ASYMMETRIC DIFFUSION TRANSFORMER: REPRESENTATION SUPERVISION VIA TRAINING-TIME QUERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained representations are now widely used to train diffusion transformers, and the open question is not whether to use them but what role to give them. Existing methods answer in two ways: representation alignment (REPA) keeps the representation outside the transformer and pulls an internal activation toward it with an explicit loss, while joint diffusion brings it inside but makes it a variable the model must generate alongside the latent. We introduce **Asym-DiT**, a third option: the representation enters the transformer as a training-time query, yet never conditions the generative path, so it is dropped at inference with no added cost. An asymmetric mask lets representation tokens read the latent while the latent never reads them; the representation solves its own denoising task by reading the latent, and its loss reaches the latent only through backpropagation, reshaping the latent hidden states to expose the information the representation needs. The representation's information enters the latent through this query-induced learning rather than an explicit loss: the representation does not tell the latent what to be, only what information to expose, leaving its form free. Because the objective never fixes that form, it does not compete with generation the way an alignment loss does, and it keeps helping where REPA stalls, scaling better with model size and, unlike REPA, giving large gains when fine-tuning an already trained transformer.

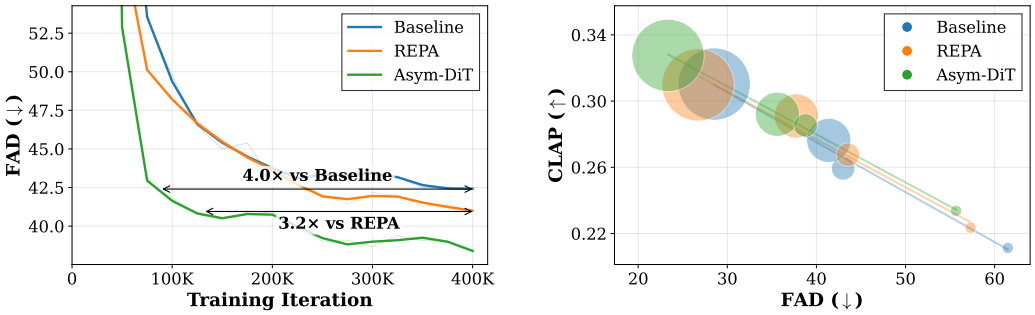


Figure 1: Asym-DiT trains faster and scales better than REPA and the vanilla baseline. (Left) Asym-DiT reaches a target FAD (MuQ, ↓) 4.0x faster than the baseline and 3.2x faster than REPA. (Right) Across model scales (marker size = model size), Asym-DiT achieves better quality (FAD MuQ) and text-audio alignment (CLAP MuQ-MuLan) than both baselines.

1 INTRODUCTION

Many recent works have successfully leveraged pretrained representations to train better diffusion transformers (DiT) (Peebles & Xie, 2023). A remaining question is what role the auxiliary representation should take. Existing methods answer this question in two ways (Table 1). The representation alignment (e.g., REPA) family (Yu et al., 2024; Zheng et al., 2026b) keeps representation supervision outside the transformer: an alignment loss matches a particular intermediate activation of a transformer to an auxiliary representation. The joint diffusion (e.g., ReDi) family (Kouzelis et al., 2026b; Wu et al., 2026) instead brings auxiliary representations inside the transformer alongside the primary variables (e.g., VAE latent) and generate both together.

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.

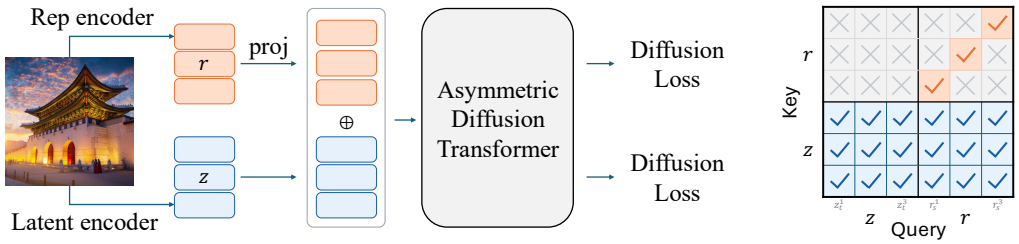


Figure 2: The proposed Asym-DiT and its asymmetric attention. (Left) Latent $\mathbf{z}$ and auxiliary representation $\mathbf{r}$ (PCA-projected) are concatenated and fed to a shared DiT, each with its own diffusion loss; $\mathbf{r}$ is dropped at inference. (Right) Asymmetric mask (✓= allowed, ✗= masked): latent queries see only latent keys, while representation queries attend to latent keys and their own token.

Both frameworks have their own drawbacks. In joint diffusion, auxiliary representations provide supervision from inside the transformer computation, at the cost of admitting more generative variables that incur sampling overhead. On the other hand, while REPA objectives do not introduce more variables to model, explicit constraints on internal statistics may sometimes conflict with denoising objectives (Zheng et al., 2026b).

We argue this dichotomy is not inevitable. The central question this paper poses is the following:

Table 1: Design space of pretrained representation integration in DiT.

Generative variable?	Supervision inside transformer? No	Supervision inside transformer? Yes
No	Representation alignment (REPA)	**Asymmetric-DiT (Ours)**
Yes	–	Joint diffusion (ReDi)

> Can an auxiliary representation provide **supervision from inside the transformer computation** without becoming a generative variable itself?

Here, what we call the auxiliary representation may come from any pretrained encoder and carry any semantics. Likewise, the primary generative variables may take any form, from VAE latents (e.g., LDM) (Rombach et al., 2022) to pretrained features (e.g., RAE) (Zheng et al., 2026a). Primary and auxiliary are thus defined by use rather than properties of an encoder, and the question is which direction of dependency to impose between the two: how an auxiliary representation can shape the primary without ever entering the generative process at inference-time.

We answer this question by Asym-DiT, in which we enable a third role for an auxiliary representation in DiT training. The primary generative variable and the auxiliary representation, which we call the *latent* $\mathbf{z}$ and the *representation* $\mathbf{r}$ throughout, both enter the DiT as tokens concatenated along the token axis as in Fig. 2, interacting through self-attention. The asymmetric attention that governs this interaction comprises two components: (i) *lat-to-rep masking*, where the latent $\mathbf{z}$ cannot attend to the representation $\mathbf{r}$ while $\mathbf{r}$ attends to $\mathbf{z}$; and (ii) *cross-rep masking*, where representation tokens cannot attend to one another. An immediate tension arises: if $\mathbf{z}$ never attends to $\mathbf{r}$ during training, how could this possibly improve generation of $\mathbf{z}$?

The key is that $\mathbf{z}$ being blind to $\mathbf{r}$ does not make the backward gradient blind to $\mathbf{r}$ as well. Rather, to solve its own auxiliary denoising task, the representation query must read the latent's keys and values and pull latent-side information, and the gradient flows back into the latent along that path, driving the $\mathbf{z}$-side intermediate activations to expose what the representation could not recover on its own. Unlike REPA, which forces a particular activation towards an external representation, our mechanism does not dictate how the latent stores the information; the alignment emerges inside the attention rather than from an explicit loss. And because no representation is present at inference, any gain in generation quality is attributable to how the representation shaped the network during training rather than to conditioning during sampling, a causal clarity joint diffusion cannot attain.

Since Asym-DiT uses an auxiliary representation to aid the training of the primary variable, it is agnostic to *when* in training it is applied. Its representation loss never pulls against the generative objective, so it can be attached to an already pre-trained DiT: after an expensive pre-training run, one can add a representation targeting a desired property and fine-tune, obtaining significant gains

within a few thousand steps. REPA, by contrast, forces a chosen latent activation toward a fixed target representation, so fine-tuning under it is slow to converge and reaches a markedly weaker final quality. The method also scales up to 1.7B, tracing a better performance curve than REPA, see Fig. 1.

2 PRELIMINARY

Latent diffusion models (Rombach et al., 2022) encode data $\mathbf{x} \sim p_{\text{data}}$ into a latent $\mathbf{z}$ and perturb it along a linear interpolant (Lipman et al., 2022), $\mathbf{z}_t = (1-t)\,\mathbf{z} + t\,\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. A diffusion transformer (Peebles & Xie, 2023) $v_{\boldsymbol{\theta}}$ is trained to predict the velocity of this path by flow matching, minimizing $\mathbb{E}[\|v_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - (\boldsymbol{\epsilon} - \mathbf{z})\|_2^2]$. At inference a sample is drawn from $\mathcal{N}(0, \mathbf{I})$ and integrated back to $t = 0$ along the predicted velocity, then decoded to data space.

3 METHODOLOGY: TRAINING-TIME QUERY IN DIFFUSION TRANSFORMERS

3.1 CONSTRUCTING REPRESENTATION TOKENS

Given $\mathbf{z}$ as the primary representation and $\mathbf{r}$ an auxiliary representation, we match channel dimensions of $\mathbf{r}$ to $\mathbf{z}$ with PCA projection (Hotelling, 1933) and rescale it by a single global scalar: $\mathbf{r} \to \mathbf{r}^{\text{pca}}$. We then perturb each at its own noise scale, $\mathbf{z}$ by t and $\mathbf{r}^{\text{pca}}$ by s, giving $\mathbf{z}_t$ and $\mathbf{r}_s^{\text{pca}}$. Corruption depends on both the token dimension and the noise level; matching the dimension removes the former, so the noise level alone sets how corrupted each token is. This makes t and s directly comparable, and we tie them through a single scalar $\alpha > 0$ (Esser et al., 2024),

$$s = \frac{\alpha t}{1 + (\alpha - 1)t},$$

so that the auxiliary tokens can be systematically noisier or cleaner than the latent at the same training step ($\alpha > 1$ gives $s > t$). This α serves as a control knob on the auxiliary supervision to the latent. We then concatenate the two token sequences along the token dimension $\mathbf{z}_t \oplus \mathbf{r}_s^{\text{pca}}$ and feed the fused tokens to a shared DiT backbone. Latent and auxiliary tokens are thus processed by the same transformer layers and interact only through self-attention.

3.2 AUXILIARY OBJECTIVE

We train the model with the combined diffusion objective

$$\mathcal{L}_{\mathbf{z}} + \lambda \mathcal{L}_{\mathbf{r}}. \tag{1}$$

Here, $\mathcal{L}_{\mathbf{z}} := \mathbb{E}[\|v_{\mathbf{z}} - (\boldsymbol{\epsilon} - \mathbf{z})\|_2^2]$ is the usual denoising objective on the latent and $\mathcal{L}_{\mathbf{r}} := \mathbb{E}[\|v_{\mathbf{r}} - (\boldsymbol{\epsilon}' - \mathbf{r}^{pca})\|_2^2]$ is the same denoising objective applied to the auxiliary representation, with λ controlling the strength of the auxiliary supervision, and $v_{\mathbf{z}}, v_{\mathbf{r}}$ are corresponding output of velocity prediction network. Both losses are computed from a single forward pass over the concatenated sequence (Algorithm 1). Adding $\mathcal{L}_{\mathbf{r}}$ does not by itself guarantee that the latent benefits: if the auxiliary query can solve its denoising task by reading neighboring auxiliary tokens, it never needs the latent, and no signal reaches it. The latent is taught only when the query is forced to read the latent instead, which is what the attention mask below arranges.

3.3 ASYMMETRIC ATTENTION MASK

The interaction between the two token types is controlled by a role-dependent asymmetric mask as in Fig. 2. Let $\mathcal{Z}$ and $\mathcal{R}$ the index sets of latent and representation tokens, and let $M_{ij} = 1$ indicate that query i attends to key j. The mask sets $M_{ij} = 0$ on exactly two groups of query-key pairs,

$$\begin{aligned} &\textbf{(lat-to-rep masking)} \quad M_{ij} = 0, \quad i \in \mathcal{Z},\ j \in \mathcal{R}, \\ &\textbf{(cross-rep masking)} \quad M_{ij} = 0, \quad i \in \mathcal{R},\ j \in \mathcal{R},\ i \neq j, \end{aligned}$$

and allow all remaining pairs. The two groups are the two components of the asymmetric attention. cross-rep masking turns out to matter as much as the lat-to-rep masking. The reason is that representation queries have a shortcut available to them. Tokens from the same pre-trained encoder are

Algorithm 1 Asym-DiT training

Require: Encoders E and E'; Network $\mathbf{v}_{\boldsymbol{\theta}}$
once: fit representation $\mathbf{r}$ PCA to latent dimension
repeat
  Sample $\mathbf{x} \sim p_{\text{data}}$ and convert $\mathbf{z} = E(\mathbf{x})$ and $\mathbf{r} = E'(\mathbf{x})$
  Sample $t \sim p(t)$ and compute $s = \frac{\alpha t}{1+(\alpha-1)t}$
  Diffuse by $\mathbf{z}_t = (1-t)\mathbf{z} + t\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
  Diffuse by $\mathbf{r}_s^{\text{pca}} = (1-s)\mathbf{r}^{\text{pca}} + s\boldsymbol{\epsilon}'$ for $\boldsymbol{\epsilon}' \sim \mathcal{N}(0, \mathbf{I})$
  Compute $v_{\mathbf{z}}, v_{\mathbf{r}} = v_{\boldsymbol{\theta}}(\mathbf{z}_t \oplus \mathbf{r}_s^{\text{pca}})$
  Compute $\mathcal{L}_{\mathbf{z}} = \|v_{\mathbf{z}} - (\boldsymbol{\epsilon} - \mathbf{z})\|_2^2$
  Compute $\mathcal{L}_{\mathbf{r}} = \|v_{\mathbf{r}} - (\boldsymbol{\epsilon}' - \mathbf{r}^{pca})\|_2^2$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} (\mathcal{L}_{\mathbf{z}} + \lambda \mathcal{L}_{\mathbf{r}})$
until converged

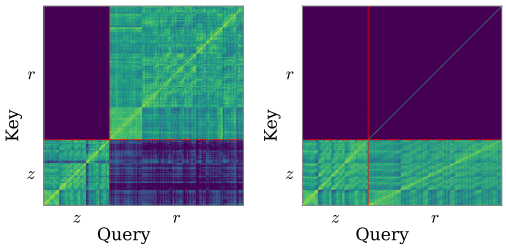


Figure 3: Attention maps (left) without and (right) with cross-rep masking, which lifts attention mass on latent tokens from 9.3% to 95.5%.

strongly correlated, so a noisy auxiliary token is often already legible from its neighbors: a linear probe recovers a token well from its nearest auxiliary neighbor ($R^2 \approx 0.65$), far better than from its nearest latent token ($R^2 \approx 0.12$).

If the auxiliary queries are allowed to attend one another, they take this shortcut: they solve their denoising task among themselves instead of reading the latent, and little signal reaches it. Masking these edges closes the shortcut and leaves the latent-side hidden states as the query's main source of the signal it needs. Fig. 3 confirms this: (b) without cross-rep masking the auxiliary queries send only 9.3% of their attention mass to latent tokens, whereas (c) with it 95.5% is directed there.

Now that the auxiliary query reads the latent, the auxiliary objective becomes supervision for it. When the auxiliary prediction is wrong, $\mathcal{L}_{\mathbf{r}}$ backpropagates through exactly the latent keys and values the query read, reshaping the latent-side hidden states to expose the information the query could not recover on its own. The latent therefore receives a learning signal from the auxiliary objective even though its own forward pass never reads the representation: the supervision travels backward through the shared attention rather than forward through conditioning, so it shapes the latent during training yet adds no cost once the auxiliary tokens are dropped at inference.

4 Analysis: What the Latent Learns to Expose

We saw in Section 3 that the auxiliary loss reaches the latent only through backpropagation: the query reads the latent-side hidden states, and when its prediction is wrong the gradient reshapes those states. But being read is not the same as being useful. The query could attend the latent and still recover nothing there that its own noisy token does not already supply, in which case the reshaping changes nothing and the method reduces to a plain DiT. This section examines whether the latent is actually used (Section 4.1), what the auxiliary objective asks it to hold (Section 4.2), and how far we can push it (Section 4.3).

4.1 Does the Representation Help?

Measuring the latent's contribution. To ask whether the query benefits from reading the latent, we compare the auxiliary loss of the normal rep output $v_{\mathbf{r}} = v_{\boldsymbol{\theta}}(\mathbf{z}_t \oplus \mathbf{r}_s^{\text{pca}})$, which attends the latent, with $v_{\mathbf{r}}^{\phi} = v_{\boldsymbol{\theta}}(\mathbf{r}_s^{\text{pca}})$, the same output when latent access is masked out. Their difference,

$$\Delta := \underbrace{\mathbb{E}\big[\|v_{\mathbf{r}}^{\phi} - (\boldsymbol{\epsilon} - \mathbf{r}^{\text{pca}})\|_2^2\big]}_{\text{Term 1}} - \underbrace{\mathbb{E}\big[\|v_{\mathbf{r}} - (\boldsymbol{\epsilon} - \mathbf{r}^{\text{pca}})\|_2^2\big]}_{\text{Term 2}}, \tag{2}$$

measures how much the latent $\mathbf{z}_t$, read through the hidden states the representation query attends, exposes about the auxiliary target $\mathbf{r}^{\text{pca}}$ beyond what the noisy auxiliary $\mathbf{r}_s^{\text{pca}}$ token already provides. It is a residual quantity, not an absolute one: it counts only the information the latent adds on top of $\mathbf{r}_s^{\text{pca}}$. A positive[1] Δ therefore says two things at once, that the latent holds such information and that the query has learned to draw on it; a Δ near zero says the latent adds nothing the auxiliary token did not already carry.

[1] If both terms were optimal, Δ would be nonnegative due to the law of total variance.

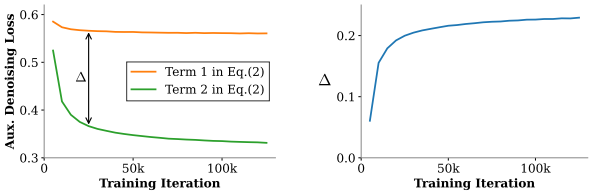


Figure 4: The latent exposes information for the auxiliary task. (Left) two terms independently measured. (Right) their gap Δ grows over training.

Table 2: A noise target, carrying no latent-relevant information, leaves FAD at the baseline; only the representation target improves it. 125k steps.

Rep. target	$\text{FAD}_{\text{MuQ}} \downarrow$	$\text{FAD}_{\text{MusicFM}} \downarrow$
None	46.50	43.87
Noise	46.50	43.64
MusicFM	**40.76**	**37.98**

The latent is used, not just read. Fig. 4 shows that Δ is positive throughout training and widens as training proceeds. The latent is therefore not merely read but used: it comes to expose information the query depends on, and exposes progressively more of it. This is the effect the asymmetric mask was designed to create, here observed directly rather than assumed.

Where the gain comes from. We have shown the query benefits from the latent, but not yet why. The benefit could come from information the latent genuinely carries, or it could be a side effect of adding a second loss, which might ease optimization regardless of its content. To separate the two, consider the converse: If the target is unrelated to the latent, the latent carries no information about it, so reading the latent cannot help the query and Δ must be zero. We build exactly this case by replacing the target with a noise, independent of the latent by construction. Under it, Δ stays at zero. The gain therefore comes from the information the latent carries; the auxiliary loss only channels it into the network, and channels nothing when there is nothing to carry.

The two move together. Table 2 shows the same in generation: FAD_{MuQ} sits at the baseline under the noise target and drops under the representation target. The exposed information and the generation quality rise and fall as one. Remove the information from the target and the gain vanishes, even though the auxiliary loss is still there.

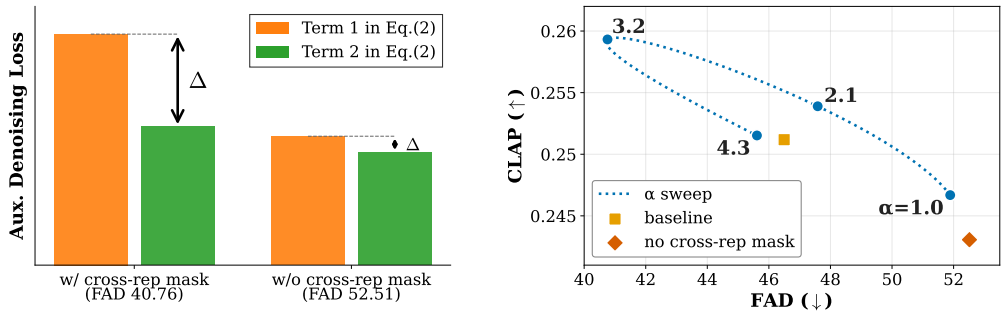
4.2 What The Latent is Asked to Hold

A problem of solving two posteriors at once. The two losses in Eq. (1) ask the shared latent hidden states to support two predictions at the same time. Minimizing $\mathcal{L}_{\mathbf{z}}$ drives the latent toward the posterior mean $\mathbb{E}[\mathbf{z}_0 \mid \mathbf{z}_t]$[2] it needs to denoise itself, while minimizing $\mathcal{L}_{\mathbf{r}}$ drives it to expose whatever the query needs to recover $\mathbb{E}[\mathbf{r}_0 \mid \mathbf{r}_s, \mathbf{z}_t]$. Because both predictions read the same latent, the network must find hidden states that serve them jointly: the latent has to carry the information for its own denoising *and* the information the representation query cannot recover from $\mathbf{r}_s$ alone.

The latent is constrained in information, not in form. Crucially, the second demand fixes *what* the latent must carry, not *what it must look like*. The query only needs to recover $\mathbf{r}_0$; any hidden state from which $\mathbf{r}_0$ is recoverable will do. Many such hidden states exist, so the auxiliary loss does not pin the latent to a single target but leaves it a family of equivalent choices. In the limit where the latent exposes everything the representation offers, it becomes a sufficient statistic for the representation: it holds all the information, while the form in which the information is held stays free.

This is where we depart from alignment. REPA takes the opposite stance. It forces a chosen activation $\mathbf{h}$ to match the representation itself, $\text{MLP}(\mathbf{h}) \approx \mathbf{r}_0$, so the latent must not just carry the representation but look like it. Our objective asks less: the query only has to recover $\mathbf{r}_0$, so the latent must carry the information but may store it in any form. That slack matters. The latent still has its own job, denoising, and it can now keep the representation's information in whatever shape also serves that job, instead of being pulled toward a shape that may fight it. Whether this freedom makes the two objectives cooperate, and improves generation, is what we test in Section 6.

[2]Velocity prediction and $\mathbf{x}$-prediction (Karras et al., 2022) are equivalent (Lai et al., 2025), so we express the optimum in the cleaner $\mathbf{x}$-prediction form, the posterior mean $\mathbb{E}[\mathbf{z}_0 \mid \mathbf{z}_t]$.



(a) Without the cross-rep mask, representation tokens attend each other freely, and the query leans on its neighbors instead of the latent, shrinking the latent-mediated gain.

(b) Changing α changes the relative corruption and therefore the supervision regime seen by the auxiliary query.

Figure 5: Two controls on query supervision. cross-rep masking removes an auxiliary-side shortcut, while the decoupled noise controls the relative difficulty of the auxiliary and latent denoising tasks.

4.3 Two Knobs on How Much the Representation Is Used

So far we have established that the latent comes to expose information for the representation, and asked what form that exposure takes. What we have not asked is whether we can control its degree: given that the latent exposes such information at all, how much more can we push it to expose? This is a question of design rather than analysis, and our method offers two knobs for it. The cross-rep mask decides whether the query has any alternative to the latent, and the decoupled noise scale α sets how hard the auxiliary task is relative to denoising the latent. Fig. 5 shows the effect of each.

Cross-rep masking widens the gap the latent can fill. We saw in Section 3.3 that cross-rep masking forces the query onto the latent by removing its shortcut to neighboring auxiliary tokens. Its effect on how much the latent is used follows directly. Fig. 5a makes the difference concrete. With the mask, giving the query access to the latent drops its loss sharply: the query has no other source, so it leans hard on the latent. Without the mask, the same access barely changes the loss: the query has already solved its task from neighboring tokens, so the latent adds little. The mask therefore does more than move attention onto the latent (Fig. 3); it makes reading the latent actually pay off.

The noise scale α sets how hard the query must lean. The scale α controls how much noise (s) we add to the auxiliary token $\mathbf{r}_s^{\text{pca}}$. Add a little, and the token $\mathbf{r}_s^{\text{pca}}$ is still clean enough to recover on its own, so the query has no reason to read the latent $\mathbf{z}_t$. Add too much, and the token is so corrupted that even the latent $\mathbf{z}_t$ cannot help recover it. Neither extreme gives the latent much to do. The useful range is in the middle, where the token is degraded enough that the query needs the latent but not so much that the task becomes hopeless. Fig. 5b confirms this: generation is best at an intermediate α, not the smallest or the largest.

5 Related Work

Loss-based representation alignment. REPA (Yu et al., 2024) attaches a small head to an intermediate DiT activation and regresses it toward a pretrained representation of the clean sample. The alignment target is a statistic of the same posterior the denoiser already estimates, but REPA forces a specific layer to realize that statistic rather than whatever the network would compute on its way to its own target. This coupling is why the alignment gradient comes to disagree with the denoising gradient over training (Zheng et al., 2026b), and why extending the idea to several representations is awkward: each must be assigned a depth or share one bottleneck, and follow-ups largely vary how much, where, or how strictly the forcing is applied (Singh et al., 2025; Wang et al., 2026; Pang et al., 2026; Shin et al., 2026; Chen et al., 2025) without removing the external loss itself.

Table 4: Model performance trained from scratch about 400k steps on the internal test set.

	FAD ↓				CLAP ↑	
Config.	MuQ	MusicFM	L-CLAP-M	MuQ-MuLan	L-CLAP-M	MuQ-MuLan
DiT-B	42.40	41.09	0.128	0.136	0.362	0.271
+ REPA	40.95	38.96	0.124	**0.126**	0.361	0.275
+ ReDi	58.94	58.35	0.152	0.253	0.339	0.191
+ REG	41.68	38.07	0.125	0.138	**0.363**	0.274
+ **Asym-DiT (Ours)**	**38.41**	**35.81**	**0.121**	**0.126**	0.362	**0.277**

Joint latent-representation generation. A separate line tokenizes the representation rather than forcing it onto an activation. ReDi (Kouzelis et al., 2026b) concatenates the latent and representation tokens and trains a single flow-matching objective under unrestricted attention, so each is generated alongside the other; CoReDi (Kouzelis et al., 2026a) and REG (Wu et al., 2026) are the multi- and single-token variants. Because the representation becomes something the model must generate, it cannot be dropped at inference without retraining, and the latent is exposed to an uninformative representation at noisy steps. Our method sits between the two: the representation participates directly in transformer computation, as in the joint line, yet stays purely training-time guidance and is dropped at inference, as in REPA.

6 Experiments

6.1 Implementation Details

Table 3: Training Configurations

Size	Depth	Hidden dim	Heads	Params	Batch
DiT-S	12	384	6	45M	128
DiT-B	12	768	12	179M	128
DiT-L	24	1024	16	633M	128
DiT-XXL	16	2048	32	1.7B	2048

We evaluate Asym-DiT on text-to-music generation using standard diffusion transformers (Peebles & Xie, 2023) at multiple model scales, with the configurations summarized in Table 3. We train on an internal licensed music corpus containing more than 20k hours of audio. Text conditioning is provided by Gemma 3 (Team et al., 2025) through cross-attention layers, while audio is represented in the latent space of a newly trained GenAE (Casebeer et al., 2026), which encodes each 32-second stereo clip into 394 latent tokens with 128 channels. For the auxiliary representation, we use the activation from the 7-th layer of MusicFM (Won et al., 2024) unless otherwise specified. The representation is computed from the same audio segment used to construct the latent. At inference time, the Euler sampler is used with 20 steps with the classifier-free guidance (Ho & Salimans, 2021) of a scale of 5.0 for all experiments.

We evaluate generation quality using Fréchet Audio Distance (FAD) (Kilgour et al., 2018; Heusel et al., 2017) and measure text-music adherence using CLAP (Wu* et al., 2023; Radford et al., 2021). We report FAD using four music representation backbones: MuQ-MuLan and MuQ (Zhu et al., 2025), MusicFM (Won et al., 2024), and LAION-CLAP-Music (Wu* et al., 2023). For text-music adherence, we report CLAP cosine similarity using LAION-CLAP-Music (L-CLAP-M) and MuQ-MuLan. All evaluations are performed on a held-out set of 5,000 text prompts, against a reference set of high-quality real music.

6.2 Quantitative Results

We evaluate Asym-DiT in three settings: training from scratch, adapting an existing diffusion transformer through fine-tuning, and scaling to larger model sizes. This separation is important because auxiliary representations can be used not only to improve representation-aware pretraining, but also as a supervision signal for adapting an already trained generative model.

Training from scratch. Table 4 compares Asym-DiT with vanilla DiT, REPA (Yu et al., 2024), ReDi (SP) (Kouzelis et al., 2026b), and REG (Wu et al., 2026) after 400k steps. Asym-DiT improves over all baselines across the FAD backbones: on MuQ, Asym-DiT improves FAD over the vanilla

Table 5: Fine-tuning performance on DiT-B and DiT-XXL.

	FAD ↓				CLAP ↑	
Config.	MuQ	MusicFM	L-CLAP-M	MuQ-MuLan	L-CLAP-M	MuQ-MuLan
DiT-B	43.63	42.39	0.130	0.143	0.361	0.266
+ REPA	43.24	40.59	0.129	0.130	**0.363**	0.275
+ Asym-DiT (Ours)	**35.50**	**36.33**	**0.124**	**0.111**	0.360	**0.293**
DiT-XXL	28.53	30.11	0.115	0.078	0.363	0.310
+ REPA	28.17	28.93	**0.114**	0.075	**0.364**	0.315
+ Asym-DiT (Ours)	**23.34**	**27.11**	0.115	**0.069**	0.363	**0.328**

baseline by 3.99 (42.40 to 38.41), against REPA's 1.45 (42.40 to 40.95); the gain is thus about 2.75× REPA's. The comparison with ReDi is the most telling, since ReDi and Asym-DiT share the identical codebase and representation construction and differ only in whether the auxiliary tokens interact freely or through our asymmetric mask. ReDi degrades quality sharply relative to the vanilla baseline (MuQ FAD 58.94), whereas Asym-DiT improves it. Simply admitting the auxiliary representation into the transformer is therefore not enough; the direction of information flow is what makes the difference.

Fine-tuning. We next attach the same supervision to an already trained transformer, starting Asym-DiT from the exact weights of a vanilla checkpoint so the experiment adds no new initialization. Table 5 shows that a relatively small number of fine-tuning steps is sufficient to improve the generation metrics. For DiT-B, 10k steps cut MuQ FAD from 43.63 to 35.50 and raise MuQ-MuLan CLAP from 0.266 to 0.293. REPA, applied the same way, improves little and stops improving with further steps, indicating that its alignment objective offers scant headroom once the model is already trained. The gap widens at scale: from the 1.7B DiT-XXL checkpoint, Asym-DiT cuts MuQ FAD from 28.53 to 23.34 and raise MuQ-MuLan CLAP from 0.310 to 0.328, while REPA again moves little. These results demonstrate that the auxiliary representation can serve as a post-hoc training signal: because our supervision does not fix the form of the latent, it does not fight the generation objective the model has already settled into, and can keep improving it.

Scaling. Fig. 1 studies how the benefit changes with model size, from DiT-S to the 1.7B DiT-XXL. We report each method in its best configuration. For REPA this is training from scratch: fine-tuning gives it little headroom (Table 5), and from-scratch is its intended use. For Asym-DiT it is a shallow fine-tuning stage on the vanilla checkpoint, where it is strongest. That checkpoint is the vanilla baseline shown in the figure, so the gap between the vanilla and Asym-DiT is exactly what our method adds on top of it. This gap is large at every size and does not close as the model grows, with the FAD improvement over vanilla running from 2.1 at DiT-S to 5.2 at DiT-XXL. REPA, by contrast, is far less consistent: at DiT-B it leaves FAD slightly worse than vanilla, and even at DiT-XXL it improves a little, far less than Asym-DiT does at the same size.

Training efficiency. Fig. 1 also shows that the same supervision reaches a given quality in fewer steps. The curves differ in shape from the start: REPA tracks the vanilla baseline closely through the early iterations and only pulls ahead later, whereas Asym-DiT improves rapidly from early in training. To match the vanilla baseline's target FAD, Asym-DiT needs $4.0\times$ fewer steps than the baseline and $3.2\times$ fewer than REPA. These step savings come at little per-step cost. Once the auxiliary encoder features are cached offline, the only remaining overhead is the longer sequence: per-step time is nearly identical to the baseline at DiT-B and adds only about 20% up to DiT-L. The step savings therefore translate almost directly into faster training.

6.3 Ablations

How much of $\mathbf{z}_t$ should $\mathbf{r}_s$ read? Cross-rep masking already restricts what a representation token may attend, cutting it off from other representation tokens. It is natural to ask whether restricting it further, on the latent side this time, would help: rather than reading the whole latent sequence, each representation token would attend only the latent at its own position and a local neighborhood, a cleaner and more targeted view. The answer, in Table 6, is no. It leaves most metrics nearly unchanged

Table 6: Full vs. local latent view for the representation after 125k steps.

Rep's view of latent	FAD_{MuQ}	$FAD_{MusicFM}$
Full seq (Asym-DiT)	**40.76**	**37.98**
Local window	45.65	40.98

Table 7: Diffusing at the latent dimension vs. the native dimension.

Diffused at	FAD_{MuQ}	$FAD_{MusicFM}$
Lat dim (Asym-DiT)	**40.76**	**37.98**
Native rep dim	47.12	43.74

Table 8: Ablation on the auxiliary target: which encoder layers are read and how they are aggregated across layers.

		FAD ↓		CLAP ↑
Method	Layer Configuration	MuQ	MusicFM	MuQ-MuLan
Baseline	–	43.63	42.39	0.266
REPA	MusicFM (L7)	42.32	40.61	0.276
Asym-DiT	MusicFM (L7)	34.62	35.17	**0.293**
	Concat (L1, L4, L7, L10)	38.74	36.67	0.292
	Sum (L1, L4, L7, L10)	37.71	36.51	0.292
	Sum (L9–L12)	**33.96**	**35.09**	0.291
	Sum (L3–L12)	37.97	36.97	0.292

but worsens FAD_{MuQ}, which is the signal we trust most here: the auxiliary target is MusicFM itself, so a MusicFM-based metric is entangled with the supervision, whereas MuQ is independent of it. Selective reading thus does not help, and on the independent metric it hurts. We keep the unrestricted view, which is also the more general choice: a representation need not align with the latent position by position, and may carry only global information with no corresponding location, where a locality constraint would be ill-posed.

Where should the representation be diffused? Section 3.1 matches the representation to the latent's channel dimension with a PCA projection and diffuses it there. An alternative is to leave the representation in its native, higher-dimensional space and diffuse it there instead, mapping it into the transformer with a dedicated learnable projection. This aligns not just the dimension but the manifold, which might seem to give the representation a more faithful target. Table 7 shows it does not: diffusing at the native dimension is clearly worse (FAD_{MuQ} 47.12 vs 40.76). Matching only the dimension and reusing the latent's own input projection, as we do, leaves the manifold to be bridged inside the attention, which proves the better choice while adding no new parameters. Keeping the representation on the shared projection also means nothing new must be trained, which is well suited to fine-tuning an existing model.

Which representation to read. The auxiliary target need not be a single encoder layer. Beyond the layer-7 (L7) activation we use by default, we try combining several layers, by concatenation and by summation over various ranges in Table 8. Summing the later layers (L9–L12) (Singh et al., 2026) matches the single-layer target (L7), and what governs quality is where those layers sit rather than how many: pulling in early layers or widening the range degrades it, so the sum must stay late. Every variant nonetheless improves clearly over the baseline and REPA, so the benefit does not hinge on exactly how the target is formed from the encoder.

7 Conclusion

We introduce Asym-DiT, which gives pretrained representations a third role in diffusion transformers: training-time queries that participate in the transformer while never becoming generative variables. Through asymmetric self-attention, representation tokens query latent-side hidden states, allowing their denoising loss to shape the latent through backpropagation without conditioning the generative forward path. Across experiments, Asym-DiT improves key generation metrics over vanilla DiT and REPA, remains effective for fine-tuning pretrained diffusion transformers with substantial gains at only a few thousand steps, and maintains a favorable scaling trend up to 1.7B parameters.

AI use statement

In this work, we used generative AI tools to assist in developing the theoretical analysis presented in Appendix D, including formalizing the mathematical claims (e.g., the Bayes-optimal denoising floors and the sufficient-statistic characterization) and drafting the accompanying derivations. All AI-assisted theoretical content was reviewed, verified, and corrected by the authors, who take full responsibility for its correctness. We did not use generative AI tools to propose or refine the core method, design the methodology or experiments, or interpret the experimental results; these were

carried out entirely by the authors. Generating synthetic datasets, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, and translation are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: software code for the proposed method was written and edited iteratively by the authors and AI tools in alternation, and generative AI tools were used to draft portions of the paper text and to improve its readability. All figures were produced manually by the authors. All AI-assisted code was verified and tested for correctness by the authors, and all AI-assisted writing was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility statement

We cannot release our code, model checkpoints, or training data as our models are trained on licensed data. To keep our results as reproducible as possible, however, we specify our method as an explicit algorithm (Algorithm 1) built from off-the-shelf and/or publicly documented components (a pretrained VAE, a standard diffusion transformer (DiT)), and give full treatise of any implementation details specific to our setup, in Section 6, Appendix B, and Appendix C. Because our baselines (REPA, ReDi, REG) were designed for image latents, we reimplemented the baselines ourselves for our DiT; Appendix C documents the procedure in full so others adapting the baseline methods can verify or reproduce our porting choices. We also fully specify our metrics in Appendix A.2, so the evaluation protocol is reproducible on any comparable model or dataset, independent of our specific trained model.

References

Jonah Casebeer, Ge Zhu, Zhepei Wang, and Nicholas J Bryan. A generative-first neural audio autoencoder. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2026.

Hesen Chen, Junyan Wang, Zhiyu Tan, and Hao Li. Sara: Structural and adversarial representation alignment for training-efficient diffusion models. *arXiv preprint arXiv:2503.08253*, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Zach Evans, Julian D Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Stable audio open. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025.

Junmin Gong, Yulin Song, Wenxiao Zhao, Sen Wang, Shengyuan Xu, Jing Guo, and Xuerui Yang. Ace-step 1.5: Pushing the boundaries of open-source music generation. *arXiv preprint arXiv:2602.00744*, 2026.

Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. Adapting frechet audio distance for generative music evaluation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

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

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 1933.

Yuepeng Jiang, Huakang Chen, Ziqian Ning, Jixun Yao, Zerui Han, Di Wu, Meng Meng, Jian Luan, Zhonghua Fu, and Lei Xie. Diffrhythm 2: Efficient and high fidelity song generation via block flow matching. *arXiv preprint arXiv:2510.22950*, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 26565–26577, 2022.

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

Theodoros Kouzelis, Spyros Gidaris, and Nikos Komodakis. Coevolving representations in joint image-feature diffusion. *arXiv preprint arXiv:2604.17492*, 2026a.

Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyridon Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026b.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in neural information processing systems (NeurIPS)*, 2023.

Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025.

Yizhi Li, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, et al. Mert: Acoustic music understanding model with large-scale self-supervised training. In *International Conference on Learning Representations (ICLR)*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2022.

Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2010. ISBN 978-0-13-198842-2.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Lianyu Pang, Tianlin Pan, Cheng Da, Changqian Yu, Huan Yang, Kun Gai, Song Guo, and Wenhan Luo. Maskalign: Token-subset representation alignment for efficient diffusion training. *arXiv preprint arXiv:2606.08788*, 2026.

Julian D Parker, Zach Evans, CJ Carr, Zachary Zukowski, Josiah Taylor, Matthew Rice, and Jordi Pons. Same: A semantically-aligned music autoencoder. *arXiv preprint arXiv:2605.18613*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*. PmLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF conference on computer vision and pattern recognition*, 2022.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think. *arXiv preprint arXiv:2603.14366*, 2026.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In *International Conference on Learning Representations (ICLR)*, 2025.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

Ahmet Solak, Florian Grötschla, Luca A Lanzendörfer, and Roger Wattenhofer. Bias beyond borders: Global inequalities in ai-generated music. *arXiv preprint arXiv:2510.01963*, 2025.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Minz Won, Yun-Ning Hung, and Duc Le. A foundation model for music informatics. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Ming-Ming Cheng, et al. Representation entanglement for generation: Training diffusion transformers is much easier than you think. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP()*. IEEE, 2023.

Chenyu Yang, Shuai Wang, Hangting Chen, Wei Tan, Jianwei Yu, and Haizhou Li. Songbloom: Coherent song generation via interleaved autoregressive sketching and diffusion refinement. *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations (ICLR)*, volume 2026, 2026a.

Xuzhe Zheng, Yuexiao Ma, Jing Xu, Xiawu Zheng, Rongrong Ji, and Fei Chao. Haste: Training-free video diffusion acceleration via head-wise adaptive sparse attention. *arXiv preprint arXiv:2605.14513*, 2026b.

Haina Zhu, Yizhi Zhou, Hangting Chen, Jianwei Yu, Ziyang Ma, Rongzhi Gu, Yi Luo, Wei Tan, and Xie Chen. Muq: Self-supervised music representation learning with mel residual vector quantization. *Transactions on Audio, Speech and Language Processing*, 2025.

A Supplementary: Training and Evaluation Details

A.1 Training Data and Data Pipeline

This appendix describes the composition of the training corpus and the on-the-fly data pipeline used to produce training batches, including dataset filtering, segment sampling, augmentation, and text-conditioning construction.

The model is trained on a large-scale internal corpus of music. Each recording is stored with associated metadata, including genre, tempo, key, etc. Prior to training, the corpus is filtered to remove low-quality, degenerate, or silent segments or samples. After filtering, the training set totals approximately 24k hours of licensed audio. Audio is stored/decoded at 44.1 kHz, stereo.

Time-stretching and pitch-shifting are applied as data augmentation, each with a probability of 0.25 per sample. for each, a random speech/tempo factor is drawn uniformly from the range [0.9, 1.1] times and 2, 1, +1, +2 semitones, respectively. Any tempo- or key- metadata associated with the example is adjusted consistently with the sampled augmentation so that conditioning signals remain aligned with the (possibly stretched/shifted) audio actually presented to the model.

At training time, captions are used as-is with probability 0.85, and replaced with an empty string otherwise.

Each 32-second segment corresponds to a fixed-length sequence of 394 latent frames under the model's audio latent encoder, a variant of GenAE (Casebeer et al., 2026).

A.2 Evaluation Details

This appendix describes the evaluation pipeline used to track generation quality throughout training. Two metrics are reported: Fréchet Audio Distance (FAD) and audio-text cosine similarity.

The reference set for FAD consists of several thousand real, licensed recordings, matched in domain to the training distribution. We use five models trained on music to evaluate FAD: LAION-CLAP-Music (Wu* et al., 2023), MusicFM (Won et al., 2024), MuQ-MuLan, and MuQ (Zhu et al., 2025). For LAION-CLAP-Music (`music_audioset_epoch_15_esc_90.14`), we follow the most widely used conventions from FADtk (Gui et al., 2024), to take one statistic per 10-second window, taken at a hop size of 1 seconds. As our generated samples are 32 seconds long, this amounts to having 23 statistics per sample. For MuQ-Mulan, we take three nonoverlapping 10s windows, as this was the segment size used in training (Zhu et al., 2025). For the others, we use the whole segment, resulting in one statistic per patch. For MuQ-MuLan, we use final-layer outputs, following (Solak et al., 2025). For MuQ and MusicFM, we follow vendor-exposed defaults to use layer-12 and layer-7 outputs, respectively.

Alongside FAD, we report the cosine similarities between a generated clip's audio embedding and the embedding of its conditioning text prompt, computed via the audio-text contrastive backbones (LAION-CLAP-Music and MuQ-MuLan). As per field conventions we abbreviate "cosine-similarity" to "CLAP". This metric serves as a text-adherence proxy, complementing FAD which measures overall distributional realism.

The formulation slightly changes for Table 9, as some of the systems under evaluation cannot generate beyond 30 seconds. Thus, in constructing the benchmark we crop the available renditions to 30-second segments instead of 32.

B Proposed Model Implementation Specifics

See Table 3 for the models and batch sizes used. Besides what is mentioned in the text, there is two pieces of implementation machinery that needs to be touched upon.

One, all representations used for Asym-DiT are "mono", meaning that the waveform input to MusicFM is the mean of the left and right channels of the stereo music.

Two, the frozen encoder's per-frame representation is far higher-dimensional than the audio latent's channel width, and the two streams must share a common channel dimensionality to be concatenated

into one joint sequence. In our Asym-DiT runs, we reduce the frozen representation's dimensionality with a fixed, linear principal component projection, following the precedent set by ReDi (Kouzelis et al., 2026b). The frozen encoder's centered per-frame features are projected onto the leading 128 eigenvectors of the fitted covariance, so the two streams are directly compatible for concatenation.

Two further observations motivate this choice beyond precedent alone. First, it is well known that, under a linear encoder-bottleneck-decoder setting, the PCA basis is one best possible choice for the encoder in the least-squares sense. Second, compressing the frozen encoder's high-dimensional representation down to the generative latent's own, already-fixed channel width — rather than, conversely, expanding the generative latent's channel width up to meet the encoder's native dimensionality, or introducing a learned nonlinear adapter between the two spaces — is arguably the more elegant of the available resolutions to this dimensionality mismatch. For instance, it has recently been observed that some of such approaches require an additional auxiliary loss with its own hyperparameter (Kouzelis et al., 2026a).

The PCA basis is fit once, offline, from a large cached sample of the frozen encoder's per-frame hidden states drawn from the training corpus. The full mean and covariance of the untruncated feature dimensionality are computed and eigendecomposed; the resulting mean vector, eigenvector basis, and eigenvalue spectrum are stored as fixed artifacts and loaded into the model as frozen (non-trainable) buffers.

Rather than standard per-channel whitening — dividing each retained component by its own standard deviation, which equalizes variance across all retained components and erases the relative ordering of importance — we instead applied a single scalar rescaling to the entire reduced vector, chosen so that the total variance summed across all k retained components equals a target value. This choice, again, is motivated by the precedent of ReDi (Kouzelis et al., 2026b). This preserves the relative scale ordering among the retained principal directions (leading components remain proportionally larger than trailing ones) while normalizing the overall numerical scale of the auxiliary stream to be commensurate with the audio-latent stream it is concatenated with.

In contrast, in the baseline model we follow common procedures to pre-cache mean and variance statistics for every channel to perform channel-wise whitening. A scalar value is multiplied after the whitening to constrain the largest channel variance to approximately 1.

C Baseline Model Training Specifics

C.1 Slight Nontrivialities in Adapting REPA to Audio, and a Remedy

REPresentation Alignment (REPA) (Yu et al., 2024) accelerates diffusion-transformer pretraining by adding an auxiliary alignment loss: an intermediate hidden state of the DiT, taken after a fixed number of transformer blocks and passed through a small projector MLP, is trained to match the representation a frozen, pretrained self-supervised encoder produces for the same (clean) input. Concretely, both the projected hidden state and the frozen encoder's representation are L2-normalized per token/patch and compared via negative cosine similarity, averaged over all tokens and the batch; this auxiliary loss is added to the standard denoising loss with a fixed scalar weight. At inference time, the auxiliary head is discarded and generation proceeds exactly as an unmodified diffusion transformer.

In the original image generation setting, image self-supervised models are typically made to support different image sizes. Thus, 256-by-256 Imagenet images can be resized to 224-by-224 and input to a self-supervised foundation model like DINOv2 (Oquab et al., 2023) to yield a 16-by-16 feature grid that is directly compatible with the DiT. Both REPA (Yu et al., 2024) and ReDi (Kouzelis et al., 2026b) use this tactic to match spatial resolutions.

However, this one-to-one correspondence is nontrivial to attain in *any* subdomain of audio, including music. A one-sentence summary of the reason would be that "for audio, existing deep learning model output shapes are fixed to an *absolute physical quantity* (i.e. time), which are rarely aligned". This is very unlike images, where a camera's distance to an object may change its perceived size and shape.

Essentially all audio signals are a long sequence of either integer or floating-point values ("samples"), and their interpretation is determined by their "sampling rate". These sampling rates tell any playback system (like speakers or earphones) how fast a long sequence of values should be "played" on a

system. A 48kHz audio, by definition, plays 48000 values per channel[3] per second. Conversely, if a wrong rate is used, it will sound very distorted and unnatural to human perception.

Now, virtually almost all ground-truth audio signals are usually only available at standardized sample rates such as 8kHz, 16kHz, 24kHz, 32kHz, 44.1kHz, 48kHz, 96kHz, and so on. And, most well-accepted deep learning models are trained oto handle one of these, very rarely two or three, but to our knowledge never a continuous range.

Putting the picture together, it becomes clear why the self-supervised model is bound to the duration of the audio, contrary to images that can be freely resized. For example, MusicFM (Won et al., 2024) is trained at 24kHz. On average, it outputs one feature every 960 samples. That makes 25 features per second, or, simply, 25Hz. If one instead inputs a sequence of samples meant to be played at 48kHz, the output will certainly be at a rate of 50 features per second, but the input will be very out-of-distribution, so the outputs would not be as meaningful. Conversely, the 48kHz sequence can be "downsampled" to 24kHz via well-known algorithms that are more friendly to human perception, but then the length of the sequence halves as well, since the sequence before and after the downsampling must "occupy" the same amount of time. Thus, after "downsampling", we are once again back to having a 25Hz stream of features. As such, output feature rates are effectively hardcoded for each model with respect to time, which is an absolute physical quantity. Therefore there is not an easy way to match the two.

If there were a continuously-variable-rate-capable music foundation model, it would enable an arbitrary elongation or shorten an input to match any frame rate. However, to our knowledge, no such model exists for music, and for various reasons we conjecture this to be extremely nontrivial to train, with very diminishing returns, because, once again, almost all audio signals are usually only available at standardized sample rates.

To add insult to injury, there is no widely-agreed-upon nor an immediately intuitive "rate" of features, either. While decimation is still considered essential for generative modeling, different audio models typically opt for different frame rates. On one hand, most well-known self-supervised models in music adopt 25Hz (Zhu et al., 2025; Won et al., 2024) or 75Hz (Li et al., 2024). On the other hand, for VAEs or codecs, 86Hz (Kumar et al., 2023), 43Hz (Parker et al., 2026), 25Hz (Gong et al., 2026; Yang et al., 2026)[4], 21.5Hz (Evans et al., 2025), 5Hz (Gong et al., 2026), a 13.125Hz and 36.75Hz (Casebeer et al., 2026) coexist, with these hardly being the exhaustive list. As hinted by this wide variety, works in music generation also commonly train their own VAE (Jiang et al., 2025; Gong et al., 2026; Evans et al., 2025) under vastly differing rates. This is at least in part driven due to a consensus that having lower rates seem to help some downstream tasks (Gong et al., 2026; Casebeer et al., 2026).

Thus, considering how a rate-match between the DiT and the self-supervised encoder may not be feasible nor desired, a sensible interpolation scheme should be devised and used.

To convert between signals with differing sampling rates, we utilize sinc interpolation (Oppenheim & Schafer, 2010), an established audio operation. Conceptually, we treat the intermediate hidden states of DiT as "samples" of multichannel audio, as if they were on a uniform sample grid over the underlying time axis. For our DiT-B setting (Table 3), this amounts to 768 sequences of length 394 that are thus resampled independently using a pre-cached matrix, as will be explained. Any point on this grid can be retrieved by adding several sinc functions at that point in "time"; each sinc is centered around where each DiT state is placed in "time", the height of each shall be the height of each sample, and the width of the main lobe is exactly two intervals between neighboring DiT states. At the end of the day, this is a mere linear combination, which can be precached. So, in practice, we cache a dense resampling matrix using a Hann-windowed sinc kernel, with each output row renormalized so that the operator is a proper weighted average rather than an unnormalized convolution. We assume both the source and target "positions" in time are in the center of what "range" that a 394-partition and 800-partition positions would occupy for the same span of time that the training sample occupies (32s).

[3] most commercial music is stereo, so an earphone which plays 48kHz music is actually going through 2×48000 values per second. Wow!

[4] One is now closed-source, and the other gives no indication about the data used to train the model.

We resample by upsampling the shorter, DiT-state sequence to the longer, MusicFM grid (800 vectors, from 32 seconds of audio), rather than the reverse. This yields an perfectly invertible transformation, while the reverse does not.

It should be clear by now that, for two sequences that match in length, this "resampling" procedure immediately falls back to an identity operation. Thus, our port of REPA is a very natural extension of the original algorithm which assumes matching sequence lengths, and beautifully align with fundamental signal processing ideas.

C.2 Configurations for Training ReDi and REG

For both, we re-use the same representations used for our proposed model and REPA: layer 7 output of MusicFM (Won et al., 2024). We use the "separate tokens" variant of ReDi as the baseline, both due to its similarity with our setup and its reported superior performance. The features are concatenated along the token axis after the same PCA projection used by our proposed run (see §B). They also have total variance aligned to the primary representations'. For REG, we input the whole sequence to MusicFM and take the mean of the feature vector as a proxy for global information that REG needs, as per the ablations (Wu et al., 2026) which showed this to be as effective. A motivation for this choice is that, to our knowledge, no well-validated self-supervised music model has any equivalents to a `cls` token that DINOv2 (Oquab et al., 2023) produces. We weight the separate denoising loss with factor $\beta = 0.03$.

D A More Formal Treatment of Training Gains via Asym-DiT

The lat-to-rep mask (Section 3.3) makes the latent $\mathbf{z}$ the only route through which auxiliary supervision can influence the generative computation. A natural failure mode is *shortcut collapse*: the auxiliary query may solve its denoising task from the auxiliary tokens alone and leave the latent unchanged. In this section, we formalize when a latent-mediated gain is available and characterize what the latent must preserve to realize that gain.

Setup. All quantities below are Bayes-optimal denoising floors. For a conditioning variable c, the minimum mean-squared error is $\operatorname{tr}\operatorname{Var}(\mathbf{r}_0 \mid c)$, attained by the posterior mean. Achieved network losses lie above these floors; we use their difference only as an empirical measure of the latent-mediated gain.

With abuse of notation, we write r_s^i for a noisy auxiliary i-th token and r_0^i for its clean target. We consider three conditioning levels: the auxiliary token alone, the auxiliary token together with the learned latent hidden states $\mathbf{H}$, and the auxiliary token together with the full noisy latent $\mathbf{z}_t$. Their floors are

$$\mathcal{L}_{\text{self}} := \mathbb{E}_i \mathbb{E}_{r_s^i} \operatorname{tr}\operatorname{Var}(r_0^i \mid r_s^i),$$
$$\mathcal{L}_{\text{full}} := \mathbb{E}_i \mathbb{E}_{r_s^i, \mathbf{H}} \operatorname{tr}\operatorname{Var}(r_0^i \mid r_s^i, \mathbf{H}),$$
$$\mathcal{L}^{\star}_{\text{full}} := \mathbb{E}_i \mathbb{E}_{r_s^i, \mathbf{z}_t} \operatorname{tr}\operatorname{Var}(r_0^i \mid r_s^i, \mathbf{z}_t).$$

We define the realized and available gains as

$$\Delta^{\text{r}} := \mathcal{L}_{\text{self}} - \mathcal{L}_{\text{full}},$$
$$\Delta^{\star} := \mathcal{L}_{\text{self}} - \mathcal{L}^{\star}_{\text{full}}.$$

Thus Δ^r measures the information exposed by the learned latent hidden states, while $\Delta^\star$ measures the information available in the noisy latent itself.

D.1 The Latent-Mediated Information Gain

By construction of the asymmetric mask,

$$\mathbf{H} = \mathbf{H}(\mathbf{z}_t; \boldsymbol{\theta}),$$

so $\mathbf{H}$ is computed solely from the latent path and cannot provide a hidden bypass through which the auxiliary representation re-enters the latent computation. Any reduction in the auxiliary Bayes floor obtained by reading $\mathbf{H}$ must therefore come from information exposed by the latent.

Applying the law of total variance conditioned on r_s^i gives

$$\Delta^r = \mathbb{E}_i \mathbb{E}_{r_s^i} \operatorname{tr} \operatorname{Var}\big(\mathbb{E}[r_0^i \mid r_s^i, \mathbf{H}] \,\big|\, r_s^i\big) .$$

Thus Δ^r is the additional information about $\mathbf{r}_0$ exposed by the latent hidden states $\mathbf{H}$ beyond the noisy auxiliary token $\mathbf{r}_s$. Since the forward path contains no representation-to-latent edge, the only route by which this supervision can shape the latent is the gradient of $\mathcal{L}_r$ through $\mathbf{H}$.

We cannot measure the Bayes floors directly. We therefore use the achieved-loss counterpart of Eq. 2, reproduced here for convenience:

$$\Delta := \underbrace{\mathbb{E}\big[\|v_{\mathbf{r}}^{\phi} - (\boldsymbol{\epsilon} - \mathbf{r}^{\text{pca}})\|_2^2\big]}_{\text{Term 1}} - \underbrace{\mathbb{E}\big[\|v_{\mathbf{r}} - (\boldsymbol{\epsilon} - \mathbf{r}^{\text{pca}})\|_2^2\big]}_{\text{Term 2}},$$

where $v_{\mathbf{r}} = v_{\boldsymbol{\theta}}(\mathbf{z}_t \oplus \mathbf{r}_s^{\text{pca}})$ is the normal rep output, and $v_{\mathbf{r}}^{\phi} = v_{\boldsymbol{\theta}}(\mathbf{r}_s^{\text{pca}})$ corresponds to when latent access is masked out. To avoid an out-of-distribution comparison, latent-access dropout is used during training so that the query is explicitly trained both with and without access to $\mathbf{H}$. Thus Δ is not an exact estimate of Δ^r, but $\Delta > 0$ directly shows that the trained query benefits from reading the latent. Figure 4 shows this achieved-loss gap.

D.2 A Positive Gain Is Attainable

Because $\mathbf{H}$ is a function of $\mathbf{z}_t$ alone, conditioned on $\mathbf{r}_s$, we have the Markov chain

$$\mathbf{r}_0 \to \mathbf{z}_t \to \mathbf{H}.$$

The data processing inequality therefore gives

$$0 \le \Delta^r \le \Delta^\star, \qquad \Delta^\star = \mathcal{L}_{\text{self}} - \mathcal{L}_{\text{full}}^\star.$$

Unlike Δ^r, the ceiling $\Delta^\star$ is fixed by the data distribution, the frozen encoders, and the corruption process; it does not depend on $\boldsymbol{\theta}$.

Whenever the noisy latent retains representation-relevant information not already contained in the noisy auxiliary token, $\Delta^\star > 0$. Thus zero gain is not the information-optimal solution: a positive amount of representation information is available through the latent. This establishes *availability*, not convergence of every optimization trajectory. The latter is an empirical question, which we probe with Δ. In particular, Figure 4 shows that the realized gain grows from near zero during training rather than remaining collapsed, indicating that the available latent-mediated channel is actually exploited.

D.3 The Information-Optimal Endpoint Is a Sufficient Statistic

If the available gain is fully realized,

$$\Delta^r = \Delta^\star,$$

then

$$\mathbb{E}_i \mathbb{E} \operatorname{tr} \operatorname{Var}(r_0^i \mid r_s^i, \mathbf{H}) = \mathbb{E}_i \mathbb{E} \operatorname{tr} \operatorname{Var}(r_0^i \mid r_s^i, \mathbf{z}_t).$$

In words, once r_s^i is given, $\mathbf{H}$ is as informative for recovering r_0^i as the full noisy latent $\mathbf{z}_t$. Thus $\mathbf{H}$ is a sufficient statistic of $\mathbf{z}_t$ for the auxiliary representation.

Crucially, sufficiency constrains information rather than representation form. Many hidden states may preserve the same information about $\mathbf{r}_0$, so the auxiliary objective specifies an equivalence class rather than a single target. REPA, in contrast, constrains a particular $\mathbf{h} \in \mathbf{H}$,

$$\text{MLP}(\mathbf{h}) \approx \mathbf{r}_0,$$

whereas ours constrains *recoverable information*. The generation objective is therefore free to choose, among information-equivalent hidden states, those that are also useful for denoising the latent. This flexibility provides the mechanism by which auxiliary supervision can remain compatible with the generative objective without prescribing the form of the latent representation; whether this leads to cooperative optimization dynamics is evaluated in Section 6.

D.4 Controlling the Supervision Regime

The mask and the decoupled noise schedule provide two complementary controls over how the query uses the latent. The mask removes an auxiliary-side shortcut, while the noise schedule changes the relative corruption and hence the regime in which the auxiliary query must rely on the latent.

Cross-rep masking. Under the mask, each auxiliary query sees only its own noisy token:

$$\mathcal{L}_{\text{self}} = \mathbb{E}_i \mathbb{E}\, \text{tr}\, \text{Var}(r_0^i \mid r_s^i).$$

Removing the mask allows a query to use neighboring auxiliary tokens, giving the smaller floor

$$\mathbb{E}_i \mathbb{E}\, \text{tr}\, \text{Var}(r_0^i \mid \mathbf{r}_s).$$

Because tokens from the same pretrained encoder are correlated, this shortcut reduces the uncertainty that must be supplied by the latent and narrows the available gain. The latent-conditioned floor is expected to change much less because $\mathbf{z}_t$ and neighboring auxiliary tokens are both views of the same underlying data. The mask therefore preserves a larger latent-mediated gain, consistent with the reduction in Δ when it is removed; see Figure 5b.

Decoupled noise schedule. We tie the auxiliary noise level s to the latent noise level t through

$$s = \frac{\alpha t}{1 + (\alpha - 1)t}.$$

Increasing α makes the auxiliary representation noisier than the latent at the same training step, changing how much of the auxiliary task must be solved from the latent side. Unlike the cross-rep mask, however, the effect of α on the individual Bayes floors is not monotone: changing s alters both the information available from the auxiliary token and the difficulty of the auxiliary prediction itself. We therefore view α as a control of the supervision regime rather than as a parameter that monotonically enlarges $\Delta^\star$.

At small α, the auxiliary token remains relatively informative and the latent is weakly pressured; at large α, the query receives a stronger latent-dependent signal but the auxiliary task also becomes harder to realize. We therefore expect an intermediate regime to be most effective, which is evaluated empirically in Figure 5b.

E Miscellaneous Tables

Table 9: FAD (↓) and CLAPscores (↑) on a filtered instrumental subset of Song Describer (SDD) dataset.

			FAD ↓				CLAP ↑	
System	RLHF?	Size	MuQ	MusicFM	L-CLAP-M	MuQ-MuLan	L-CLAP-M	MuQ-MuLan
MusicGen-large		3.3B	36.546	28.180	0.214	0.211	0.295	0.279
ACE-Step (base)		2B	33.055	35.808	0.261	0.368	0.218	0.150
Stable Audio 3 Medium-Base		1.4B	21.507	18.471	0.132	0.104	0.325	0.341
Stable Audio Open		1B	51.406	33.224	0.166	0.274	0.335	0.217
DiffRhythm2		1B	29.387	27.554	0.173	0.134	0.311	0.348
Stable Audio 3 Small-Music-Base		433M	30.336	23.080	0.177	0.178	0.329	0.293
Ours DiT-XXL@300k		1.7B	34.330	26.410	0.138	0.107	0.329	0.311
Ours DiT-XXL@250k + Asym-DiT 2.5k		1.7B	29.292	23.909	0.138	0.103	0.330	0.325
Ours DiT-B@400k		179M	50.440	34.313	0.146	0.161	0.322	0.271
Ours DiT-B@300k + Asym-DiT 10k		179M	39.314	25.745	0.138	0.135	0.328	0.298
ACE-Step (turbo)	✓	2B	22.560	25.178	0.221	0.260	0.260	0.224
Stable Audio 3 Medium	✓	1.4B	16.030	16.971	0.100	0.104	0.336	0.362
Stable Audio 3 Small-Music	✓	433M	23.435	18.442	0.143	0.162	0.339	0.335

* As different models have vastly different conditioning and generation mechanisms, any comparisons are inherently unfair. Parameter counts exclude any VAE or language models.

F Attention and Cross-Rep Masking: A Visualization

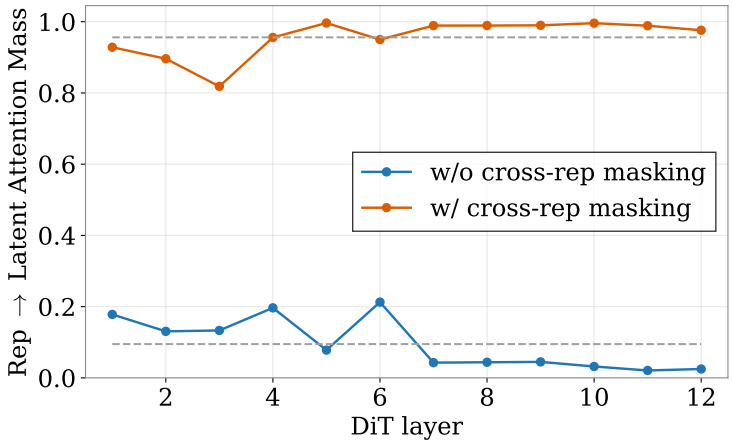


Figure 6: Layer-wise rep-to-latent attention trends by layer.

Fig. 6 plots rep-to-latent attention trends. The figure was produced at 125k steps using the proposed run (Table 4) and an ablation with cross-rep masking turned off (Fig. 5b). Without cross-rep masking, the auxiliary representations primarily attend amongst themselves, especially in the later layers. In contrast, the proposed model flips this trend to enable supervision (Section 4).

We furthermore present a qualitative demonstration of how the attention patterns differ with and without inter-rep masking. Conceptually, in a well-trained Asym-DiT, at least some attention heads should exhibit strong temporal correspondence between the latents and auxiliary representations. One example of such a pattern would be an upper-right-heading diagonal line that connects the lower left and upper right corners of the lower-right quadrant, for example as seen in Fig. 3. This is because any point on such a diagonal would be the intersection of the same temporal point on the latent and auxiliary representation sequences. Another example would be patters where the lower-right quadrant is an somewhat elongated and alternately-colored version of the lower-left. Conversely, if the lower right quadrant of a visualization is occupied by diagonal patterns parallel to the "main diagonal" of the square, one can assume there are little to no temporal relations leveraged in that head.

In Asym-DiT, we primarily find such patterns to have developed in the middle layers, especially in layers 7, which we visualize in Fig. 7. In the model trained without cross-rep masks, such patterns are not too visible.

Further attention score visualizations are given below, with individual captions indicating rep-to-latent post-softmax attention scores. For this figure and Fig. 3, the upper-right quadrant of the with-inter-rep-masking figures have been max-pooled so that the thin diagonal could be visualized at all.

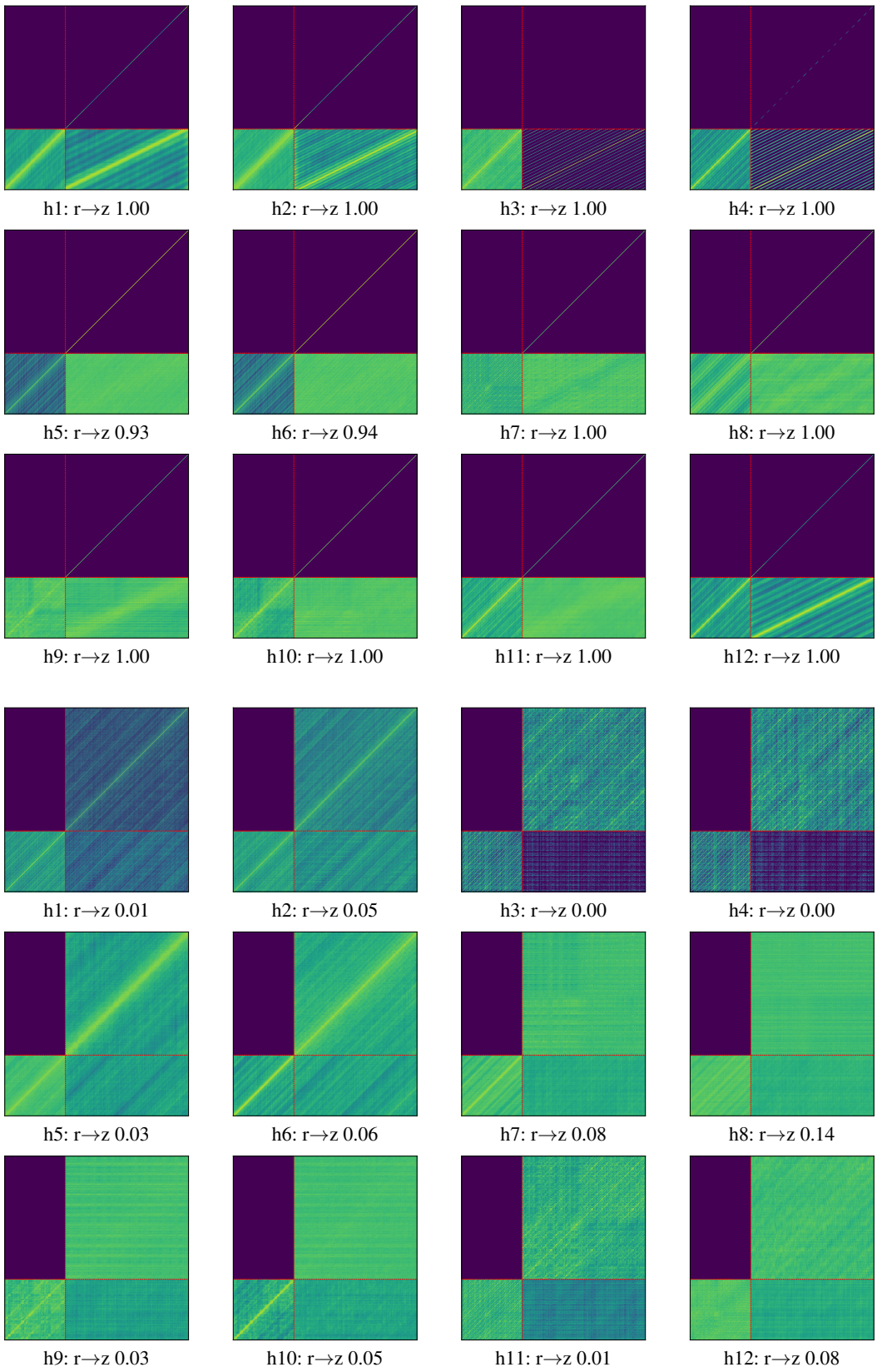


Figure 7: Attention maps for layer 7 on two models trained with (↑) and without (↓) asymmetric masking. Post-softmax rep-to-latent attention scores are indicated.