive survey. *arXiv preprint arXiv:2506.13759*, 2025.

Tianqi Zhang, Haoran Chen, and Ming Liu. Soft-masked diffusion language models. *arXiv preprint arXiv:2510.17206*, 2025.

A Implementation Details

A.1 Hyperparameters

Table 6: Complete hyperparameter configuration.

Parameter	Value
Transformer layers	8
Hidden dimension	768
Attention heads	12
MLP dimension	3072
Dropout	0.1
Batch size	400
Learning rate	10^{-4}
Weight decay	0.01
EMA decay	0.9999
Noise schedule λ	5.0
Sequence length	32
Training steps	200K

A.2 Computational Requirements

Training on a single A100 GPU takes approximately 48 hours for 200K steps. Inference using sequential decoding takes 150ms per sequence on the same hardware. Euler sampling is significantly faster at approximately 50ms per sequence due to parallel token prediction, though it achieves slightly lower quality. Memory requirements during training peak at 24GB including optimizer states and activations for the batch size of 400. The model checkpoint size is 312MB including both model parameters and EMA weights.

Table 7: Performance milestones during training for jina-v3 encoder. Sequential greedy decoding.

Training Steps	Token Acc.	Exact Match	Cosine Sim.	BLEU
10K	40.2%	1.3%	0.65	22.1
25K	58.7%	5.8%	0.76	35.4
50K	69.7%	12.3%	0.83	45.2
100K	70.8%	13.7%	0.84	46.8
200K	71.2%	14.1%	0.84	47.3