CAPACITY IS NOT ACCESSIBILITY IN AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

The high expressivity of trained decoders in autoencoder models is usually attributed to their learned weights. However, we find that even random convolutional decoders can achieve high-quality reconstructions when sufficiently high-dimensional intermediate representations are optimized directly. This raises the question: what does learning contribute if substantial capacity is already present in random decoders? We decompose autoencoder reconstruction errors into three components: a decoder-limited error, an encoder gap between the best bottleneck state and that predicted by the encoder, and an accessibility gap between the best intermediate representation and the best state reachable through the bottleneck. Across image and audio datasets, we show that increasing bottleneck dimensionality systematically reduces the accessibility gap while increasing the encoder gap, leaving the total gap comparatively stable. Training therefore does not merely provide capacity, it organizes high-dimensional capacity so that useful outputs can be reached through far fewer control variables. We further find that the accessibility gap spans many functionally relevant decoder directions and can extend under distribution shift. Together, these results suggest that poor reconstruction through a trained decoder can reflect restricted access to its capability rather than insufficient decoder capacity.

1 INTRODUCTION

Reconstruction performance in autoencoders is usually credited to the quality of the bottleneck representation and the ability of the decoder to map that representation back to the data space (Hinton & Salakhutdinov, 2006). However, this view does not clearly distinguish between what a decoder is capable of representing and what its learned bottleneck can actually access. This distinction becomes important when considering the expressivity of neural decoders. High reconstruction quality is often taken as evidence that the decoder has learned a rich model of the data. Yet neural networks are themselves highly expressive function classes (Lempitsky et al., 2018; Heckel & Hand, 2018); thus it is not obvious how much reconstruction capacity should be attributed to learned weights rather than to the dimensionality and architecture of the representation used to control them. Separating raw decoder capacity from learned accessibility is therefore necessary to understand what training actually contributes.

We study this problem by treating the bottleneck representation and an intermediate decoder representation as two nested levels of access to the same downstream network. This allows us to separate three sources of reconstruction performance: the best reconstruction supported by the decoder under direct control of its intermediate representation; the accessibility gap, measuring how much of this capability cannot be reached through the bottleneck; and the encoder gap, measuring how far the encoder is from generating the best representation available within that bottleneck. These gaps reflect fundamentally different limitations. The encoder gap can, in principle, be reduced by better inference within the same representation space; whereas the accessibility gap cannot, it reflects a restriction on which downstream states are not reachable through that representation at all.

We show that high-dimensional control can provide substantial reconstruction capacity even in random convolutional decoders, whereas autoencoder training makes useful reconstructions accessible through much lower-dimensional representations. The remaining accessibility gap spans many functionally relevant decoder directions and can persist under distribution shift. Together, these results suggest that autoencoder training should be understood not only in terms of acquiring reconstruction capacity, but also in terms of organizing and exposing that capacity through a constrained represen-

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.

tation. Although our experiments focus on autoencoders, the accessibility problem is not inherently specific to reconstruction models. Any architecture in which a learned representation provides restricted access to a larger downstream computation could exhibit a similar gap between downstream capability and what the representation can reach. Autoencoders make this distinction unusually easy to isolate because the encoder and decoder can be manipulated independently.

2 RESULTS

2.1 RECONSTRUCTION CAPACITY EXISTS WITHOUT TRAINING

Define the structure of an autoencoder as

$$x \xrightarrow{E} z \xrightarrow{F} h \xrightarrow{G} \hat{x},$$

where x is the input, E is the encoder, z is the bottleneck representation (latent code), h is an intermediate decoder representation, and G denotes the remaining decoder that produces the reconstruction $\hat{x}$. Directly optimizing h removes the constraint imposed by the bottleneck and therefore allows us to probe the reconstruction capacity supported by the decoder.

We first asked how much of this capacity requires learned decoder weights. We tested two decoder architectures: a small decoder with three convolutional layers (Fig. 1a,b) and the pretrained Stable Diffusion VAE decoder (Stability AI, 2022) (Fig. 1c). For the small decoder, we randomly initialized and froze all decoder weights and reconstructed CIFAR-10 and MNIST images by optimizing only h, while varying its dimensionality. At low h dimensionality, random decoders performed substantially worse than the trained decoder. However, once h reached approximately $5\times$ the image dimensionality ($2.5\times$ the trained h dimension), random decoders matched or surpassed the reconstruction performance of trained decoders, with further improvements at larger h. We observed the same effect in the much larger Stable Diffusion VAE. When optimization was restricted to the low-dimensional bottleneck z ($0.02\times$ the image dimensionality), the trained decoder achieved approximately $50\times$ lower reconstruction error than randomly initialized decoders of the same architecture. In contrast, when optimization was performed directly on the intermediate decoder representation h ($2.67\times$ the image dimensionality), the random decoders achieved reconstruction error comparable to, and slightly lower than, the trained decoder (Fig. 1c). Example reconstructions from the trained and random decoders are shown in Fig. S1.

These results show that substantial reconstruction capacity can arise from high-dimensional control even in the absence of learned decoder weights, but trained decoders achieve comparable reconstruction with far fewer control dimensions. Reconstruction capacity alone therefore cannot explain what autoencoder training accomplishes.

2.2 DECOMPOSING RECONSTRUCTION ERROR

Next, we ask how autoencoder training makes this capacity usable through a compact bottleneck representation. To distinguish limitations of the encoder from limitations imposed by the bottleneck itself, we decompose reconstruction error into three components.

Define the reconstruction loss obtained from the trained encoder as

$$L_{\text{rec}} = L(G(F(E(x))), x) \tag{1}$$

the best reconstruction obtainable through the bottleneck as

$$L_z^* = \min_z L(G(F(z)), x) \tag{2}$$

and the best reconstruction obtained by directly optimizing the intermediate representation h at the chosen decoder partition as

$$L_h^* = \min_h L(G(h), x) \tag{3}$$

The total reconstruction loss can then be written as

$$L_{\text{rec}} = L_h^* + (L_z^* - L_h^*) + (L_{\text{rec}} - L_z^*) \tag{4}$$

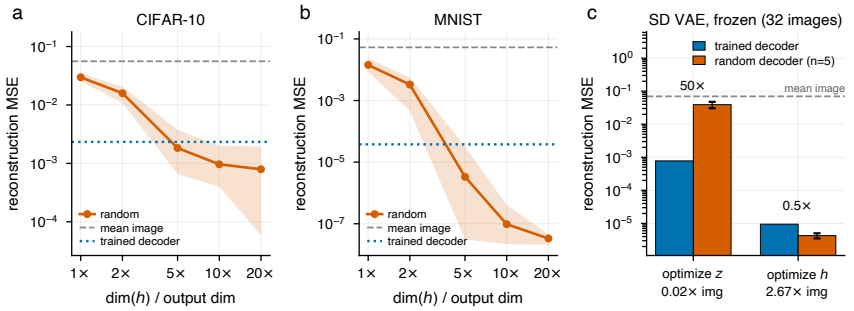


Figure 1: High-dimensional control enables accurate reconstruction through random convolutional decoders. a) CIFAR-10 images are reconstructed using a randomly initialized decoder consisting of a fully connected intermediate representation h followed by three convolutional layers. All network weights are fixed, and reconstruction is performed by directly optimizing the values of h. The dimensionality of h is varied from $1\times$ to $20\times$ the output dimensionality. MSE is averaged over 16 reconstructed images for each random initialization; the line shows the mean and shading the standard deviation across 10 random seeds. The gray dashed line shows the MSE of the mean-image baseline, and the blue dotted line shows the MSE obtained by optimizing h in a trained decoder with h dimensionality $2\times$ the output dimensionality. b) Same experiment as in a), using MNIST. c) Flowers102 (Nilsback & Zisserman, 2008) images are reconstructed using the frozen Stable Diffusion VAE decoder. The trained decoder uses the pretrained weights (Stability AI, 2022), whereas random decoders use the same architecture with decoder weights reinitialized using Kaiming initialization. Reconstruction is performed by optimizing either the bottleneck representation z, with dimensionality $0.02\times$ the image dimensionality, or an intermediate decoder representation h, with dimensionality $2.67\times$ the image dimensionality. Bars show mean MSE over 32 images; error bars indicate variation across five random decoder initializations. For CIFAR-10 and Flowers102, the reconstruction was performed independently per RGB channel.

Here, L_h^* is the decoder-limited reconstruction error floor: the error that remains even when h is optimized independently for each input at the chosen intermediate decoder layer. The encoder gap, $L_{\text{rec}} - L_z^*$, measures how much reconstruction performance is lost because the feed-forward encoder does not produce the reconstruction-optimal bottleneck code. This term is analogous to the amortization gap in amortized inference (Cremer et al., 2018), although here it is defined directly in terms of reconstruction error. The accessibility gap, $L_z^* - L_h^*$, instead measures the restriction imposed by the bottleneck-accessible family of decoder representations. In other words, the encoder gap asks whether the encoder finds the best code available within the bottleneck, whereas the accessibility gap asks how much reconstruction capability becomes available when access is expanded from the bottleneck to a given interface.

We measured these components in deterministic and variational autoencoders across image and audio datasets (Fig. 2). In the deterministic autoencoders, optimizing z produced only modest improvements over the encoder output in most datasets, with reductions of approximately $1.1\times$ for SVHN, CIFAR-10, and Speech (Panayotov et al., 2015), and $1.7\times$ for MNIST. In contrast, optimizing h beyond the best z reduced reconstruction error by $5.0\times$ to $27.1\times$ across all datasets (Fig. 2a). The same qualitative pattern was observed in variational autoencoders (Fig. 2b).

These results show that the trained encoder often captures much of the reconstruction performance available within its bottleneck representation, while substantially better reconstructions remain available to the downstream decoder outside the bottleneck-accessible family. Standard autoencoder training therefore does not necessarily make the full reconstruction capability of the learned decoder accessible through its bottleneck.

2.3 The Accessibility Gap Is High-Dimensional

We next asked whether the difference between the native decoder representation $h^0 = F(E(x))$ and the optimized representation h^* could be explained by a small number of missing directions. We analyzed deterministic autoencoders with a three-layer convolutional encoder (and decoder), a

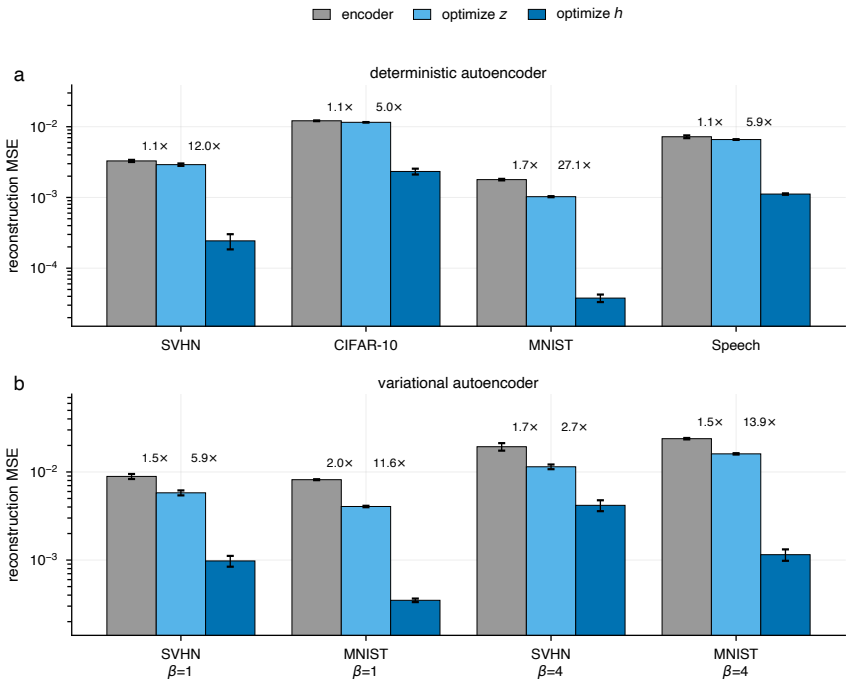


Figure 2: A substantial accessibility gap remains after autoencoder training. a) Reconstruction MSE for deterministic autoencoders across four benchmark datasets. Gray bars show reconstruction using the latent code produced by the trained encoder. Light blue bars show reconstruction after optimizing the latent code z independently for each input while keeping all model weights fixed. Dark blue bars show reconstruction after directly optimizing the intermediate decoder representation h for each input, again with all weights fixed. Numbers above the bars indicate the fold reduction in MSE from encoder output to optimized z, and from optimized z to optimized h, respectively. Bars show mean reconstruction MSE and error bars indicate standard deviation across 10 random seeds. b) Same analysis for variational autoencoders with the same architecture as in a, using $\beta = 1$ and $\beta = 4$.

32-dimensional bottleneck z, and a 2048-dimensional intermediate decoder representation h. This is the standard structure of autoencoder models tested in this paper unless otherwise specified. The same analysis was repeated across benchmark datasets.

For each input, we define the residual between the optimized and native intermediate representations as:

$$\Delta h = h^* - h^0 \tag{5}$$

We pooled Δh across test samples and performed principal component analysis. Across all four datasets, more than 100 PCs were required to explain 95% of the variance in Δh, substantially exceeding the bottleneck dimensionality of 32 (Fig. 3a). This indicates that the difference between h^0 and h^* is distributed across many linear directions in the intermediate representation space.

We then tested whether these residual directions are functionally important for reconstruction. We progressively reconstructed the residual using its first r principal components,

$$\hat{h}_i(r) = h_i^0 + B_r B_r^\top (h_i^* - h_i^0), \tag{6}$$

where the columns of B_r are the first r residual PCs. We then used $\hat{h}_i(r)$ for reconstruction and measured the fraction of the improvement from h^0 to h^* that was recovered using the average MSE across all tested images:

$$R(r) = \frac{\bar{L}(h^0) - \bar{L}(\hat{h}(r))}{\bar{L}(h^0) - \bar{L}(h^*)}. \tag{7}$$

Across most datasets, 32 residual PCs recovered only a small fraction of the improvement obtained by directly optimizing h, and hundreds of PCs were required to approach the reconstruction performance of h^* (Fig 3b). At very low ranks, partial residual reconstruction could even worsen the reconstruction error. Thus, the high dimensionality of Δh is not simply due to variance in directions that are irrelevant to the decoder; many of these directions contribute functionally to the reconstruction improvement.

We then compared the intrinsic dimensionality of h^0, h^*, and Δh directly. The number of PCs required to explain 95% of the variance in h^0 was close to the bottleneck dimensionality of 32 for most datasets, consistent with h^0 being generated from the 32-dimensional bottleneck through the learned fully connected mapping. In contrast, both h^* and Δh required substantially more PCs to explain the same fraction of variance (Fig 3c). This shows that direct optimization of h accesses a much broader set of linear directions than those normally expressed through the bottleneck.

Finally, we asked how much of the residual Δh lies within the subspace directly accessible through the learned $z \to h$ mapping. For the weight matrix W, we obtained an orthonormal basis Q for its column space using QR factorization and calculated

$$\text{accessible variance} = \|\Delta H Q\|^2 / \|\Delta H\|^2, \tag{8}$$

where ΔH contains the residuals across samples. Across datasets, only a small fraction of residual variance lay within the column space of W (Fig 3d), although this fraction was consistently above the expectation for a random 32-dimensional subspace.

Together, these results show that the difference between the native and optimized intermediate decoder representations is not a simple low-dimensional correction. Instead, the reconstruction improvement obtained by direct access to h depends on coordinated changes across many decoder-relevant directions, most of which lie outside the subspace normally accessible through the learned bottleneck route (z).

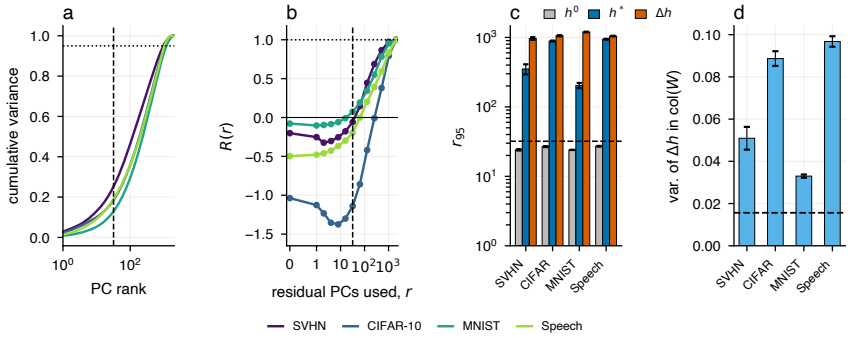


Figure 3: The accessibility gap is high-dimensional. a) Cumulative variance explained by principal components (PCs) computed from the residuals $\Delta h = h^* - h^0$ across all tested samples. The horizontal dotted line marks 95% explained variance, and the vertical dashed line marks $r = 32$, corresponding to the bottleneck dimensionality $d_z = 32$. b) Fraction of the reconstruction improvement from h^0 to h^* recovered after adding the contribution of the first r residual PCs to h^0. The vertical dashed line marks $r = 32$. c) Number of PCs required to explain 95% of the variance (r_{95}) in h^0, h^*, and Δh across datasets. The horizontal dashed line marks $r_{95} = 32$. d) Fraction of the variance in Δh lying within the column space of the learned weight matrix W connecting z to h. The dashed horizontal line indicates the expected fraction for a random 32-dimensional subspace of the 2048-dimensional h space.

2.4 Bottleneck width redistributes reconstruction error

To determine if and when the accessibility gap closes, we varied the relative bottleneck width z/h while independently testing three intermediate representation sizes, $h = 512, 1024$, and 2048, across CIFAR-10 and Speech. We repeated the sweep for deterministic AEs and VAEs with $\beta = 1$ and $\beta = 4$. Across all three model classes, increasing z/h consistently reduced the normalized accessibility gap (Fig 4a,d,g). In deterministic AEs, the normalized accessibility gap approached zero

as $z/h \to 1$; the same trend was observed in VAEs, although a small residual gap remained for some configurations. In contrast, the encoder gap increased with z/h (Fig. 4b,e,h). Consequently, the total difference between encoder reconstruction and the decoder optimum changed much less systematically across the sweep (Fig. 4c,f,i). Thus, increasing bottleneck dimensionality does not simply eliminate reconstruction suboptimality. Instead, it redistributes the dominant limitation from restricted accessibility at narrow bottlenecks toward encoder suboptimality at wider bottlenecks. The consistency of this trade-off across deterministic and variational autoencoders suggests that it is not specific to a particular autoencoder objective.

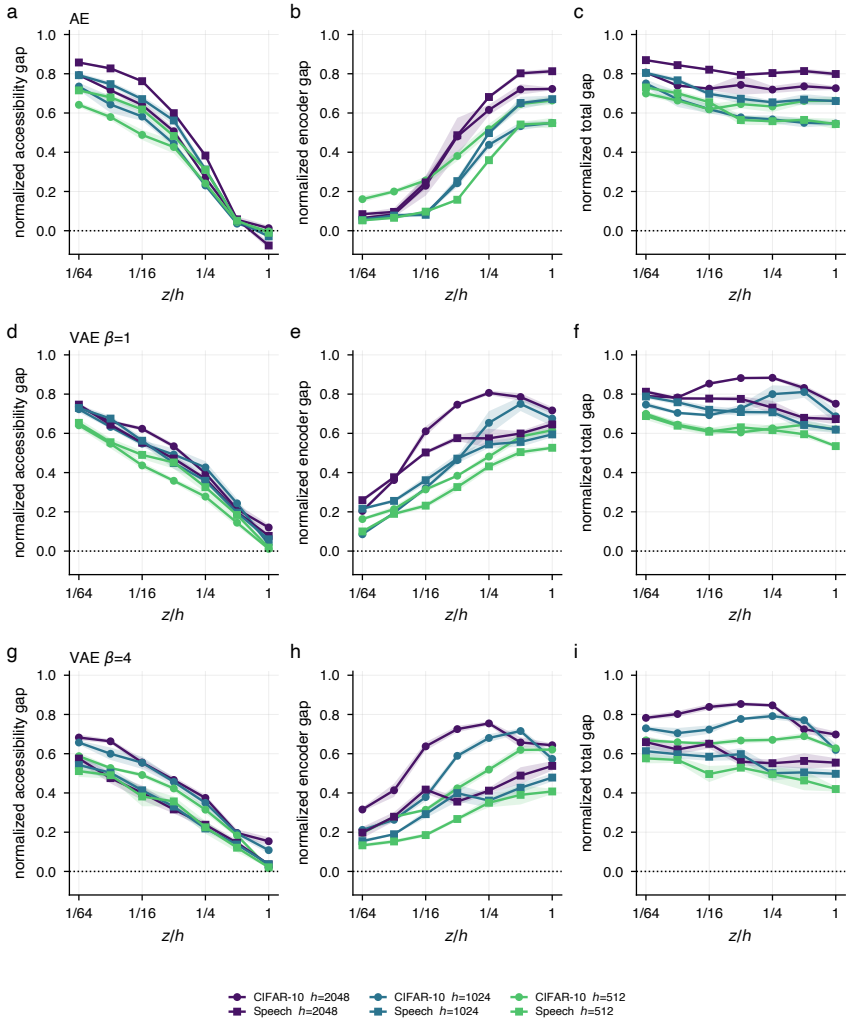


Figure 4: Bottleneck width redistributes reconstruction error. a–c) Deterministic AE, d–f) VAE with $\beta = 1$, and g–i) VAE with $\beta = 4$. a,d,g) normalized accessibility gap, $(L(z^*) - L(h^*))/L(z^*)$. b,e,h) Normalized encoder gap, $(L(z^0 - L(z^*))/L(z^0)$. c,f,i) Normalized total gap, $(L(z^0) - L(h^*))/L(z^0)$. Results are shown for CIFAR-10 and Speech with intermediate representation sizes $h = 512, 1024$, and 2048. Lines show the mean across five random seeds and shaded regions indicate the standard deviation. Unnormalized losses show the same decrease in the accessibility gap and increase in encoder error across much of the scan (Fig. S3). The absolute decoder floor also varies with z/h, although its dependence differs between deterministic and variational autoencoders.

2.5 The accessibility gap extends beyond the training distribution

We next asked whether the additional decoder capacity revealed by direct h-optimization is limited to the training distribution or can also support shifted target distributions. We tested four

source–target dataset pairs spanning image and audio domain shifts, comparing zero-shot reconstruction, per-sample optimization of z or h, and several parameter-adaptation strategies that retrain different parts of the autoencoder.

Direct optimization of h revealed substantial target-domain reconstruction capacity in the frozen source-trained decoder (Fig 5b). This was particularly pronounced for CIFAR-10 $\rightarrow$ STL and Speech M $\rightarrow$ F, where h-optimization substantially outperformed both latent optimization and the learned adaptation methods. Learned adaptation performance depended strongly on the target dataset: retraining the decoder or full model worked well for SVHN $\rightarrow$ MNIST and MNIST $\rightarrow$ USPS, but provided little improvement for CIFAR-10 $\rightarrow$ STL. Moreover, modifying model weights generally degraded source-domain reconstruction, sometimes by orders of magnitude (Fig 5a), whereas per-sample optimization of z or h leaves the pretrained model unchanged (their negative source-MSE changes reflect improved source reconstruction under per-sample optimization rather than changes in the stored source model).

MNIST $\rightarrow$ USPS provides a clear qualitative example (Fig. S2). Although both datasets contain handwritten digits, USPS has smoother grayscale statistics than MNIST. Methods constrained by the original $z \rightarrow h$ mapping retained much of the MNIST-like appearance, whereas directly optimizing h, or adapting the route from z to h, reproduced the USPS statistics more closely.

Together, these results show that a source-trained decoder can support target-domain reconstructions that are not accessible through its source-trained bottleneck representation. The accessibility gap therefore extends beyond the training distribution: poor zero-shot reconstruction does not necessarily imply that the downstream decoder is incapable of representing the target. Instead, part of the limitation can arise from the restricted route through which the decoder is normally accessed.

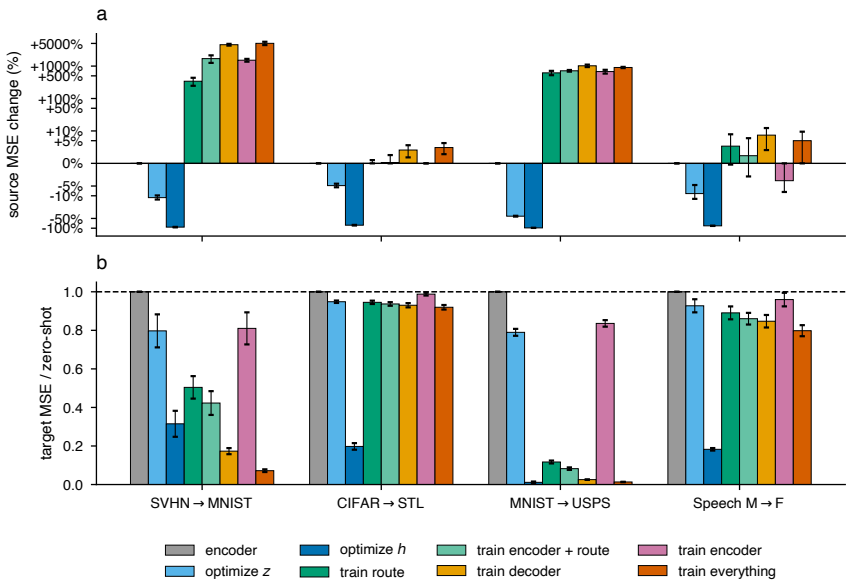


Figure 5: Accessibility under domain transfer. a) Percentage change in source-domain reconstruction MSE after target-domain adaptation across four source–target dataset pairs. Positive values indicate increased source reconstruction error after adaptation, whereas negative values indicate improved reconstruction relative to the source-domain encoder baseline. Bars show the mean and error bars the standard deviation across 10 random seeds. b) Target-domain reconstruction MSE normalized by the zero-shot encoder reconstruction MSE for each transfer pair. A value of 1 corresponds to the source-trained encoder baseline, and lower values indicate improved target reconstruction. Optimize z and optimize h perform per-sample optimization without modifying model weights; train route optimizes the weight matrix W between z and h; the remaining adaptation conditions update the indicated components of the pretrained autoencoder. Bars show the mean and error bars the standard deviation across 10 random seeds.

3 Related Work

3.1 Untrained networks and architectural capacity

The expressive power of neural networks does not arise exclusively from trained weights. The lottery ticket hypothesis suggests that inside over-parameterized neural networks exist subnetworks that have comparable accuracy to a dense target network with learned weights (Frankle & Carbin, 2018; Zhou et al., 2019; Ramanujan et al., 2020; Malach et al., 2020; Pensia et al., 2020; Orseau et al., 2020). More broadly, Gaier & Ha (2019) showed that network topology alone can encode task solutions even when all weights share a single random value. In generative networks, Lempitsky et al. (2018) showed that the architecture of an untrained convolutional network itself captures substantial low-level image statistics, enabling image restoration without dataset-level pretraining — an effect Heckel & Soltanolkotabi (2020) later attributed formally to the structural bias of convolutional architectures. Heckel & Hand (2018) similarly demonstrated that untrained neural architectures can provide effective low-dimensional image representations. These works optimize network parameters for individual images, whereas we keep randomly initialized decoder weights fixed and optimize only their intermediate representations. This allows us to isolate reconstruction capacity provided by the dimensionality and architecture of the decoder control space from capacity acquired through weight learning.

3.2 Inference suboptimality and amortization

Cremer et al. (2018) and Krishnan et al. (2018) decomposed VAE inference error into an approximation gap, arising from limitations of the variational family, and an amortization gap, arising when the recognition network fails to find the best member of that family. They found that recognition-network suboptimality can account for a substantial fraction of inference error and that the learned generator adapts to restrictions imposed by the approximate posterior. Subsequent works have been focusing on reducing the amortization gap (Kim et al., 2018; Marino et al., 2018) and regularization (Shu et al., 2018). Margossian & Blei (2023) characterized conditions under which the gap can be closed entirely. Our encoder gap is analogous in spirit to amortization gap, while measured directly in reconstruction space. The accessibility gap, on the other hand, addresses a totally different restriction: whether any bottleneck code can reach the decoder representation required for optimal reconstruction.

3.3 Intermediate-layer optimization and generator inversion

Prior work on generative-model inversion has shown that intermediate representations can be substantially more expressive than native latent spaces (Abdal et al., 2019; 2020; Daras et al., 2022; 2021; Gu et al., 2020; Bora et al., 2017; Dhar et al., 2018; Roich et al., 2022; Gunn et al., 2022). Abdal et al. (2019; 2020) demonstrated that optimizing in StyleGAN's layer-wise intermediate space yields dramatically improved reconstruction over the native latent space, and Pividori et al. (2019) found that optimizing representations after the first dense layer of a GAN generator enables reconstructions that cannot be obtained from the original latent space. A series of works formalized this idea in Intermediate Layer Optimization, progressively moving optimization into deeper generator layers to enlarge the feasible family for inverse problems (Daras et al., 2021; 2022; Gunn et al., 2022). These works primarily exploit intermediate-layer expressivity as an inversion method. We instead treat the discrepancy between nested representation spaces as an object of study: we separate it from encoder suboptimality, manipulate the relative dimensions of the bottleneck and intermediate representation, and characterize the geometry of the resulting accessibility gap.

3.4 Autoencoder bottlenecks and modern visual tokenizers

The dimensionality and structure of autoencoder bottlenecks have recently received renewed attention because learned tokenizers form the interface between pixels and modern generative models (Rombach et al., 2022). Hansen-Estruch et al. (2025) systematically scaled visual tokenizers and found that increasing bottleneck capacity improves reconstruction, while encoder and decoder scaling have different effects on reconstruction and downstream generation. Other recent work emphasizes that richer latent representations are not unambiguously beneficial. Yao et al. (2025)

showed that increasing latent feature dimensionality improves reconstruction but makes the latent distribution more difficult for diffusion models to learn — a tradeoff also observed in the design of Stable Diffusion 3 (Esser et al., 2024). Skorokhodov et al. (2025) identified latent spectral properties associated with poor downstream diffusability, particularly for wider bottlenecks. Representation Autoencoders similarly replace conventional low-dimensional VAE latents with richer pretrained representations, but require new mechanisms to make high-dimensional representations tractable for diffusion modeling (Yu et al., 2024; Zheng et al., 2026). Whereas these studies ask how bottleneck design affects reconstruction or downstream generation, we ask how bottleneck dimensionality changes where reconstruction suboptimality arises—between decoder capacity, accessibility through the bottleneck, and the encoder's ability to identify the best accessible state.

4 Discussion

Our results separate three sources of autoencoder reconstruction performance: decoder capacity, the encoder gap, and the accessibility gap. These sources are usually evaluated together, but behave differently as bottleneck dimensionality changes (Fig. 4). Narrow bottlenecks restrict which decoder states can be reached, producing a large accessibility gap, whereas wider bottlenecks reduce this restriction but increase encoder error. This trade-off suggests a structural tension in amortized representation learning between making a representation sufficiently expressive and making its useful states easy to identify. Whether this balance generalizes beyond the architectures studied here remains an open question.

Related observations have appeared separately in prior work. Untrained convolutional networks can possess substantial representational capacity, while recent visual-tokenizer and latent-diffusion studies have found that increasing latent dimensionality improves reconstruction but makes latent representations harder to model or learn. Our contribution is to connect these observations through a common decomposition and show systematically how reconstruction error shifts between accessibility and inference as bottleneck width changes.

The random-decoder experiments further suggest that learning does not simply create reconstruction capacity. Rather, it organizes existing capacity so that useful outputs become reachable through far fewer control variables. Under distribution shift, the same distinction matters as well: poor transfer may sometimes reflect an access problem rather than an absence of decoder capability.

Our analysis has several limitations. First, direct h-optimization is a diagnostic tool rather than a practical method, and the learned adaptation methods we tested did not consistently recover its performance. Second, strong reconstruction from very high-dimensional random representations also does not imply generative usefulness, since these representations are optimized for known targets rather than sampled or predicted. Finally, our experiments focus on convolutional autoencoders and a limited set of image and audio datasets, so the generality of these findings across architectures and objectives remains to be tested.

In summary, our results show that autoencoder reconstruction quality reflects not only decoder capacity, but also how that capacity is utilized through the bottleneck and accessed by the encoder. Bottleneck dimensionality redistributes error between accessibility and inference, while learning makes useful decoder states reachable through far fewer degrees of freedom. This distinction may be important beyond autoencoders, where the performance exposed through a learned representation need not fully reflect the capability of the downstream network.

AI use statement

In this work, we used generative AI tools for design or provide feedback on research methodology or experiments, implement methods, interpret results. We have not used generative AI tools for help develop theoretical models or conceptual frameworks, propose or refine hypotheses. Clean and reformat dataset, generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, support qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for summarize or analyse existing literature, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, propose a title or keywords for a research paper. We have reviewed all AI-assisted work. LLM-generated code

was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4431–4440. IEEE, 2019.

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8293–8302. IEEE, 2020.

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed sensing using generative models. In *International conference on machine learning*, pp. 537–546. PMLR, 2017.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International conference on machine learning*, pp. 1078–1086. PMLR, 2018.

Giannis Daras, Joseph Dean, Ajil Jalal, and Alexandros G Dimakis. Intermediate layer optimization for inverse problems using deep generative models. *arXiv preprint arXiv:2102.07364*, 2021.

Giannis Daras, Yuval Dagan, Alexandros G Dimakis, and Constantinos Daskalakis. Score-guided intermediate layer optimization: Fast langevin mixing for inverse problems. *arXiv preprint arXiv:2206.09104*, 2022.

Manik Dhar, Aditya Grover, and Stefano Ermon. Modeling sparse deviations for compressed sensing using generative models. In *International Conference on Machine Learning*, pp. 1214–1223. PMLR, 2018.

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 *Forty-first international conference on machine learning*, 2024.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019.

Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3009–3018. IEEE, 2020.

Sean Gunn, Jorio Cocola, and Paul Hand. Regularized training of intermediate layers for generative models for inverse problems. *arXiv preprint arXiv:2203.04382*, 2022.

Philippe Hansen-Estruch, David Yan, Ching-Yao Chung, Orr Zohar, Jialiang Wang, Tingbo Hou, Tao Xu, Sriram Vishwanath, Peter Vajda, and Xinlei Chen. Learnings from scaling visual tokenizers for reconstruction and generation. *arXiv preprint arXiv:2501.09755*, 2025.

Reinhard Heckel and Paul Hand. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv preprint arXiv:1810.03982*, 2018.

Reinhard Heckel and Mahdi Soltanolkotabi. Compressive sensing with un-trained neural networks: Gradient descent finds a smooth approximation. In *International conference on machine learning*, pp. 4149–4158. PMLR, 2020.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Rahul Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *International conference on artificial intelligence and statistics*, pp. 143–151. PMLR, 2018.

Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov. Deep image prior. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454. IEEE, 2018.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International conference on machine learning*, pp. 6682–6691. PMLR, 2020.

Charles C Margossian and David M Blei. Amortized variational inference: When and why? *arXiv preprint arXiv:2307.11018*, 2023.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pp. 3403–3412. PMLR, 2018.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33:2925–2934, 2020.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in neural information processing systems*, 33:2599–2610, 2020.

Marcos Pividori, Guillermo L Grinblat, and Lucas C Uzal. Exploiting gan internal capacity for high-quality reconstruction of natural images. *arXiv preprint arXiv:1911.05630*, 2019.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network?(2020). *arXiv preprint cs.CV/1911.13299*, 2020.

Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on graphics (TOG)*, 42(1):1–13, 2022.

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 (CVPR)*, pp. 10684–10695, 2022.

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems*, 31, 2018.

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. *arXiv preprint arXiv:2502.14831*, 2025.

Stability AI. Sd-vae-ft-mse: Fine-tuned vae decoder for stable diffusion. `https://huggingface.co/stabilityai/sd-vae-ft-mse`, 2022. Computer Software.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712. IEEE, 2025.

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*, volume 2026, pp. 35791–35820, 2026.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

A Appendix

A.1 Supplement Figures



Figure S1: Examples of reconstructed images in Flowers102. Top row: original images, center row: reconstructed images with trained decoder, bottom row: reconstructed images by optimizing intermediate layer h (dimensionality $2.67\times$ image size). Per-image MSE above each panel. Reconstruction was performed independently for each RGB channel.

A.2 Methods

A.2.1 Architecture

The autoencoder models (both deterministic and variational) share the same architecture design that is shown below:

$$\underset{(C,32,32)}{x} \xrightarrow{\text{encoder}} \underset{(32)}{z} \xrightarrow{\text{decoder.fc}} \underset{(2048)}{h} \xrightarrow{\text{decoder.body}} \underset{(C,32,32)}{\hat{x}}$$

The encoder E contains 3 convolutional layers, GroupNorm followed by ReLU were applied after each convolutional layer. The last layer of the encoder is a flatten layer of size 2048. This layer is connected with the latent layer z.

The decoder G shares the same architecture as E, but in reverse. The first layer h in decoder is called the decoder.fc or the intermediate decoder layer with the size 2048.

$z = 32$ and $h = 2048$ for all models tested in the paper except for in Fig. 4, where the size of z and h (and the size of corresponding convolutional layers) are adjusted according to each h size and z/h ratio tested.

A.2.2 Optimization Notes

Fig. 1: a)-b): h-optimization was run for 20,000 steps, with $lr = 0.05$, 16 images were reconstructed in the test. For CIFAR-10 and Flowers102, the optimization was done for each color channel independently. c): both optimization methods were run for 3,000 steps, with $lr = 0.05$ and learning rate

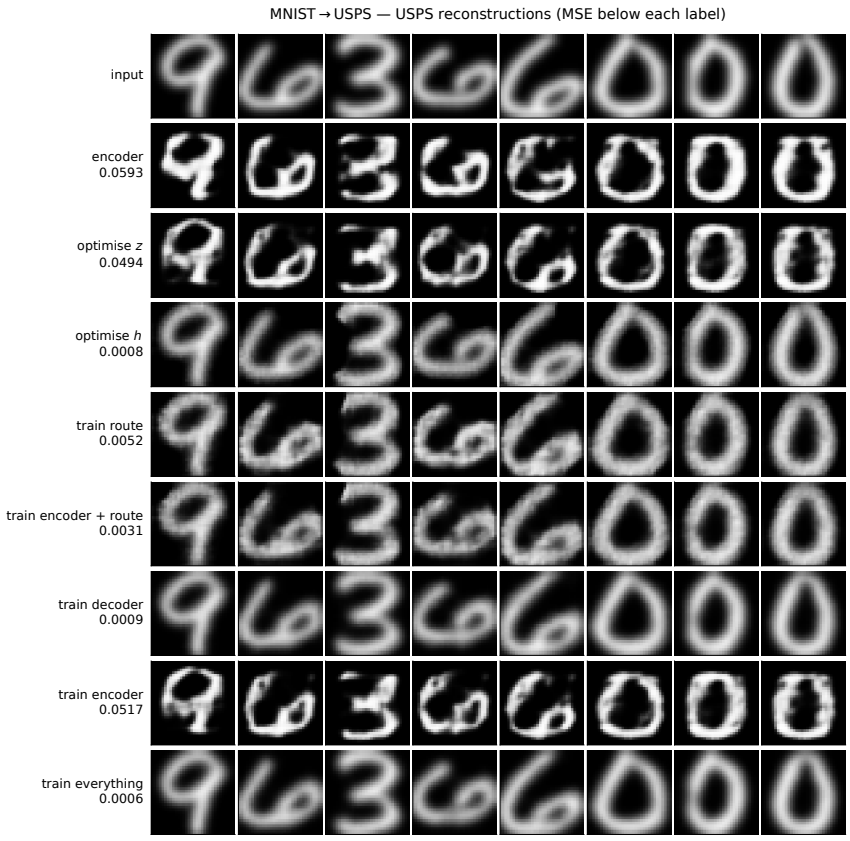


Figure S2: Reconstruction examples of USPS images after learning on MNIST. Each row is a different transfer method. Reconstruction MSE under method name.

scheduling. The optimization was performed with a warm-start, i.e., z^0 and h^0 which are output directly from the trained encoder were used as the initial condition. 32 images were reconstructed in total.

Fig. 2: During training, Adam optimization with $lr = 10^{-3}$. z-optimization was run for 300 steps with $lr = 0.05$ (and already converged, improvement $< 1\%$ with 2000 steps), and h-optimization was run for 20,000 steps, $lr = 0.2$. Only the male recording data in Speech dataset (Panayotov et al., 2015) was used in this figure.

Fig. 4: z-optimization steps $N = 20,000$ for $z = 2,048$, $N = 5,000$ for $z = 1,024$, $N = 3,000$ for the remaining sizes of z. h-optimization steps are 3,000 for all values of h. $lr = 0.05$ for z-optimization and $lr = 0.2$ for h-optimization.

Fig. 5: Similar to Fig. 2 a and b.

Different learning rates (from 0.05 to 0.2 to 0.5) and optimization steps were tested for z-optimization and h-optimization in all experiments, the best values were chosen for the respective method independently.

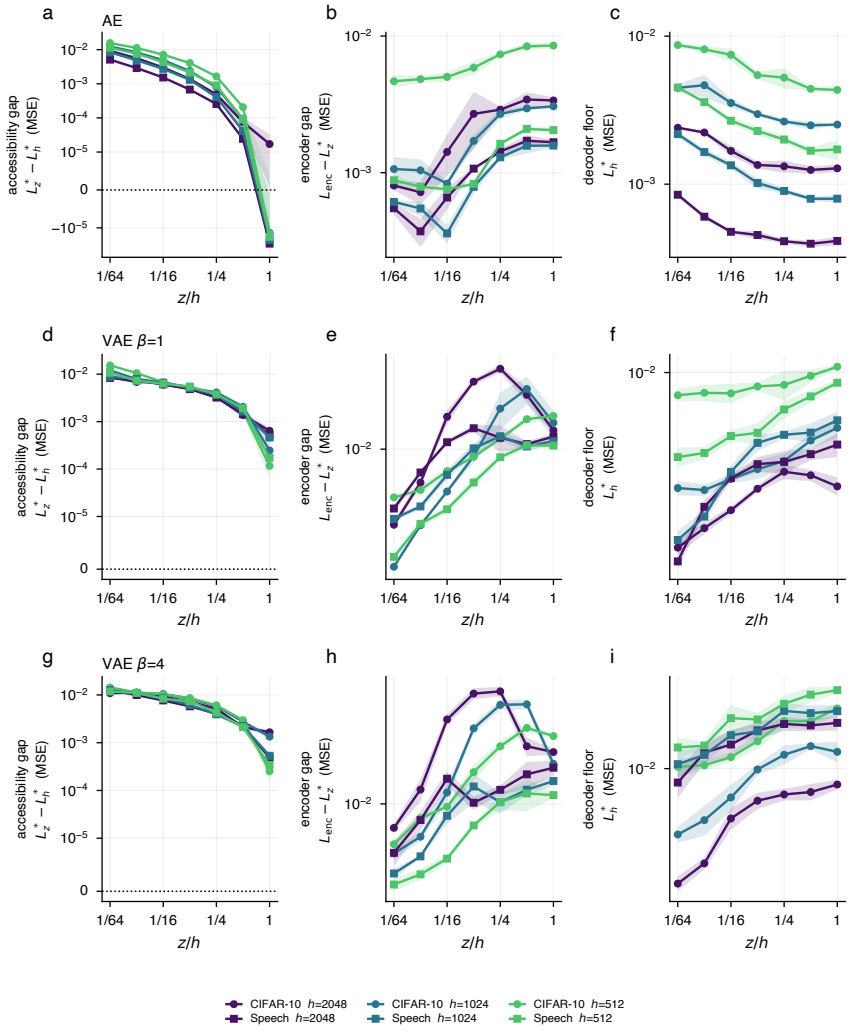


Figure S3: Unnormalized reconstruction decomposition across bottleneck widths. Absolute accessibility gap $L_z^* - L_h^*$ (a,d,g), encoder gap $L_{\text{enc}} - L_z^*$ (b,e,h), and decoder reconstruction floor L_h^* (c,f,i) as a function of z/h for deterministic AEs (a–c), VAEs with $\beta = 1$ (d–f), and VAEs with $\beta = 4$ (g–i). Results are shown for CIFAR-10 and Speech with $h = 512, 1024$, and 2048. Lines show the mean across five seeds and shaded regions indicate standard deviation.