One Latent, Many Tokens: Jointly Learning Compressed Embeddings for Efficient Language Diffusion

Anonymous authors
Paper under double-blind review

Abstract

Most continuous diffusion language models process one latent position per token at each sampling step, making generation expensive. Two-stage methods lower the cost by reducing the latent length, but they fix the compressed embedding space before training the diffusion model. Embeddings from the fixed space can be difficult to model with diffusion and decode reliably into tokens, which limits generation quality after compression. To address this problem, we introduce JPEG-DLM (Joint-embedding Prediction for Efficient Generation with Diffusion Language Model), which jointly trains a compressor, a flow matching model and a decoding module. With joint-embedding prediction, JPEG-DLM learns compressed embeddings that are more structured, easier to model with diffusion and reliably decodable into tokens. JPEG-DLM achieves the lowest mean Gen-PPL and highest throughput among recent diffusion and flow models on LM1B and OWT. At a compression rate of 0.5 on OWT, it reaches a Gen-PPL of 34.52 and approximately $2.3\times$ ELF's throughput. These results suggest that jointly learning compressed embeddings offers a promising path toward efficient diffusion language modeling. We include our implementation code in the supplementary material.

1 Introduction

Diffusion language models (DLMs) generate text by iteratively refining a corrupted sequence and can update multiple token positions in parallel (Nie et al., 2025; Ye et al., 2025). Existing methods operate either in discrete token space (Austin et al., 2021; Lou et al., 2024) or continuous representation space (Li et al., 2022; Hu et al., 2026). Masked diffusion, a prominent discrete approach, has achieved competitive language modeling performance, including at large-model scale (Sahoo et al., 2024; Shi et al., 2024; Nie et al., 2025). Continuous DLMs instead encode tokens as continuous features, connect the data and noise distributions through a corruption process or conditional probability path (Li et al., 2022; Lipman et al., 2023), and decode the generated representation back into tokens (Li et al., 2022; Han et al., 2023; Hu et al., 2026; Lovelace et al., 2023). Continuous states support direct gradient-based guidance (Li et al., 2022), while learned latents can aggregate context across token positions (Guo et al., 2026). The same flexibility underlies latent image diffusion (Rombach et al., 2022) and joint text–image diffusion (Bao et al., 2023).

Parallel updates do not, by themselves, make generation inexpensive. Autoregressive language models generate one token per step and are inherently sequential (Radford et al., 2019), whereas discrete DLMs can update multiple positions (Sahoo et al., 2024; Nie et al., 2025). Most continuous DLMs, however, assign one latent position to each token (Li et al., 2022; Hu et al., 2026; Chen et al., 2026). Because self-attention computation grows quadratically with sequence length, reducing latent length provides quadratic savings in attention computation at each model evaluation. However, stronger compression can make it harder to preserve generation quality. This raises the question of how to reduce the latent length used for generation without substantially compromising generation quality.

Prior work (Meshchaninov et al., 2025; Liang et al., 2026) on latent-length compression for diffusion language models uses two-stage training. It first trains an encoder and decoder to reconstruct text from compressed embeddings and then trains a generative model on the fixed embedding distribution. A fixed embedding space can limit how well the generative model learns the text distribution. Recent work (Meshchaninov et al., 2026) instead jointly trains the latent encoder, diffusion model,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

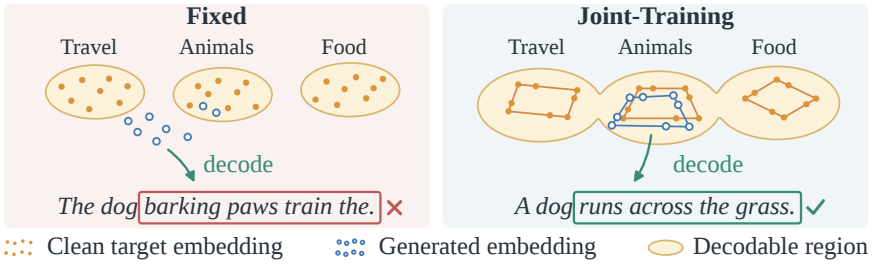


Figure 1: Schematic comparison of fixed and jointly learned compressed spaces, showing more structured embeddings and broader decodable regions with joint training.

and decoder, so the generative objective also updates the representation. A natural idea, as illustrated in Figure 1, is therefore to learn compressed embeddings jointly with the flow matching model so that they support generation and remain reliably decodable into tokens.

Inspired by this insight, we introduce JPEG-DLM, which jointly trains a compressor, a flow matching model, and a decoding module for efficient text generation with diffusion language models. In particular, a fixed compressed embedding space does not ensure that clean embeddings can be recovered from corrupted compressed views. Drawing on JEPA (LeCun, 2022; Assran et al., 2023), which predicts one embedding from another, joint-embedding prediction (JEP) trains the context compressor and shared-weight network to recover the clean target embedding from a corrupted compressed view. In the denoise branch, flow matching (Lipman et al., 2023) trains the shared-weight network to recover clean embeddings from noisy embeddings formed by interpolating clean target embeddings with Gaussian noise. The clean-embedding prediction from the decode branch then enters the decoding module. Feature reconstruction and token-level cross-entropy train this module to map the predicted clean embedding to tokens reliably. Compared with methods that use a fixed compressed embedding space, JPEG-DLM jointly learns the compressed embeddings with the generative model and achieves a marked improvement in text generation quality. Experiments on LM1B and OWT further show that it achieves the lowest Gen-PPL and highest throughput among diffusion and flow models.

The main contributions of this paper are as follows:

- We introduce JPEG-DLM, which learns latent-length compression jointly with generation. This yields more structured compressed embeddings that are easier to model with diffusion and decode into tokens reliably.
- We use JEP to predict clean target embeddings from corrupted compressed views, training the compressor to preserve structure that remains predictable under corruption.
- Under latent-length compression, JPEG-DLM reaches generation quality comparable to or substantially better than the baselines on LM1B and OWT while achieving the highest generation throughput.

2 Related Work

2.1 Diffusion and Flow Language Models

Diffusion language models generate text by iteratively transforming a corrupted state into a clean sample. Discrete methods define this process directly over token categories, using either general transition matrices (Austin et al., 2021; Lou et al., 2024) or an absorbing mask state (Shi et al., 2024; Sahoo et al., 2024). In masked diffusion, a forward process replaces tokens with a mask symbol, while generation uses learned reverse transition probabilities to recover masked positions iteratively. The generative state therefore remains categorical at every token position throughout sampling. Continuous methods instead map tokens into continuous representations and denoise in the resulting space, using token embeddings in Diffusion-LM (Li et al., 2022) and Plaid (Gulrajani & Hashimoto, 2023), or a simplex defined over the vocabulary in SSD-LM (Han et al., 2023) and TESS (Karimi Mahabadi et al., 2024). Flow matching (Lipman et al., 2023) defines a probability path between the noise and data distributions, and learns the velocity field of this path at every continuous time t. An ODE solver then integrates this field to transport noise into the embedding distribution. Recent flow models learn

their own token embeddings in LangFlow (Chen et al., 2026), or reuse contextual features from a frozen encoder in ELF (Hu et al., 2026), and reach generation quality that rivals discrete diffusion. Like these flow models, JPEG-DLM uses flow matching to generate continuous embeddings, which are then decoded into tokens.

2.2 Efficient Generation in Diffusion Language Models

A single forward pass of a discrete diffusion model predicts all masked positions at once, so a sampler can unmask the positions it is most confident about and leave the rest masked for later steps. Decoding methods for these models select which positions to commit at each step (Yuan et al., 2026; Qi et al., 2026), and committing several positions per pass raises throughput while keeping task accuracy close to the backbone (Wu et al., 2026). Continuous models decode only after the final state is reached (Hu et al., 2026; Meshchaninov et al., 2026), so no position is fixed while the sequence is still being refined, and this form of parallel commitment is not suited to them. Their generation cost instead follows the number of latent positions processed at each step, so a shorter latent sequence offers a means of reducing it (Lovelace et al., 2023; Liang et al., 2026).

LD4LG (Lovelace et al., 2023) compresses frozen encoder features into a fixed-length latent sequence with a Perceiver resampler, and PLANNER (Zhang et al., 2023) encodes a paragraph into 16 latent codes with a variational encoder. Both decode text with a pretrained autoregressive decoder, so generation depends on the preceding tokens as well as the latent representation. COSMOS (Meshchaninov et al., 2025) removes this dependence by training a Perceiver-resampler autoencoder whose decompressor and token projection decode the sequence directly. AURORA-LM (Liang et al., 2026) builds an ordered latent sequence with a causal autoencoder and sets its latent length as a fixed fraction of the token length. In all of these methods the embedding space is fixed before the diffusion model is trained, and such a space may limit the generation quality that can be reached. JPEG-DLM instead learns the compressed embedding space jointly with the flow matching model and the decoding module to reach better generation quality.

2.3 Learning an Embedding Space with Joint-Embedding Prediction

Latent diffusion language models depend on the quality of their embedding space (Meshchaninov et al., 2026). Joint-embedding predictive architectures (LeCun, 2022) approach this as a representation learning problem. They first corrupt an input by masking part of it or adding noise. A context encoder maps the corrupted view to a context embedding. A predictor then uses this context embedding to predict the embedding produced by a target encoder for the clean input (Assran et al., 2023; Baevski et al., 2022). The loss is thus computed in embedding space, not input space. The target encoder is an exponential moving average of the context encoder, and its output enters the loss with a stop-gradient. Without this asymmetry the objective admits a trivial solution in which the encoder produces a constant representation regardless of its input (Bardes et al., 2024). These architectures have been instantiated across modalities, including speech, vision and language in data2vec (Baevski et al., 2022), video in V-JEPA (Bardes et al., 2024), image generation in D-JEPA (Chen et al., 2025), and text in LLM-JEPA (Huang et al., 2026).

JPEG-DLM uses joint-embedding prediction to learn a compressed embedding space and trains a flow matching model to generate from it. Recent methods, such as LDLM (Meshchaninov et al., 2026) and Cola DLM (Guo et al., 2026), also train the latent space together with its generative model. In both, the generative objective reaches the encoder at every noise level, and each adds a regularizer to keep the representation from drifting. JPEG-DLM updates the compressor through the $t = 1$ decode branch, while the denoise branch trains the shared-weight network using stop-gradient compressed embeddings as targets. At high noise levels, the predicted clean embedding approaches the data mean and carries little information about individual tokens (Karras et al., 2022). Token-level cross-entropy on such a prediction would drive large gradients through the token projection and destabilize training.

3 Method

We introduce JPEG-DLM, a joint training method for efficient text generation in a continuous latent space with reduced latent length. As illustrated in Figure 2, JPEG-DLM uses joint-embedding prediction to shape the compressed latent space, applies flow matching to generate compressed em-

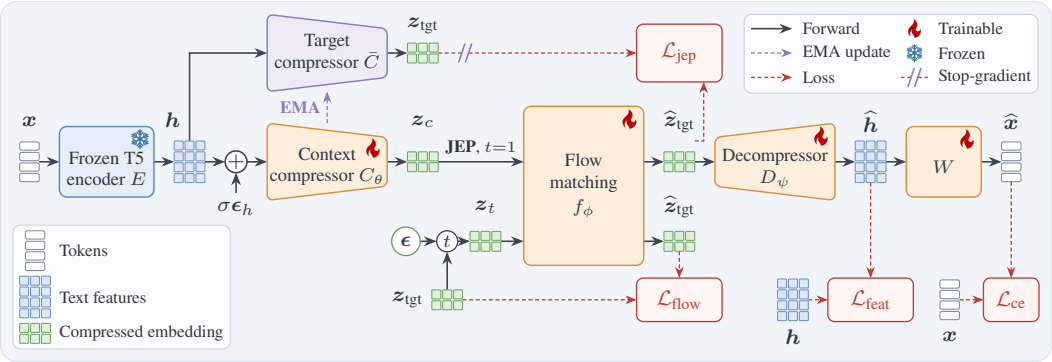


Figure 2: Overview of JPEG-DLM. The frozen T5 encoder maps tokens to features. The context compressor maps a corrupted view to $\boldsymbol{z}_c$, while the target compressor provides the stop-gradient clean target embedding $\boldsymbol{z}_{\text{tgt}}$ and is updated by an exponential moving average. The shared-weight network predicts the clean target embedding from $\boldsymbol{z}_c$ for JEP and from the noisy embedding $\boldsymbol{z}_t$ for flow matching. The decoding module maps the predicted clean embedding back to features and tokens using $\mathcal{L}_{\text{feat}}$ and $\mathcal{L}_{\text{ce}}$.

beddings, and trains a decoding module to map predicted clean embeddings to tokens. Section 3.1 first introduces the background of latent diffusion and flow matching. Section 3.2 then explains how joint-embedding prediction shapes the compressed latent space under latent-length compression. Section 3.3 presents joint training for embedding generation and reliable token decoding.

3.1 Preliminaries

Latent diffusion models move iterative generation from the original data space to a continuous latent representation (Rombach et al., 2022). For text generation, an encoder maps a token sequence $\boldsymbol{x}$ to a continuous latent $\boldsymbol{z}_1$, a latent generative model learns the distribution of $\boldsymbol{z}_1$, and a decoding module maps generated latents back to tokens (Lovelace et al., 2023). We write $\boldsymbol{z}_1 \in \mathbb{R}^{N\times d}$, where N denotes latent length and d denotes latent width. When $N < L$, the generative model processes fewer latent positions than the original token sequence at each sampling step.

We use flow matching (Lipman et al., 2023) to learn the latent distribution by transporting a simple noise distribution p_0 to the latent data distribution p_1 along a continuous probability path. Given a clean embedding $\boldsymbol{z}_1 \sim p_1$ and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$, we construct the noisy embedding $\boldsymbol{z}_t$ by linear interpolation,

$$\boldsymbol{z}_t = t\boldsymbol{z}_1 + (1-t)\boldsymbol{\epsilon}, \qquad t \in [0,1], \tag{1}$$

Here, $t = 0$ corresponds to Gaussian noise and $t = 1$ corresponds to the clean embedding. Following JiT (Li & He, 2026), we use x-prediction, where the flow matching model f_ϕ predicts the clean embedding $\boldsymbol{z}_1$ from the noisy embedding $\boldsymbol{z}_t$ instead of directly regressing the velocity. With $t \sim \pi$, where π is a logit-normal distribution over t, we train this embedding prediction using the mean squared error $\|\cdot\|_2^2$ over positions and channels

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\substack{\boldsymbol{z}_1\sim p_1,\ t\sim\pi,\\ \boldsymbol{\epsilon}\sim\mathcal{N}(0,\boldsymbol{I})}} \left[\|f_\phi(\boldsymbol{z}_t, t) - \boldsymbol{z}_1\|_2^2 \right], \tag{2}$$

At inference, an Euler ODE solver (Li & He, 2026) performs K evaluations of f_ϕ to evolve Gaussian noise to the terminal state. We then evaluate f_ϕ once at $t = 1$ to obtain the clean embedding for decoding, which the decoding module maps back to the token sequence $\boldsymbol{x}$. A detailed derivation of the induced velocity field is provided in Appendix A.1.5, and the sampling FLOPs under latent-length compression are reported in Table 5.

3.2 Joint-Embedding Prediction with Latent-Length Compression

Our first goal is to learn a compressed latent space that reduces latent length, preserves the information needed for reliable token decoding, and allows clean embeddings to be predicted under corruption. A frozen pretrained T5 encoder (Raffel et al., 2020) maps the token sequence to the feature matrix $\boldsymbol{h} = E(\boldsymbol{x}) \in \mathbb{R}^{L\times d}$. We use a Perceiver-style compressor (Alayrac et al., 2022) that maps its L token

Algorithm 1 JPEG-DLM: training.

```
# Input: x; frozen E; σ, π
# Models: C_θ, C̄, f_φ, D_ψ, W
h ← E(x)
z_tgt ← sg(C̄(h))

# decode branch (t = 1)
ε_h ~ N(0, I)
z_c ← C_θ(h + σ · ε_h)
ẑ_tgt ← f_φ(z_c, t=1)
L_jep ← ||ẑ_tgt − z_tgt||_2^2
ĥ ← D_ψ(ẑ_tgt)
L_feat ← ||ĥ − sg(h)||_2^2
ℓ ← W(ĥ)
L_ce ← CE(ℓ, x)

# denoise branch
t ~ π;  ε ~ N(0, I)
z_t ← t z_tgt + (1 − t)ε
L_flow ← ||f_φ(z_t, t) − z_tgt||_2^2
```

Algorithm 2 JPEG-DLM: inference.

```
# Models: trained f_φ, D_ψ, W
# State (N, d); times t_0 = 0, ..., t_K = 1
z ~ N(0, I)

# start sampling
for k = 0 to K − 1 do
  ẑ_1 ← f_φ(z, t=t_k)
  v ← (ẑ_1 − z)/(1 − t_k)
  z ← z + (t_{k+1} − t_k)v
end for

# predict clean embedding at t = 1
ẑ_dec ← f_φ(z, t=1)

# decode tokens
ĥ ← D_ψ(ẑ_dec)
ℓ ← W(ĥ)
x̂_i ← arg max_{v∈V} ℓ_{i,v},   i = 1, ..., L
```

positions to N latent positions. Specifically, N learned latent queries cross-attend to all L token-level features, and the resulting query states form the compressed embedding $\boldsymbol{z} \in \mathbb{R}^{N \times d}$. We define the latent-length compression rate as $r = N/L$, where $r = 1.0$ preserves one latent position per token and a smaller value indicates stronger compression. To restore token-length features, the decompressor D_ψ uses L token-position queries to cross-attend to all N compressed positions, allowing each restored position to use information from the complete compressed embedding. Because both mappings attend over their complete inputs, the compressor and decompressor process the sequence bidirectionally.

However, the architecture of the compressor and decompressor does not by itself learn a latent space in which clean embeddings can be predicted under corruption. Following JEPA (LeCun, 2022; Assran et al., 2023), we train a predictor to recover a clean compressed embedding from a corrupted compressed view. Specifically, we retain $\boldsymbol{h}$ as the clean feature matrix and form the corrupted view $\widetilde{\boldsymbol{h}} = \boldsymbol{h} + \sigma\boldsymbol{\epsilon}_h$, where $\boldsymbol{\epsilon}_h \sim \mathcal{N}(0, \boldsymbol{I})$, after which the context compressor maps $\widetilde{\boldsymbol{h}}$ to the compressed embedding $\boldsymbol{z}_c = C_\theta(\widetilde{\boldsymbol{h}})$. The predictor estimates the corresponding clean compressed embedding $\widehat{\boldsymbol{z}}_{\text{tgt}}$ from $\boldsymbol{z}_c$, while its stop-gradient target $\text{sg}(\bar{C}(\boldsymbol{h}))$ is computed by a copy $\bar{C}$ whose parameters are updated as an exponential moving average of C_θ to prevent representation collapse (Assran et al., 2023; Bardes et al., 2024). We define the JEP loss as the mean squared error between the predicted embedding and its stop-gradient target.

$$\mathcal{L}_{\text{jep}} = \left\| \widehat{\boldsymbol{z}}_{\text{tgt}} - \text{sg}(\bar{C}(\boldsymbol{h})) \right\|_2^2 . \tag{3}$$

This loss updates the predictor and context compressor, encouraging $\boldsymbol{z}_c$ to preserve the information needed to recover the clean target embedding under corruption. Overall, we use joint-embedding prediction to learn a compressed latent space whose representations have reduced latent length and allow clean embeddings to be predicted under corruption.

3.3 Joint Training for Embedding Generation and Reliable Token Decoding

We use the flow matching model f_ϕ in the denoise branch to model the distribution of the compressed embeddings learned in Section 3.2. The denoise branch samples $t \sim \pi$, constructs the noisy embedding $\boldsymbol{z}_t$ using the interpolation in Equation 1, and minimizes $\mathcal{L}_{\text{flow}}$ in Equation 2 to train f_ϕ to recover the clean target embedding $\boldsymbol{z}_{\text{tgt}} = \text{sg}(\bar{C}(\boldsymbol{h}))$.

To make the generated compressed embeddings easier to decode into tokens, we follow ELF (Hu et al., 2026) and share the parameters of f_ϕ between the JEP predictor and the flow matching model with x-prediction. Model-mode tokens identify the denoise and decode branches, which use separate output heads. At $t = 1$, the decode branch receives $\boldsymbol{z}_c$, whose corruption is controlled by σ independently of the flow time. It predicts the clean target embedding $\widehat{\boldsymbol{z}}_{\text{tgt}} = f_\phi(\boldsymbol{z}_c, t{=}1)$, which is supervised by $\mathcal{L}_{\text{jep}}$ in Equation 3. The decoding module then receives $\widehat{\boldsymbol{z}}_{\text{tgt}}$ and produces token logits $\boldsymbol{\ell}$. We apply

token-level cross-entropy to these logits only in the decode branch at $t = 1$ and train the decoding module with

$$\mathcal{L}_{\text{ce}} = -\frac{1}{L}\sum_{i=1}^{L} \log \frac{\exp(\ell_{i,x_i})}{\sum_{v\in\mathcal{V}} \exp(\ell_{i,v})}. \tag{4}$$

Here, $\mathcal{V}$ denotes the token vocabulary. Once ODE sampling reaches the final state, we evaluate $f_\phi(\cdot, 1)$ once in the decode branch, and the decoding module receives the predicted clean embedding. This gives the decoding module the same type of input during training and inference.

We first use the decompressor D_ψ to restore the token-length feature matrix $\widehat{\boldsymbol{h}} = D_\psi(\widehat{\boldsymbol{z}}_{\text{tgt}})$. A token projection W then maps these features to token logits $\boldsymbol{\ell} = W(\widehat{\boldsymbol{h}})$. The projection consists of two linear layers with a GELU activation between them. We supervise the decompressed features directly with a feature reconstruction loss before applying the token projection

$$\mathcal{L}_{\text{feat}} = \left\| \widehat{\boldsymbol{h}} - \text{sg}(\boldsymbol{h}) \right\|_2^2. \tag{5}$$

By supervising $\widehat{\boldsymbol{h}}$ before the token projection, $\mathcal{L}_{\text{feat}}$ provides a direct training signal to the decompressor.

Accordingly, $\mathcal{L}_{\text{flow}}$ trains the model to generate compressed embeddings, $\mathcal{L}_{\text{jep}}$ shapes the compressed representation, $\mathcal{L}_{\text{ce}}$ supervises token decoding, and $\mathcal{L}_{\text{feat}}$ stabilizes feature reconstruction. We sum these four losses to obtain the main objective

$$\mathcal{L}_{\text{main}} = \mathcal{L}_{\text{jep}} + \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{ce}}. \tag{6}$$

We also use additional regularization losses described in Appendix A.1.3. The denoise branch updates f_ϕ through $\mathcal{L}_{\text{flow}}$ using stop-gradient target embeddings. The decode branch updates C_θ and f_ϕ through $\mathcal{L}_{\text{jep}}$, $\mathcal{L}_{\text{feat}}$, and $\mathcal{L}_{\text{ce}}$. The feature and token losses also update the decoding module.

For further details of the training and inference procedures, see the pseudocode in Algorithms 1 and 2, respectively.

4 Experiments

Datasets. We evaluate unconditional generation on the One Billion Word Benchmark (LM1B) (Chelba et al., 2014) and OpenWebText (OWT) (Gokaslan & Cohen, 2019), using sequence lengths of 128 and 1024, respectively. We also use OWT with a sequence length of 128 (OWT-128) to analyze compressed latent representations and conduct ablation studies.

Evaluation. We report generative perplexity (Gen-PPL), the perplexity of generated text under a pretrained GPT-2 Large (Radford et al., 2019), and average unigram entropy as a measure of diversity. A low Gen-PPL can also come from degenerate text (Franca & Tong, 2026), so we report MAUVE (Pillutla et al., 2021), which compares the generated and reference distributions and reflects generation quality more faithfully. The main comparison also reports generation speedup relative to the AR baseline on each dataset.

Models and baselines. JPEG-DLM uses a frozen T5-small encoder (Raffel et al., 2020), a six-layer compressor, and a six-layer decompressor with latent width $d = 512$. We use a DiT (Peebles & Xie, 2023) as the flow matching model. The DiT and token projection have 105M parameters. We compare JPEG-DLM with an autoregressive Transformer (AR), the discrete diffusion models MDLM (Sahoo et al., 2024), SEDD-Absorb (Lou et al., 2024), and Duo (Sahoo et al., 2025), the flow matching models FLM (Lee et al., 2026), LangFlow (Chen et al., 2026), and ELF (Hu et al., 2026), and the two-stage latent diffusion model COSMOS (Meshchaninov et al., 2025).

Training and inference. We train JPEG-DLM in bfloat16 with the AdamW optimizer (Loshchilov & Hutter, 2019). The flow matching model uses a learning rate of 1×10^{-3} on both datasets. The compressor and decompressor both use 3×10^{-4} on LM1B and 2×10^{-4} on OWT. During inference, we use self-conditioning (Chen et al., 2023) and the Euler ODE sampler for 32 steps on both datasets, followed by greedy token decoding. Appendix A.4 lists the model configurations, training schedules, and evaluation details.

4.1 Main Comparison

JPEG-DLM uses $r = 0.25$ on LM1B and $r = 0.5$ on OWT, reducing 128 and 1024 token positions to 32 and 512 latent positions, respectively. We report the mean over five seeds, with 1,024 samples per seed. MDLM, SEDD-Absorb, Duo, FLM, and LangFlow use a common sampling budget of 128 steps on LM1B and 1024 steps on OWT. ELF (Hu et al., 2026) reports its main unconditional result with 32 sampling steps. We retain this step count with its ODE sampler, and JPEG-DLM also uses 32-step ODE sampling.

On both datasets, JPEG-DLM achieves the lowest mean Gen-PPL and the highest throughput among the diffusion and flow models in Table 1. We use AR without KV cache as the speed reference. At 32 ODE sampling steps, JPEG-DLM achieves lower Gen-PPL than ELF. It also uses fewer sampling steps than the remaining diffusion and flow models. On OWT, JPEG-DLM reaches $2.31\times$ the throughput of ELF and $4.69\times$ that of AR with KV cache.

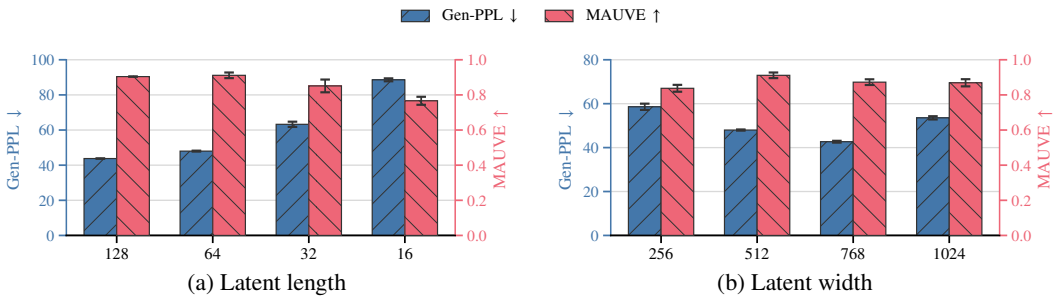
Table 1: Unconditional generation on LM1B and OWT. Speed is relative to AR without KV cache. Among diffusion and flow models, **bold** and underlined mark the best and second-best Gen-PPL, MAUVE, and speed. JPEG-DLM has 105M DiT and token projection parameters, or 154M including the compressor and decompressor.

		LM1B ($L = 128$)				**OWT** ($L = 1024$)			
Method	**Param. (M)**	**Gen-PPL** ↓	**Ent.**	**MAUVE** ↑	**Speed** ↑	**Gen-PPL** ↓	**Ent.**	**MAUVE** ↑	**Speed** ↑
AR (Radford et al., 2019)	130	151.36	4.397	0.947	1.00× (1.62× w/ KV cache)	41.06	5.583	0.930	1.00× (7.14× w/ KV cache)
MDLM (Sahoo et al., 2024)	130	192.80	4.410	0.923	0.52×	41.46	5.287	0.755	0.32×
SEDD-Absorb (Lou et al., 2024)	170	182.57	4.405	0.941	0.40×	41.15	5.260	0.728	0.25×
Duo (Sahoo et al., 2025)	130	161.80	4.382	0.941	0.41×	75.88	5.536	<u>0.903</u>	0.15×
FLM (Lee et al., 2026)	179	176.74	4.402	0.863	0.66×	61.37	5.335	0.373	0.21×
LangFlow (Chen et al., 2026)	130	<u>141.35</u>	4.395	0.935	0.72×	<u>36.48</u>	5.244	**0.922**	0.28×
ELF (Hu et al., 2026)	105	148.76	4.344	<u>0.948</u>	<u>4.01×</u>	45.50	5.307	0.842	<u>14.50×</u>
JPEG-DLM (Ours)	105 (154)	**96.76**	4.263	**0.951**	**9.13×**	**34.52**	5.067	0.801	**33.49×**

4.2 Analysis of Compressed Latent Representations

We study latent-length compression on OWT-128 at $d = 512$. Figure 3(a) shows that generation quality declines only moderately as compression becomes more aggressive. The modest degradation indicates that JPEG-DLM maintains strong generation quality under strong compression.

We next vary the latent width at $r = 0.5$. As shown in Figure 3(b), increasing the latent width from 256 to 512 substantially improves generation quality. This indicates that a small latent width limits representation capacity. Beyond 512, however, further increasing the latent width does not consistently improve performance.



(a) Latent length (b) Latent width

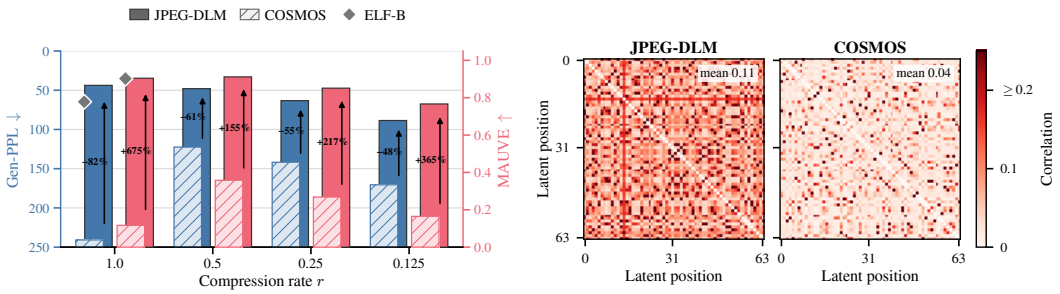
Figure 3: Generation quality at different latent lengths (a) and latent widths (b).

4.3 Joint Training vs Two-Stage Training

In this section, all experiments are conducted on OWT-128. We compare two approaches to generation in a compressed latent space. JPEG-DLM jointly trains the compressor, flow matching model, and decoding module. COSMOS first trains an autoencoder and then fits a diffusion model to its fixed embeddings. We retrain COSMOS on the same data with the T5 tokenizer and frozen T5-small encoder used by JPEG-DLM. Its compressor and decompressor have the same architectures as ours.

We evaluate JPEG-DLM and COSMOS at $r \in \{1.0, 0.5, 0.25, 0.125\}$. Following its default implementation, COSMOS uses final-layer T5-small features at every compression rate. Appendix A.4.6 analyzes its unusually poor result at $r = 1.0$. We also train ELF-B (Hu et al., 2026) on the same data as an uncompressed reference.

Generation quality. As shown in Figure 4(a), JPEG-DLM achieves lower Gen-PPL and higher MAUVE than COSMOS at every compression rate, using 64 rather than 200 sampling steps. It also achieves lower Gen-PPL than the uncompressed ELF-B reference, which uses 64 steps, at $r \in \{1.0, 0.5, 0.25\}$. These results show that joint training yields better generation quality at different compression rates.



(a) Generation quality (b) Global structure

Figure 4: Joint and two-stage training. (a) Gen-PPL and MAUVE across compression rates; arrows show relative changes from COSMOS to JPEG-DLM, and diamonds mark ELF-B. COSMOS uses final-layer T5-small features at every rate. (b) Correlation between latent positions at $r = 0.5$.

Global structure. To understand why joint training yields better generation quality, we analyze the global structure of the compressed embeddings. Correlation between latent positions measures how similarly they vary across texts. Higher correlations indicate a clearer structure that is easier for a generative model to learn. At $r = 0.5$, we compute pairwise correlations among the 64 positions. Figure 4(b) shows correlations in both embeddings, but the mean off-diagonal correlation is higher for JPEG-DLM (0.105) than for COSMOS (0.039). Joint training therefore produces a compressed embedding with stronger global structure, helping explain the better generation quality in Figure 4(a). Further details are provided in Appendix A.4.6.

Reliable decoding. During generation, the decoding module receives predicted clean embeddings rather than clean target embeddings. Prediction errors can move these embeddings away from the clean target embedding space, so the decoding module should still decode them into high-quality text. At a compression rate of $r = 0.5$, we construct interpolation paths between pairs of clean target embeddings and evaluate each path directly and after adding noise at three strengths. Interpolation traces a controlled path through the embedding space, while the added noise creates progressively larger deviations from clean target embeddings. Figure 5 shows that COSMOS decodes high-quality text at the endpoints, but its decoding quality deteriorates sharply near the midpoint. As noise increases, poor decoding extends across a larger part of the interpolation path. In contrast, the decoding quality of JPEG-DLM varies smoothly along the path and remains higher near the midpoint at every noise level. These results show that JPEG-DLM maintains a broader decodable region between clean target embeddings. This pattern suggests greater tolerance to deviations that can arise in generated compressed embeddings. Further details are provided in Appendix A.4.6.

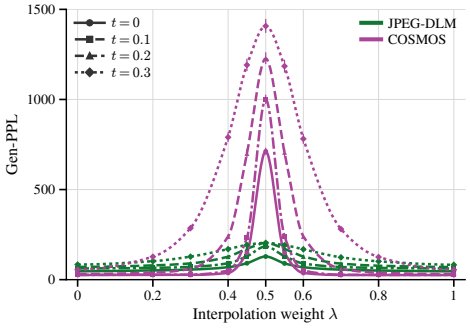


Figure 5: Gen-PPL over interpolations between 1,024 pairs of clean target embeddings at a compression rate of $r = 0.5$, without added noise and at three noise levels. Here, t denotes the COSMOS forward time, with $t = 0$ indicating no added noise. For the three noisy settings, JPEG-DLM uses noisy embeddings at the same SNRs along its linear path.

4.4 Ablation Studies

We ablate the main design choices on OWT-128. All runs use $r = 0.5$, latent width $d = 512$, a 64-step Euler ODE sampler, and self-conditioning (Chen et al., 2023) at scale 3.0. The shaded row of Table 2 is the reference point for these studies. Appendix A.4.6 gives the full OWT-128 configuration.

Shared-weight network. We share f_ϕ between the denoise and decode branches so that flow matching can indirectly shape the compressed embedding space. Retaining JEP but assigning separate networks to the two branches substantially reduces generation quality, showing that weight sharing helps the compressor learn embeddings that are better suited to flow matching.

JEP loss and noise scale. $\mathcal{L}_{\text{jep}}$ directly trains the context compressor and f_ϕ in the decode branch to predict clean target embeddings from corrupted compressed views, shaping a compressed embedding space that supports reliable prediction under corruption. With shared weights, $\mathcal{L}_{\text{jep}}$ and $\mathcal{L}_{\text{flow}}$ use different inputs, but both train the same f_ϕ to predict the clean target embedding. Removing JEP alone therefore causes a modest decline because $\mathcal{L}_{\text{flow}}$ continues to train the parameters used by the decode branch to predict clean target embeddings. With separate networks, $\mathcal{L}_{\text{flow}}$ no longer provides this supervision to the decode branch. Removing JEP then causes generation quality to drop sharply, showing that JEP is critical to JPEG-DLM.

We further study the corruption noise scale σ within JEP. A small scale provides too little pressure to predict clean embeddings robustly, leaving a narrow decodable region. A larger scale exposes the decode branch to larger deviations and can broaden this region, but excessive noise removes information needed to predict the clean embedding. Table 2 shows this balance. The intermediate scale $\sigma = 1.0$ performs best, while both $\sigma = 0.5$ and $\sigma = 1.5$ reduce generation quality.

Feature reconstruction loss. Token-level cross-entropy supervises text output after token projection, whereas $\mathcal{L}_{\text{feat}}$ directly supervises the decompressed token-length features. Removing $\mathcal{L}_{\text{feat}}$ slows convergence and reduces final generation quality, showing that feature reconstruction stabilizes joint training and improves decoding quality.

Timesteps for the token-level cross-entropy. We apply $\mathcal{L}_{\text{ce}}$ only to the clean embedding predicted by the decode branch at $t = 1$. Extending this loss to embeddings predicted by the denoise branch at $t \in [0.5, 1]$ substantially reduces generation quality. These embeddings contain less token-level information, so direct token supervision at intermediate noisy states disrupts the compressed representation. We therefore apply token-level supervision only at $t = 1$.

Table 2: Ablation study on OWT-128. Base uses a shared-weight network.

Variant	$\mathcal{L}_{\text{jep}}$	$\mathcal{L}_{\text{feat}}$	σ	$\mathcal{L}_{\text{ce}}$ timesteps	Gen-PPL ↓	Ent.	MAUVE ↑
Base	✓	✓	1.0	$t = 1$	**48.00**	4.064	**0.912**
sep. nets	✓	✓	1.0	$t = 1$	92.45	4.117	0.786
w/o $\mathcal{L}_{\text{jep}}$	×	✓	1.0	$t = 1$	51.65	4.016	0.849
w/o $\mathcal{L}_{\text{jep}}$ + sep. nets	×	✓	1.0	$t = 1$	131.94	4.188	0.632
w/o $\mathcal{L}_{\text{feat}}$	✓	×	1.0	$t = 1$	51.57	4.069	0.825
$\sigma = 0.5$	✓	✓	0.5	$t = 1$	61.43	3.993	0.817
$\sigma = 1.5$	✓	✓	1.5	$t = 1$	92.78	4.046	0.545
$\mathcal{L}_{\text{ce}}$ at $t \in [0.5, 1]$	✓	✓	1.0	$t \in [0.5, 1]$	146.80	4.140	0.645

5 Conclusion

We introduced JPEG-DLM, which jointly trains a compressor, a flow matching model, and a decoding module to generate text with reduced latent length. JEP shapes the compressed embedding space by training the context compressor and shared-weight network to predict clean target embeddings from corrupted compressed views. Further analyses show stronger global structure and more reliable decoding, supporting our insight that compressed embeddings should be easy for the diffusion model to learn and decode into tokens. JPEG-DLM outperforms the two-stage COSMOS baseline at different compression rates and maintains strong generation quality while reaching the highest throughput among diffusion and flow models on LM1B and OWT. These results could motivate further work on jointly learning compressed embeddings for efficient diffusion language modeling.

AI Use Statement

Generative AI tools were used to draft and edit parts of the manuscript, suggest the manuscript organization, search for and summarize literature, verify citations and metadata, format references, refine figure presentation, assist with LaTeX formatting, and check the interpretation and presentation of experimental results. All AI-assisted revisions were manually reviewed, cited claims were checked against the original sources, and reported results were verified against the underlying experimental records. We take full responsibility for the final content of this paper, including all text, claims, figures, citations, and artifacts produced with the assistance of generative AI.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736. Curran Associates, Inc., 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, June 2023. doi: 10.1109/CVPR52729.2023.01499. URL https://openaccess.thecvf.com/content/CVPR2023/html/Assran_Self-Supervised_Learning_From_Images_With_a_Joint-Embedding_Predictive_Architecture_CVPR_2023_paper.html.

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, pp. 17981–17993. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1298–1312. PMLR, 2022. URL https://proceedings.mlr.press/v162/baevski22a.html.

Randall Balestriero and Yann LeCun. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025. URL https://arxiv.org/abs/2511.08544.

Fan Bao, Shen Nie, Kaiwen Xue, Chongxuan Li, Shi Pu, Yaole Wang, Gang Yue, Yue Cao, Hang Su, and Jun Zhu. One transformer fits all distributions in multi-modal diffusion at scale. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1692–1717. PMLR, 2023. URL https://proceedings.mlr.press/v202/bao23a.html.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=QaCCuDfBk2.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *Proceedings of Interspeech*, pp. 2635–2639, 2014. doi: 10.21437/Interspeech.2014-564. URL https://www.isca-archive.org/interspeech_2014/chelba14_interspeech.html.

Dengsheng Chen, Jie Hu, Xiaoming Wei, and Enhua Wu. Denoising with a joint-embedding predictive architecture. In *International Conference on Learning Representations*, pp. 22479–22517, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/389ae3090873f86af03df533884aecc2-Abstract-Conference.html.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026. URL https://arxiv.org/abs/2604.11748.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Antonio Franca and Alexander Tong. Hacking generative perplexity: Why unconditional text evaluation needs distributional metrics. *arXiv preprint arXiv:2606.08417*, 2026. URL https://arxiv.org/abs/2606.08417.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus, 2019. URL http://Skylion007.github.io/OpenWebTextCorpus.

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 16693–16715. Curran Associates, Inc., 2023. doi: 10.52202/075280-0730. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/35b5c175e139bff5f22a5361270fce87-Abstract-Conference.html.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026. URL https://arxiv.org/abs/2605.06548.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.647. URL https://aclanthology.org/2023.acl-long.647/.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026. URL https://arxiv.org/abs/2605.10938.

Hai Huang, Yann LeCun, and Randall Balestriero. LLM-JEPA: Large language models meet joint embedding predictive architectures. In *International Conference on Learning Representations*, volume 2026, pp. 105717–105737, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ac818ec2da1c976b267de32f59709c69-Abstract-Conference.html.

Rabeeh Karimi Mahabadi, Hamish Ivison, Jaesung Tae, James Henderson, Iz Beltagy, Matthew E. Peters, and Arman Cohan. TESS: Text-to-text self-conditioned simplex diffusion. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2347–2361. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.eacl-long.144. URL https://aclanthology.org/2024.eacl-long.144/.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022. doi: 10.52202/068431-1926. URL https:

//proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012/.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview technical report, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Version 0.9.2, 27 June 2022.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026. URL https://arxiv.org/abs/2602.16813.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_Back_to_Basics_Let_Denoising_Generative_Models_Denoise_CVPR_2026_paper.html.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343. Curran Associates, Inc., 2022. doi: 10.52202/068431-0313. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract.html.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, Shiqi Yang, B. Yang, Caifeng Shan, Ziwei Liu, and Chenyang Si. AURORA-LM: Autoencoding unified representation for continuous-latent diffusion language modeling. *arXiv preprint arXiv:2608.02602*, 2026. URL https://arxiv.org/abs/2608.02602.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32819–32848. PMLR, 2024. URL https://proceedings.mlr.press/v235/lou24a.html.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56998–57025. Curran Associates, Inc., 2023. doi: 10.52202/075280-2492. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b2a2bd5d5051ff6af52e1ef60aefd255-Abstract-Conference.html.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. COSMOS: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 14271–14299. Curran Associates, Inc., 2025. doi: 10.52202/085713-0479. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1506930fb75c82246e4d8648a66e4b27-Abstract-Conference.html.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space. *arXiv preprint arXiv:2605.07933*, 2026. URL https://arxiv.org/abs/2605.07933.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 50608–50646. Curran Associates, Inc., 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, October 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4816–4828. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/260c2432a0eecc28ce03c10dadc078a4-Abstract.html.

Heqiang Qi, Wei Huang, Mingyuan Bai, and Xiangming Meng. Cluster-level attention-guided parallel decoding for masked diffusion language models. *arXiv preprint arXiv:2605.29607*, 2026. URL https://arxiv.org/abs/2605.29607.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language-models.pdf.

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. URL https://www.jmlr.org/papers/v21/20-074.html.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, June 2022. doi: 10.1109/CVPR52688.2022.01042. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

Subham Sekhar 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, pp. 130136–130184. Curran Associates, Inc., 2024. doi: 10.52202/079017-4135. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/eb0b13cc515724ab8015bc978fdde0ad-Abstract-Conference.html.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T. Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619. PMLR, 2025. URL https://proceedings.mlr.press/v267/sahoo25a.html.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. TEncDM: Understanding the properties of the diffusion model in the space of language model encodings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25110–25118, 2025. doi: 10.1609/aaai.v39i23.34696. URL https://ojs.aaai.org/index.php/AAAI/article/view/34696.

Junzhe Shen, Jieru Zhao, Ziwei He, and Zhouhan Lin. CoDAR: Continuous diffusion language models are more powerful than you think. *arXiv preprint arXiv:2603.02547*, 2026. URL https://arxiv.org/abs/2603.02547.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 103131–103167. Curran Associates, Inc., 2024. doi: 10.52202/079017-3277. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/bad233b9849f019aead5e5cc60cef70f-Abstract-Conference.html.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5d8d4e6061c3ba96c240b7fa1ae3471d-Abstract-Conference.html.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. URL https://arxiv.org/abs/2508.15487.

Yulin Yuan, Hongshuo Zhao, and Xiangming Meng. Visual-redundancy-controlled parallel decoding for diffusion-based multimodal large language models. *arXiv preprint arXiv:2605.25820*, 2026. URL https://arxiv.org/abs/2605.25820.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. PLANNER: Generating diversified paragraph via latent language diffusion model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 80178–80190. Curran Associates, Inc., 2023. doi: 10.52202/075280-3513. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/fdba5e0a9b57fce03e89cc0cad0a24e9-Abstract-Conference.html.

A Additional Method and Experimental Details

A.1 Method Details

A.1.1 Context and Target Compressors

Joint-embedding prediction uses a compressed representation of the corrupted view to predict the clean target embedding. We therefore use a context compressor C_θ to encode the corrupted view and a target compressor $\bar{C}$ to provide the clean target. The context compressor maps the corrupted view to $\boldsymbol{z}_c$ and receives gradients from the JEP objective. The target compressor maps the clean feature matrix $\boldsymbol{h}$ to the stop-gradient target $\boldsymbol{z}_{\text{tgt}} = \text{sg}\big(\bar{C}(\boldsymbol{h})\big)$ and receives no gradients. The two compressors share the same architecture and omit additional positional embeddings for the tokens, but follow different update rules.

We update the target compressor as an exponential moving average of the context compressor,

$$\bar{\theta} \leftarrow m\bar{\theta} + (1-m)\theta, \tag{7}$$

where m is the EMA momentum. The stop-gradient fixes the target embedding during each optimization step, while the EMA updates the target compressor from the context compressor. This asymmetric update follows I-JEPA (Assran et al., 2023) and V-JEPA (Bardes et al., 2024). The momentum schedule for each dataset is reported in Table 5.

A.1.2 Latent Standardization

The compressed embeddings can have different means and scales across channels, while the flow matching path starts from standard Gaussian noise. Similar to COSMOS (Meshchaninov et al., 2025), we standardize the compressed embeddings before flow matching. This places the clean embeddings and Gaussian noise on comparable scales. We restore the original scale before decoding. Each compressed embedding contains N vectors in $\mathbb{R}^d$. During the first part of training, we compute per-channel statistics over all valid latent vectors in each batch. We update running estimates with the momentum reported in Table 5 and then fix their final values. Let $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the resulting mean and standard deviation. We apply the same transformation to the outputs of the context and target compressors,

$$\bar{\boldsymbol{z}} = (\boldsymbol{z} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma}, \tag{8}$$

Here $\oslash$ denotes element-wise division. Joint-embedding prediction and flow matching operate on $\bar{\boldsymbol{z}}$, and f_ϕ predicts standardized embeddings. Before decoding, we invert the transform as $\boldsymbol{z} = \bar{\boldsymbol{z}} \odot \boldsymbol{\sigma} + \boldsymbol{\mu}$ and pass the restored embedding to the decompressor during both training and sampling.

A.1.3 Embedding Regularization

Standardization aligns the mean and scale of each latent channel with Gaussian noise, but it does not control the full latent distribution or prevent channels from losing variation and becoming correlated during joint training. We therefore use Sketched Isotropic Gaussian Regularization (SIGReg) from LeJEPA (Balestriero & LeCun, 2025) and the variance and covariance terms from VICReg (Bardes et al., 2022). SIGReg encourages an isotropic Gaussian distribution. The variance term maintains variation within each latent channel, and the covariance term reduces redundancy across channels.

Let $U \in \mathbb{R}^{B\times d}$ contain the vectors produced by the context compressor and used to compute the regularizers, with row $\boldsymbol{u}_b^\top$. Here $B > 1$ is the number of vectors and d is the latent width. SIGReg projects these vectors onto M random unit directions $\boldsymbol{a}_m$, sampled uniformly from the unit sphere. For each direction, we compute the empirical characteristic function

$$\widehat{\varphi}_m(\omega) = \frac{1}{B}\sum_{b=1}^{B} \exp\big(\mathrm{i}\omega \boldsymbol{a}_m^\top \boldsymbol{u}_b\big), \qquad \mathrm{i}^2 = -1. \tag{9}$$

This function describes the distribution of the projected embeddings. We compare it with the characteristic function $\varphi_0(\omega) = \exp(-\omega^2/2)$ of a standard Gaussian. Using the Epps–Pulley statistic recommended by LeJEPA, the regularizer is

$$\mathcal{L}_{\text{sigreg}} = \frac{B}{M}\sum_{m=1}^{M}\int_{\mathbb{R}} \left|\widehat{\varphi}_m(\omega) - \varphi_0(\omega)\right|^2 w(\omega)\,\mathrm{d}\omega. \tag{10}$$

Here w is a Gaussian weight function. LeJEPA's numerical implementation uses $w(\omega) = \exp(-\omega^2/2)$ and trapezoidal quadrature. The loss compares projected distributions beyond their means and variances. Its gradients pass through the projections to the context compressor.

To compute the variance and covariance terms, we first center the vectors and compute their sample covariance,

$$\boldsymbol{\mu}_U = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{u}_b, \qquad \Sigma_U = \frac{1}{B-1}\sum_{b=1}^{B} (\boldsymbol{u}_b - \boldsymbol{\mu}_U)(\boldsymbol{u}_b - \boldsymbol{\mu}_U)^\top. \tag{11}$$

The diagonal entries give the variance of each channel, and the off-diagonal entries measure how channels vary with one another. Following VICReg, we use a hinge penalty on the standard deviations and a squared penalty on the off-diagonal covariances,

$$\begin{aligned} \mathcal{L}_{\text{var}} &= \frac{1}{d}\sum_{i=1}^{d} \max\Big(0, 1 - \sqrt{(\Sigma_U)_{ii} + \epsilon}\Big), \\ \mathcal{L}_{\text{cov}} &= \frac{1}{d}\sum_{i \neq j} (\Sigma_U)_{ij}^2. \end{aligned} \tag{12}$$

Here $\epsilon > 0$ stabilizes the square root. The variance term penalizes channel standard deviations below the unit target used by VICReg. The covariance term penalizes nonzero off-diagonal covariances. Both terms backpropagate to the context compressor.

The full training objective adds these regularizers to the main objective,

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{main}} + 0.1\mathcal{L}_{\text{sigreg}} + 0.2\mathcal{L}_{\text{var}} + 0.002\mathcal{L}_{\text{cov}}, \tag{13}$$

where $\mathcal{L}_{\text{main}}$ is the four-term objective in Equation 6. The target compressor follows the EMA update in Equation 7. We also ablate the effects of these regularizers in Section A.2.1.

A.1.4 Self-Conditioning

During flow matching training, each noisy embedding is sampled independently. During ODE sampling, the model instead predicts a clean embedding at each point along one trajectory. Self-conditioning (Chen et al., 2023) passes the clean embedding predicted at one step to the next, so each step reuses the previous estimate. This technique is widely used in diffusion language models. Following ELF (Hu et al., 2026), we also use the previously predicted clean embedding as the conditioning signal for guidance and sample the guidance scale w in the same way. Given a noisy embedding $\boldsymbol{z}_t$, we first predict a clean embedding $\widehat{\boldsymbol{z}}'$ and then condition a second prediction of the clean embedding, $\widehat{\boldsymbol{z}}$, on it. We stop gradients through both predicted embeddings and form the training target

$$\begin{aligned} \widehat{\boldsymbol{z}}' &= \mathrm{sg}\big(f_\phi(\boldsymbol{z}_t, t; \mathbf{0}, w)\big), \\ \widehat{\boldsymbol{z}} &= \mathrm{sg}\big(f_\phi(\boldsymbol{z}_t, t; \widehat{\boldsymbol{z}}', w)\big), \\ \boldsymbol{z}_{\text{SC}} &= \boldsymbol{z}_{\text{tgt}} + \left(1 - \frac{1}{w}\right)\left(\widehat{\boldsymbol{z}} - \widehat{\boldsymbol{z}}'\right). \end{aligned} \tag{14}$$

The coefficient $1 - 1/w$ scales the correction $\widehat{\boldsymbol{z}} - \widehat{\boldsymbol{z}}'$ and therefore controls the strength of self-conditioning. At $w = 1$, the target reduces to $\boldsymbol{z}_{\text{tgt}}$. The final pass uses $\boldsymbol{s} = \widehat{\boldsymbol{z}}'$ and $\boldsymbol{z}_{\text{SC}}$ as the target in $\mathcal{L}_{\text{flow}}$. We apply this target with probability $p_{\text{sc}} = 0.5$; otherwise, the pass uses $\boldsymbol{s} = \mathbf{0}$ and the clean target $\boldsymbol{z}_{\text{tgt}}$. Only the final pass updates the model.

During ODE sampling, the first step uses $\boldsymbol{s}_0 = 0$. Each subsequent step uses the clean embedding predicted at the preceding step,

$$\widehat{\boldsymbol{z}}_1^{(k)} = f_\phi(\boldsymbol{z}^{(k)}, t_k; \boldsymbol{s}_k, w), \qquad \boldsymbol{s}_{k+1} = \widehat{\boldsymbol{z}}_1^{(k)}. \tag{15}$$

At inference, each ODE step uses one model evaluation conditioned on w. We use $w = 3.0$ on LM1B and OWT-128, and $w = 2.0$ on OWT-1024.

A.1.5 x-Loss and v-Loss for Flow Matching

The prediction target determines what the flow matching model estimates, while the loss determines how errors at different times are weighted. We consider these choices separately because they affect optimization in different ways. With x-prediction, the model directly predicts the clean embedding $\boldsymbol{z}_1$. With v-prediction, it directly predicts the velocity along the probability path. JiT (Li & He, 2026) shows that x-prediction can be easier for a neural network than v-prediction in high-dimensional spaces. We therefore let f_ϕ predict $\boldsymbol{z}_1$ directly.

For the linear path $\boldsymbol{z}_t = t\boldsymbol{z}_1 + (1-t)\boldsymbol{\epsilon}$ in Equation 1, differentiating with respect to t gives the target velocity

$$\boldsymbol{u}_t = \frac{\mathrm{d}\boldsymbol{z}_t}{\mathrm{d}t} = \boldsymbol{z}_1 - \boldsymbol{\epsilon}. \tag{16}$$

Standard flow matching (Lipman et al., 2023) directly predicts this velocity and minimizes the v-loss

$$\mathcal{L}_v = \mathbb{E}_{\boldsymbol{z}_1,t,\boldsymbol{\epsilon}}\left[\|\boldsymbol{v}_\phi(\boldsymbol{z}_t,t) - \boldsymbol{u}_t\|_2^2\right]. \tag{17}$$

With x-prediction, the corresponding x-loss is computed directly on the predicted clean embedding,

$$\mathcal{L}_x = \mathbb{E}_{\boldsymbol{z}_1,t,\boldsymbol{\epsilon}}\left[\|f_\phi(\boldsymbol{z}_t,t) - \boldsymbol{z}_1\|_2^2\right], \tag{18}$$

which is the flow matching objective used in Equation 2. An x-prediction can be converted into the corresponding velocity prediction for any $t < 1$,

$$\boldsymbol{v}_\phi(\boldsymbol{z}_t,t) = \frac{f_\phi(\boldsymbol{z}_t,t) - \boldsymbol{z}_t}{1-t}. \tag{19}$$

Substituting this conversion into the v-loss gives

$$\|\boldsymbol{v}_\phi - \boldsymbol{u}_t\|_2^2 = \frac{1}{(1-t)^2}\,\|f_\phi(\boldsymbol{z}_t,t) - \boldsymbol{z}_1\|_2^2. \tag{20}$$

Equation 20 weights each x-prediction error by $(1-t)^{-2}$ when it is expressed as a velocity error. This weight increases as t approaches 1. Our logit-normal distribution over t places more probability mass near $t = 0$, and the x-loss preserves this sampling emphasis. Recent work (Liang et al., 2026) also finds that predicting the clean embedding with an x-loss outperforms predicting velocity with a v-loss. We therefore use the x-loss as the default objective for learning the latent prior with flow matching.

A.2 Additional Results

A.2.1 Regularizer Ablation

SIGReg shapes the overall latent distribution, while the variance and covariance terms maintain channel variation and reduce correlations between channels. We therefore evaluate them separately and jointly in Table 3 to measure their effects on generation quality. The shaded row is the Base configuration from Table 2.

The runs with only SIGReg and only the variance and covariance terms both have lower Gen-PPL and higher MAUVE than the run without either regularizer. With the variance and covariance terms enabled, adding SIGReg changes entropy from 3.965 to 4.064, MAUVE from 0.868 to 0.912, and Gen-PPL from 40.77 to 48.00. The Base configuration therefore has the highest MAUVE in this ablation, while the run without SIGReg has the lowest Gen-PPL.

Table 3: Regularizer ablation on OWT-128 with $r = 0.5$, $d = 512$, 64 sampling steps, and self-conditioning at scale 3.0. The variance and covariance terms are enabled or disabled as a pair. The shaded row reproduces Base from Table 2.

SIGReg	VICReg	Gen-PPL↓	MAUVE↑	Entropy
×	×	63.11	0.856	4.090
✓	×	48.60	0.865	4.062
×	✓	**40.77**	0.868	3.965
✓	✓	48.00	**0.912**	4.064

A.2.2 Generation Quality Across Sampling Steps on LM1B

To compare the methods across a wider range of sampling budgets, we evaluate 8 to 256 steps on LM1B. The sweep includes JPEG-DLM at $r = 0.25$, MDLM, Duo, FLM, LangFlow, and ELF. We use the checkpoints and scoring from the main comparison. Each point averages five seeds with 1,024 samples per seed. The ELF results use the checkpoint reported in Table 1.

Figure 6 and Table 4 show that JPEG-DLM at $r = 0.25$ has lower Gen-PPL than every uncompressed baseline at every step count. Gen-PPL decreases steadily with more steps. MAUVE rises quickly, peaks at an intermediate budget, and remains high thereafter.

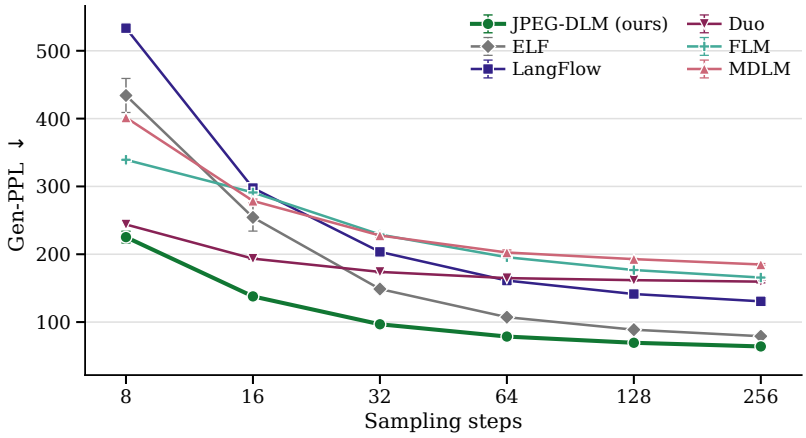


Figure 6: Gen-PPL on LM1B across sampling steps. Error bars show standard deviations over five seeds. Standard deviations are unavailable for the 8-step baselines.

Table 4: Generation quality of JPEG-DLM on LM1B with $r = 0.25$ at different sampling budgets. Results are means over five seeds. The shaded row is the 32-step configuration used in the main comparison.

Steps	Gen-PPL ↓	Ent.	MAUVE ↑
8	225.24	4.162	0.432
16	137.86	4.270	0.896
32	96.76	4.263	**0.951**
64	78.64	4.248	0.940
128	69.45	4.238	0.941
256	**64.07**	4.230	0.933

A.3 Limitations and Future Work

Limitations. First, although JPEG-DLM outperforms all uncompressed baselines in short-text generation while using latent-length compression, this advantage does not yet extend uniformly to long-text generation. With twofold latent-length compression, JPEG-DLM does not yet outperform every uncompressed baseline across all quality measures (Table 1). Nevertheless, Figure 4 shows that JPEG-DLM consistently outperforms the existing two-stage compressed baseline across compression rates. These results suggest that jointly learning compressed embeddings is a promising direction for text generation under latent-length compression.

Second, the corruption noise scale is a training hyperparameter that requires careful calibration. It controls an interpretable balance between robust prediction and information preservation. The ablation in Table 2 follows this expected pattern. An intermediate scale produces the best overall quality. Weaker corruption provides insufficient pressure to predict clean embeddings robustly, whereas excessive corruption removes information needed to recover the clean target embedding.

Future work. Our results show that learning compressed embeddings jointly with the generative model yields better generation quality than two-stage training at different compression rates. Unlike two-stage training, joint training allows the embedding space to adapt to the generative objective while it is still trained for token decoding. The goal is therefore to learn compressed embeddings with a structure that is easy for the diffusion model to learn and allows generated embeddings to be decoded reliably into tokens. Accordingly, we consider three directions for improving text generation quality under stronger latent-length compression.

Compressor architecture. The current compressor and decompressor attend over their complete inputs, so compressed positions have no fixed correspondence with token spans. AURORA-LM (Liang et al., 2026) instead uses a query-based encoder with causal attention to form a prefix-aligned latent sequence. Its results suggest that causal encoding is a promising direction for retaining positional structure as latent length decreases.

Contextual token decoding. The current token projection predicts each token independently from its decompressed feature and does not model dependencies among output tokens. CoDAR (Shen et al., 2026) finds that an autoregressive Transformer decoder maps continuous representations back to tokens more accurately than a position-wise linear head. A contextual decoder may therefore improve decoding under stronger compression.

Alternative joint training objectives. JEP is one way to shape the compressed embedding space during joint training. Another approach is to update the compressor directly with the generative loss at each sampled time step. LDLM (Meshchaninov et al., 2026) follows this strategy and uses several techniques to stabilize training. In our preliminary experiments, applying flow matching gradients to the compressor at each sampled time step was difficult to stabilize and led to representation collapse. Further work on training objectives and stabilization may yield a compressed embedding space that is easier for the diffusion model to learn. This could improve text generation quality under stronger compression.

A.4 Experimental Details

Table 5 lists the model configurations and training schedules for LM1B and OWT-1024. The OWT-128 comparisons and ablations are detailed in Appendix A.4.6.

A.4.1 Data Preparation

We follow the packed LM1B preprocessing of Duo (Sahoo et al., 2025), adapted for the frozen T5 encoder. For OWT-1024 we pack text along sentence boundaries into blocks of exactly 1024 T5 tokens, so every position holds a real token and no padding is needed. Packing makes EOS a document boundary rather than a sequence terminator, so we keep the text after the first EOS when scoring.

A.4.2 Truncated MAUVE

For MAUVE (Pillutla et al., 2021), we represent each text with the hidden state at its final retained GPT-2 token. Because this state is extracted at the end of the retained context, the resulting representation varies with text length and boundary placement as well as language quality. Consequently, differences in length distribution can confound the comparison between generated and reference texts. For OWT-1024, we construct one reference set by splitting a GPT-2 token stream into fixed blocks of 1,024 tokens. ELF produces its samples differently. It generates on a 1,024-position T5 canvas and decodes with T5 SentencePiece. After GPT-2 re-tokenization, its outputs therefore follow a different length distribution. Figure 7 also shows a second OWT reference set that was first tokenized with T5 and then decoded.

This construction difference leads to a large length mismatch, as shown in Figure 7. For the 32-step ELF run with seed 0, the generated documents have a median length of 950 GPT-2 tokens and a standard deviation of 31.9. By comparison, the fixed-length reference has a median length of 1,021 tokens and a standard deviation of 1.0. After discretizing document lengths into 204 equal-frequency bins, the intersection of the two histograms is only 0.9%. Consequently, MAUVE computed on the full texts can reflect the length mismatch in addition to differences in generated text.

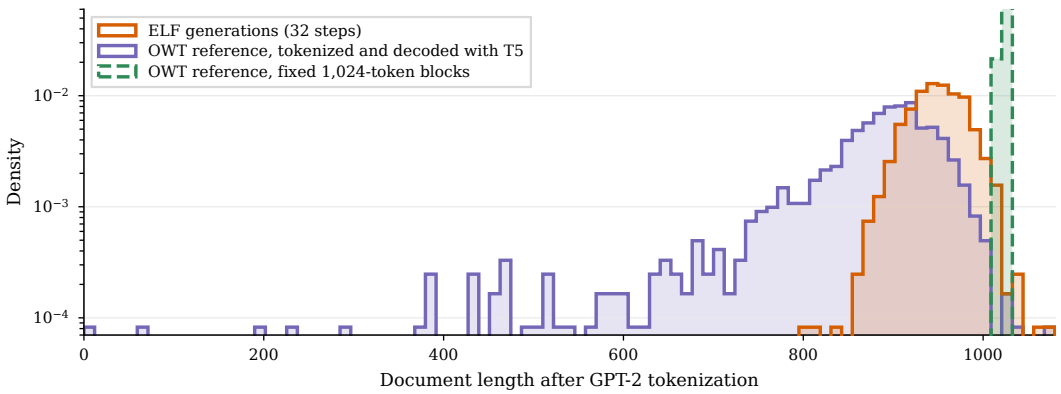


Figure 7: GPT-2 token lengths of ELF generations and two OWT reference sets. The reference constructed from fixed 1,024-token blocks concentrates near the block length, whereas the reference tokenized and decoded with T5 and the ELF samples vary in length. After discretization, the intersection between the ELF and fixed-length reference histograms is only 0.9%.

TEncDM (Shabalin et al., 2025) reports the same sensitivity on ROCStories. It truncated both generated and reference texts to 40 tokens. After truncation, MAUVE increased from 0.762 to 0.940 with BERT encodings and from 0.647 to 0.913 with RoBERTa encodings. Thus, text length can substantially change the MAUVE score. Accordingly, we report truncated MAUVE by truncating generated and reference texts to the same GPT-2 token budget before feature extraction. As a result, the final retained token appears at the same position for nearly all texts, which reduces the influence of length on the resulting representation. We use a budget of 96 tokens on LM1B and OWT-128 and 768 tokens on OWT-1024. Both budgets retain 75% of the nominal sequence length.

A.4.3 Model and Training Configuration

The latent DiT follows ELF-B (Hu et al., 2026), with separate output heads for the denoise and decode branches. It prepends four time tokens, four self-conditioning tokens, and four model-mode tokens to the latent sequence.

A.4.4 Baseline Details

Each model uses the tokenizer it was trained with, as listed in Table 6. The AR baseline uses nucleus sampling (Holtzman et al., 2020).

On OWT, we use the official checkpoints of MDLM (Sahoo et al., 2024), SEDD-Absorb (Lou et al., 2024), Duo (Sahoo et al., 2025), FLM (Lee et al., 2026), LangFlow (Chen et al., 2026), and ELF (Hu et al., 2026), and the reference AR checkpoint released with MDLM. On LM1B, we use the official FLM and LangFlow checkpoints, and the MDLM and Duo checkpoints released with FLM. AR, SEDD-Absorb, and ELF have no public LM1B weights, so we train these models ourselves following their official recipes.

A.4.5 Evaluation Details

For the main comparison, each model uses its own sampler and the step count in Table 6. For JPEG-DLM, we draw the initial embedding from a standard Gaussian and integrate the learned velocity field with a 32-step Euler solver. The decode branch then applies $f_\phi(\cdot, 1)$ once, restores the predicted clean embedding to its original scale, and maps it to T5 tokens with greedy decoding. Because the models use different vocabularies, we decode each sample to text and re-tokenize it with GPT-2 Large before scoring. We also lowercase the LM1B outputs of ELF to match the uncased reference. Gen-PPL is the exponential of the average negative log-likelihood over all scored tokens, and unigram entropy is reported in nats.

We measure median generation throughput in FP32 at batch 64 on LM1B and batch 16 on OWT-1024. The timing excludes model loading and detokenization. AR without a KV cache defines the $1.00\times$ baseline. With a KV cache, AR reaches $1.62\times$ this throughput on LM1B and $7.14\times$ on OWT-1024.

Table 5: Model and training configurations of JPEG-DLM on LM1B and OWT-1024. Sampling compute counts one multiply–accumulate as one operation. For K solver steps, JPEG-DLM uses $K+1$ DiT evaluations, including the final clean embedding prediction, followed by one decoding pass. The estimate at full latent length uses the same $K+1$ evaluation budget.

Parameter	**LM1B**	**OWT-1024**
Data and tokenization		
Dataset	LM1B (Chelba et al., 2014)	OWT
Token sequence length	128	1024
Sequence construction	Blocks packed across sentence boundaries	Blocks packed across sentence boundaries (no padding)
Model architecture		
Frozen text encoder	T5-small	T5-small
Latent length / compression rate r	32 / 0.25	512 / 0.5
Latent width	512	512
Compressor / decompressor layers	6 / 6	6 / 6
Compressor attention heads	8	8
Feature bottleneck	128	128
DiT layers / hidden size / heads	12 / 768 / 12	12 / 768 / 12
DiT MLP ratio / dropout	4 / 0	4 / 0
DiT + token projection parameters	105M	105M
Including compressor and decompressor	154M	154M
Training		
Training epochs	20	5
Number of GPUs	8	8
Random seed	42	42
Local batch / gradient accumulation	384 / 2	24 / 2
Effective global batch size	6144	384
Optimizer	AdamW (β_1=0.9, β_2=0.999)	AdamW (β_1=0.9, β_2=0.999)
Precision	bfloat16	bfloat16
Gradient clipping / weight decay	1.0 / 0.01	1.0 / 0.01
Flow matching learning rate	1×10^{-3}	1×10^{-3}
Compressor / decompressor learning rate	3.0×10^{-4}	2×10^{-4}
Feature perturbation	Gaussian, $\sigma = 0.8$	Gaussian, $\sigma = 0.8$
Flow matching / JEP / token / feature weights	1 / 1 / 1 / 1	1 / 1 / 1 / 1
SIGReg weight	0.1	0.1
Latent standardization momentum	0.99	0.99
EMA decay for target compressor	$0.995 \rightarrow 0.999$	$0.996 \rightarrow 0.999$
Model weight EMA	0.999	0.999
Time sampling	Logit-normal $(-1.5, 0.8)$	Logit-normal $(-1.5, 0.8)$
Self-conditioning probability	0.5	0.5
Sampling compute (compressed / full latent length)	0.13 / 0.40 TFLOPs	1.68 / 3.56 TFLOPs

A.4.6 Experiments on OWT-128

Training configuration. We use the linluqiu/openwebtext-len128-packing_padding dataset, whose sequences contain 128 T5 tokens. These experiments use the OWT-1024 architecture and optimization schedule in Table 5. For OWT-128, we use $\sigma = 1.0$, 64 sampling steps, and self-conditioning scale 3.0. The studies below specify the compression rate and latent width that they vary.

Latent compression and component ablations. The latent-length study fixes $d = 512$ and evaluates $r \in \{1.0, 0.5, 0.25, 0.125\}$. The latent-width study fixes $r = 0.5$ and evaluates $d \in \{256, 512, 768, 1024\}$. Figure 3 reports these results. The component ablations in Table 2 use $r = 0.5$ and $d = 512$. They evaluate joint-embedding prediction, feature reconstruction, noise scale, and cross-entropy timesteps.

Baselines. We retrain the T5 variant of COSMOS with the same frozen T5-small encoder, tokenizer, and compressor and decompressor depths as JPEG-DLM. Table 7 lists its configurations across com-

Table 6: Tokenization and sampling configurations for the main unconditional generation comparison. BERT (Devlin et al., 2019) denotes `bert-base-uncased`, and GPT-2 (Radford et al., 2019) denotes GPT-2 BPE. T5 (Raffel et al., 2020) uses SentencePiece (Kudo & Richardson, 2018). A checkmark in the SC column indicates self-conditioning.

			LM1B		OWT-1024	
Method	**SC**	**Samples**	**Tokenizer**	**Steps**	**Tokenizer**	**Steps**
AR	×	1,024	BERT	–	GPT-2	–
MDLM	×	1,024	BERT	128	GPT-2 + mask	1024
SEDD-Absorb	×	1,024	GPT-2	128	GPT-2	1024
Duo	×	1,024	BERT	128	GPT-2	1024
FLM	×	1,024	BERT	128	GPT-2	1024
LangFlow	✓	1,024	BERT	128	GPT-2	1024
ELF	✓	1,024	T5	32	T5	32
JPEG-DLM (ours)	✓	1,024	T5	32	T5	32

Table 7: Configuration of COSMOS on OWT-128.

Parameter	**OWT-128**
Dataset	linluqiu/openwebtext-len128-packing_padding
Sequence length	128
Text encoder / tokenizer	Frozen T5-small / T5
Encoder feature	Final hidden layer
Latent length / compression rate r	128 / 1.0, 64 / 0.5, 32 / 0.25, 16 / 0.125
Latent width	512
Compressor / decompressor layers	6 / 6
Attention heads	8
Vocabulary size	32,100
Diffusion schedule parameters	$N = 200$, $d = 5$
Token masking / MLM probability	0.5 / 0.3
Gaussian noise weight / scale	0.5 / 0.7
Latent masking probability	0.4

pression rates. ELF-B uses its official hyperparameters on this dataset and serves as the uncompressed baseline.

COSMOS performance at $r = 1.0$. Following its default implementation, COSMOS feeds the final hidden layer of the frozen T5-small encoder to the compressor. We denote this feature as layer -1. With this feature, COSMOS performs worse at $r = 1.0$ than at $r = 0.5$, even though $r = 1.0$ retains one latent position per token. We examine whether the encoder feature contributes to this result by training additional $r = 1.0$ models using hidden states from layers -2 and -3.

Table 8: COSMOS performance at $r = 1.0$ with T5-small layer indices -1, -2, and -3. Results are means over five seeds using 1,024 samples per seed.

Layer index	**Gen-PPL ↓**	**Ent.**	**MAUVE ↑**
-1	240.811	3.9033	0.1167
-2	154.551	3.9736	0.3009
-3	163.196	3.9717	0.2677

The poor result at $r = 1.0$ cannot be explained by limited latent capacity, because this configuration retains one latent position per token. COSMOS trains its autoencoder before the diffusion model, so the diffusion model must fit the latent space induced by the selected T5 feature. LDLM (Meshchaninov et al., 2026) reports that the encoder layer has a large effect on a COSMOS-style latent space. The final layer gives substantially worse MAUVE than earlier layers. LDLM sug-

gests that the smoothing and robustness objectives are sensitive to the structure of the pretrained representation.

Our results show the same pattern with T5-small. Changing the input from layer -1 to layer -2 lowers Gen-PPL from 240.81 to 154.55 and raises MAUVE from 0.117 to 0.301. Layer -3 reaches a Gen-PPL of 163.20 and a MAUVE score of 0.268. It is slightly worse than layer -2 but remains much better than layer -1. This comparison shows that the final T5 feature accounts for much of the unusually poor result at $r = 1.0$. However, changing the encoder layer does not close the gap. The $r = 0.5$ model with layer -1 reaches a Gen-PPL of 122.46 and a MAUVE score of 0.358. At $r = 0.5$, each latent position must aggregate information from multiple token features. This stronger bottleneck reduces token-level variation and imposes more structure on the latent representation. The remaining gap is therefore consistent with compression making the latent distribution easier for the diffusion model to learn. Figure 4(a) follows the default COSMOS implementation and reports layer -1 at every compression rate. The results from layers -2 and -3 are used only for this diagnostic.

Generation quality. For Figure 4(a), COSMOS uses 200 sampling steps, while JPEG-DLM and ELF-B use 64. We re-tokenize the decoded samples with GPT-2 Large to compute Gen-PPL and unigram entropy. MAUVE uses 1,024 OWT reference documents and the first 96 GPT-2 tokens from each generated and reference text.

Global structure. We sample 2,048 sequences from distinct OWT documents with seed 20260919 and remove duplicate token sequences. We encode them with frozen T5-small and apply each compressor at $r = 0.5$. The JPEG-DLM target compressor and the COSMOS autoencoder compressor produce clean compressed embeddings with 64 latent positions and width 512.

We transform the embeddings into the standardized coordinates used by each generative model. At each latent position, we subtract the mean across sequences. We then flatten each position across sequences and channels, normalize the resulting 64 vectors, and compute their 64×64 Gram matrix. Figure 4(b) displays the off-diagonal entries with a shared color scale.

Reliable decoding. We form 1,024 pairs of clean compressed embeddings using random seed 20260920. For each pair $(\boldsymbol{z}_a, \boldsymbol{z}_b)$, we compute

$$\boldsymbol{z}_\lambda = (1-\lambda)\boldsymbol{z}_a + \lambda \boldsymbol{z}_b, \qquad \lambda \in \{0, 0.2, 0.3, 0.4, 0.45, 0.5, 0.55, 0.6, 0.7, 0.8, 1\}. \tag{21}$$

We interpolate in the standardized coordinates used by each generative model. Without added noise, we invert each model's standardization and pass $\boldsymbol{z}_\lambda$ to its decoding module without first applying the generative model. The three noise levels use COSMOS forward times $t \in \{0.1, 0.2, 0.3\}$ with schedule parameter $d = 5$ and $\mathrm{SNR}(t) = [d \tan(\pi t/2)]^{-2}$. For JPEG-DLM, the times $t' \in \{0.5581, 0.3810, 0.2819\}$ on the linear path match these three SNR values.

We draw Gaussian noise with seed 0 and normalize each tensor to unit RMS. A shared noise tensor is applied to both methods at every λ. At each noisy state, the generative model predicts one clean embedding, which is restored to the original latent scale and mapped to text. For this diagnostic, JPEG-DLM uses self-conditioning scale 1.0, and COSMOS receives a zero self-conditioning input. We compute Gen-PPL with GPT-2 Large for the 1,024 decoded texts at each (λ, t) pair. The figure reports 95% intervals from 2,000 bootstrap resamples over texts and connects the measured values with monotone cubic interpolation.

A.5 Qualitative Generation Samples

We include two unconditional generations produced by the JPEG-DLM configuration used for the OWT-1024 results in Table 1.

Example 1 OWT-1024

But the fact that ESPN has proven to be the most successful news company in the United States is far more complicated than the nature of it. The reason why ESPN is changing, as Mr. Sargent asserted, is about what happens in the US. While Mr. Warner has not commented with ESPN's mediocry pay-and-ftar business, his comments have received widespread criticism from some executives, many who don't have a grasp of what is most important for him. It's a reflection of ESPN's business culture.

For nearly 20 years, Mr. Fox has been on the radar of prominent executives and broadcast executives, conducting face-to-face interviews with the San Angeles-based NPR network. In 2015, he made his relationship with Richie Fitina, the company's mosttime content officer, to ESPN's $1 billion-million video streaming service, Newstime.

Sargent said he is not aware of Mr. Warner's comments as a sign of personal bias, but that he has spent many time working on media shows. "I've never seen any time in my career that no person or anyone in a media company has been in the US," he said. While Sargent hasn't seen ESPN's own news media legecies yet, he says that's not a good thing. Its motivations are driven on the nature of ESPN's relatively established media brand. "My pitch-and-feek partner, John J. Sargent has brought me to the point of what he really cares to me at ESPN," Sargent adds. "I think that's not in the same way that ESPN has," he said. "I think that's my own opinion, and I really believe with that."

As a journalist, Sargent argues that ESPN's arrogive, sknical view of the nature of its media business has made him more likely to carry out its own TV shows with no much scrutiny. When he gave a deal with ESPN in September in late 2014, Sargent's comments sparked widespread criticism and outsternation about the company's leadership. Those claims are false. Authors like Foxcasts.com's Cook, who is ESPN's VP producer and chief director for FoxWorks, and Fox Warner Inc.'s David Smith of Hollywood have dismissed the criticism of Mr. Warner's comments, Sargent said. But those claims aren't true. "I understand the validity of these claims," Sargent said. "But I think it very important is that if you watch shows like ESPN and Showtime on a very large basis, people will get away with them. And if you're a journalist, I think that is correct."

A key concern is that ESPN is based on its brand, rather than content. For some people, ESPN says it has very more control with its content model - that it wants its subscribers to watch shows they show on on their own channels. However, its criticisms, unsubstantsive, have nothing to do with broadcast viewers' behavior, as the company's ability to include content on their channels would encourage viewers to sign up to their shows. And despite Mr. Warner' claims, there is also skepticism that ESPN's content strategy - like Mr. Warner's work with Fasttime - will be difficult to ignore. Sargent's argument is that ESPN will offer more content to its subscribers on its channel, as opposed to keeping them out at the top of their TV list. But the rise of social media and streaming platforms has made it more likely that ESPN will deliver more content on its channels if it can. For many people, this is what ESPN is doing now.

It's a two-year problem, and has led to a sharp increase in criticism from executives of all stripes. In an interview post, John Benzee ounced ESPN's relationship to ESPN's upcoming programs. Last month, Benzee lagrily accused Mr. Warner of making comments - denouing the company's plan to launch a new service, Newstime - "unappropriable." Benzee refused to make questions about ESPN's new service, Newstime and its new Newstime service, aiming to save the company's so-called content to be more accessible to other audiences. "Let's not forget that no one in the United States is happy about this now," said David Barey, University of Chicago's Department of Business, Media, and Politics. "It concerns me of the importance of ESPN and his people of all the world and the fact that he's still at the top of the line."

Example 2 OWT-1024

Even though Windberg's statements are correct, this week's decision will show that that's consistent with the laws we're going to make in California," California Publicity Chairman John Russler said. "Bushberg's statements were not accurate," in a statement. "Bushberg had said he had violate to Section 230 of the Bill of Rights, but the court was satisfied that [Bushberg] had held violation of his law." The ruling, which will be heard in a California court court, is to begin in month.

On Friday the U.S. Court of Appeals spelled a court order pending an order to halt Windway, an organization whose content is non-ivo content. Gov. Mike Schugan is slated to speak before the Supreme House on it on Dec. 1. In a statement on Tuesday, Schuich said the state would block sale of the news company's business as a marketing service to California-based Fastfeed Media, an a media company founded in 2005 and founded by Rida Bardo. California Gov. Scott Scott said the court's decision didn't change Windberg's claims. "When I'm elected, I'm disgusted," he said. The state has refused to block the sale of Winddoor's advertising service to Amazon.

"This decision clearly has a serious impact to the lives of people who want to share our media content. I am deeply appalened by it, and I believe that it would be contrary of the constitutional principles of the federal Amendment," Windway CEO John Gushberg said in a written statement. Although Windwater shifted its business to Amazon, it also shifted its business by promoting a paid service. In June, Windwater decided to cease its business.

In a decision issued Monday by the U.S. Department of Justice, the California Court of Appeals said Winddoor had suspended its business. The news company viewed more than 100 videos posted on YouTube, including videos like "FikeBids" on a wide range of topics, including movies, "Bake," and "Bakeball," David Smith, one of Windway's founders at Goldman Networks Inc., argued in a statement. Earlier Tuesday, Michael J. Saffer, who turned out Windberg's opinion in the ruling, said that prosecutors would now intend to issue an order on Winddoor's business, which violated of the Freedom of Information Act and the Freedom of Information Act. "Today, the government is going to ensure that all consumers and consumers will be protected by this decision."

Jared Suly, a spokesperson for the Electronic Privacyties Center, a media and advocacy group, said the decision deemed it "unful and unfair" for Google to suspend its advertising service. "It's a serious example of corporate discrimination that is in defoul of the Constitution," Saffy said. He added that the government still respects commercial practices and was trying to ensure that media companies would violate commercial practices by violating the company's business. "At the time, Windwater touted TV shows, TV shows, and other content on its platforms - such as SportsSeeds, Star Warner Films, and Google Networks - as media services," he said.

In addition, Michael Baffer, a California law representative, declined to comment on an agreement with California-based Disney Communications Inc., which is working in its media network and produces online content. Last month, NBC News reported that Windwater planned to sign a $100 million deal in Vixiiraz, a California-based news company that has sold more than 1 million videos from more than 50 countries. Last month, the news company told Reuters that it was planning to close its business. Iinaz earned more than 90 percent of its digital revenue, and now has more than 1 million TV subscriptions. Windway's business has grown more than 98 percent of its revenue since 2009, according to NBC Press reports.

"It is impossible to be an I-ir," Ivan Irsshin, an attorney of Iiniz, said in a statement that the agreement, which took effect in January, was a "tipping tacat." "We want to make the public know that we need to be fair and that our media companies are doing things that they don't like," he said. "We can't do that at all." Iisshin said the deal was a wake-ahead for Windberg and his competitors, adding, "but the fact that we are promoting an I-id is very harmful to the country."