Embedding Inversion via Conditional Masked Diffusion Language Models

Han Xiao
Jina AI by Elastic
han.xiao@jina.ai

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 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 8 forward passes through a 78M-parameter model with no encoder access, versus 20+ iterations through both a 220M model and the target encoder for Vec2Text. We evaluate on three embedding models and compare four decoding strategies. On 32-token sequences, the model achieves 81.3% token accuracy and 0.87 cosine similarity on Qwen3-Embedding-0.6B. Architecture-agnostic inversion without encoder access changes the threat model for deployed embedding systems: any published embedding can be targeted without knowledge of or interaction with the encoder that produced it.

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., 2026] 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 an alternative 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.

Last updated: February 11, 2026

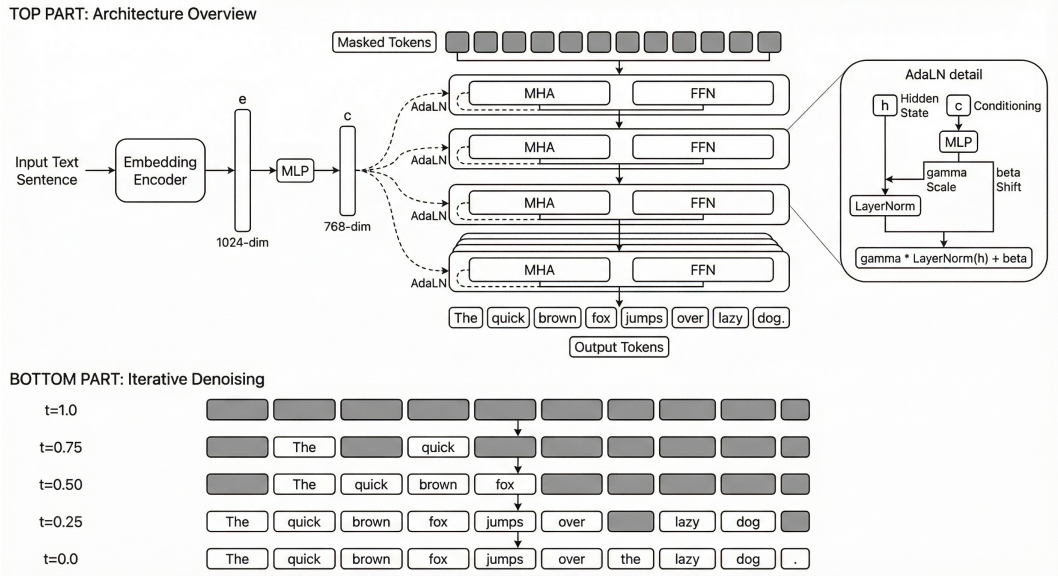


Figure 1: Architecture of the Conditional Masked Diffusion Language Model. The embedding vector is projected and injected into each transformer layer via AdaLN conditioning. The model predicts original tokens at masked positions through iterative denoising.

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 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.

We present the first application of masked diffusion language models to embedding inversion, replacing autoregressive generation and iterative re-embedding with parallel denoising. The approach is encoder-agnostic: we train on three embedding models without alignment training or architecture-specific modifications. We systematically compare four decoding strategies, identifying adaptive re-masking during Euler sampling as the best quality-efficiency trade-off for parallel generation. On 32-token sequences, our 78M-parameter model recovers up to 81.3% of tokens with 0.87 cosine similarity from a single embedding vector, requiring no 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 has advanced rapidly 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., 2026] 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× ROUGE-L improvement over baselines in black-box cross-domain settings. Together, these methods show that embedding inversion generalizes across architectures and data regimes. 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 [Cardei 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

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

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. 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. 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 (Figure 1). 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. 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 $|\mathcal{V}| = 50257$.

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.

4 Experimental Results

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. We evaluate on three embedding models with different architectures and dimensionalities: jina-embeddings-v3 [Sturua et al., 2024] with 570M parameters and 1024-dimensional embeddings, Qwen3-Embedding-0.6B with 600M parameters and 1024-dimensional embeddings, and EmbeddingGemma-300m with 300M parameters and 768-dimensional embeddings. We train separate models for each encoder using multilingual data from mC4 to assess generalization across embedding spaces.

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	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

We compare four decoding strategies at inference time. Sequential greedy decoding iteratively unmasks 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, producing highly coherent text through left-to-right generation but sacrificing the parallel nature of diffusion. Euler sampling uses the Euler method for the reverse diffusion process, starting from $x_1 = [\text{MASK}]^n$ and taking uniform timesteps from $t = 1$ to $t = 0$, sampling from $p_\theta(x_{0,i}|x_t, t, \mathbf{e})$ for all positions simultaneously. Euler with re-masking re-masks positions where $\max_v p_\theta(v|x_t, t, \mathbf{e}) < \tau$ after each Euler step, refining low-confidence predictions in subsequent steps. Two-stage decoding combines sequential and parallel approaches by first generating a hypothesis via sequential greedy decoding, then refining it using Euler sampling initialized at this hypothesis. We use token accuracy, exact match, cosine similarity, BLEU, and perplexity under GPT-2 as evaluation metrics.

4.1 Performance Across Encoders

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 76.0%. All models are trained on multilingual data from mC4.

Table 2 shows detailed results on jina-v3 comparing four decoding strategies. Sequential greedy achieves highest token accuracy at 76.0% and cosine similarity at 0.83. Euler sampling achieves 65.2% accuracy with faster parallel generation.

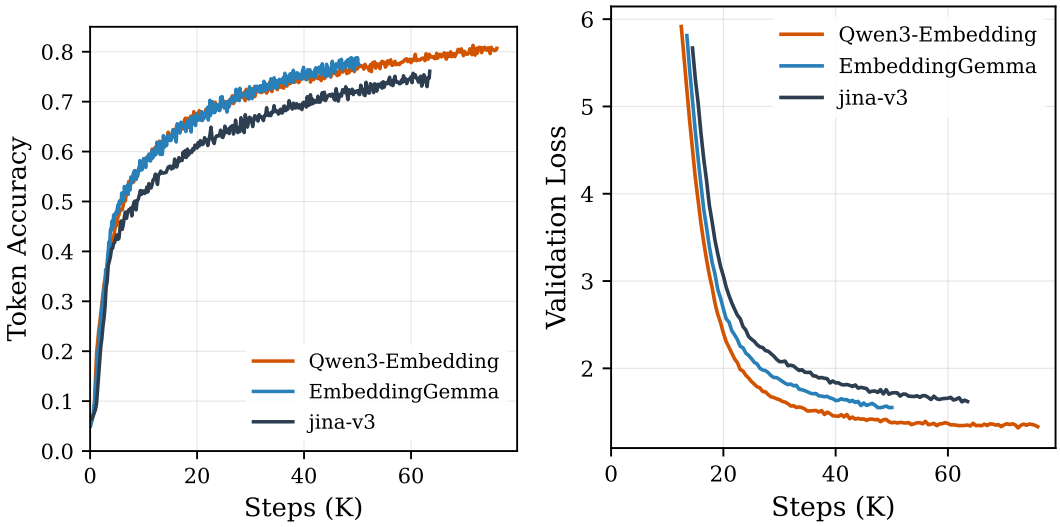
Euler with re-masking at 0.05 improves over vanilla Euler by 2.6 percentage points in token accuracy. Two-stage decoding achieves highest exact match at 13.1%. Baselines confirm that embedding conditioning is essential: random tokens achieve 0.02% accuracy, while unconditional LM achieves 2.1% despite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

4.2 Decoding Strategies and Re-masking

Table 3 shows optimal performance at remask probability 0.05 for Euler sampling with adaptive re-masking. Higher rates discard correct predictions, lower rates provide insufficient correction.

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



(a) Token accuracy

(b) Validation loss

Figure 2: Training dynamics across four embedding encoders. Qwen3-Embedding reaches 81.3% token accuracy at 72.5K steps. Validation loss correlates inversely with accuracy, with Qwen3 achieving the lowest loss at 1.32.

4.3 Training Dynamics

Figure 2 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 (see Table 5 for detailed milestones). Diminishing returns after 50K motivated checkpoint selection.

5 Conclusion

We presented embedding inversion via conditional masked diffusion, achieving up to 81.3% token accuracy and 0.87 cosine similarity across three embedding models with a single 78M-parameter decoder. Unlike prior methods that require access to the target encoder at inference time, our approach conditions solely through AdaLN modulation, making it architecture-agnostic. The progression from Vec2Text (architecture-specific) to ALGEN (alignment-dependent) to Zero2Text (API-dependent) to our work (architecture-agnostic) shows that the barrier to mounting inversion attacks is steadily decreasing. Organizations deploying embedding-based systems should treat embeddings as sensitive data requiring protection equivalent to the original text. Current limitations include the 32-token sequence constraint and diminishing returns beyond 50K-70K training steps, suggesting that initializing from pretrained masked language models such as ModernBERT may provide stronger language priors than training from scratch.

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.

Michael Cardei, Jacob K. Christopher, Thomas Hartvigsen, Brian R. Bartoldson, Bhavya Kailkhura, and Ferdinando Fioretto. Constrained language generation with discrete diffusion models. *arXiv preprint arXiv:2503.09790*, 2025.

Yiyi Chen, Qiongkai Xu, and Johannes Bjerva. Algen: Few-shot inversion attacks on textual embeddings via cross-model alignment and generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

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.

Doohyun Kim, Donghwa Kang, Kyungjae Lee, Hyeongboo Baek, and Brent Byunghoon Kang. Zero2text: Zero-training cross-domain inversion attacks on textual embeddings. *arXiv preprint arXiv:2602.01757*, 2026.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848, 2024.

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.

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 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, Nan Wang, and Han Xiao. jina-embeddings-v3: Multilingual embeddings with task lora, 2024.

A Implementation Details

A.1 Hyperparameters

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 4: 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

Table 5: 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