A pipeline for interpretable neural latent discovery

Jai Bhagat
Sainsbury Wellcome Centre
University College London
jkbhagatio@gmail.com

Anaya Pouget
Sainsbury Wellcome Centre
University College London
pouget.anaya@gmail.com

Sara Molas-Medina
University College London
saramolas18@gmail.com

Abstract

Mechanistic understanding of the brain requires interpreting large-scale neuronal computations. Many latent variable models excel at decoding but yield opaque latent spaces. We address this with NLDisco, a pipeline for interpretable neural latent discovery. Motivated by sparse dictionary learning, NLDisco allows hidden layer neurons in sparse encoder-decoder models to learn interpretable representations across varying recording modalities and experimental paradigms. We validate the pipeline on synthetic and real datasets, demonstrating it recovers ground-truth features and reveals meaningful representations. We conclude with a discussion of future development and applications, and provide an open-source software package to facilitate neuroscientific discovery.

1 Introduction

Understanding the principles of neural computation requires extracting interpretable features from high-dimensional neural data [1–6]. Traditional dimensionality reduction techniques, while useful for visualization and basic analyses, often make invalid assumptions about or altogether fail to disentangle distributed representations [7,8]. Recent work has produced promising latent variable model (LVM) approaches capable of identifying low-dimensional subspaces that can accurately decode aspects of behavior and environment [9–31]. However, these methods typically value decoding accuracy over latent interpretability, and consequently have limitations such as opaque latent spaces, complex priors, supralinear scaling w.r.t. dataset size, and more (Table 1).

We address these limitations in NLDisco, which provides a highly configurable, user friendly pipeline for interpretable latent discovery from high-dimensional neural data. We consider a latent's interpretability in two key aspects: 1) its *correspondence* to a specific external variable – a "natural" behavioral or environmental feature[1]; 2) its explicit *composition* from contributing neural activity. Unlike other approaches that require a search for meaningful directions or dynamics in latent space (Figure S1), NLDisco outputs individual latents in the form of hidden layer neurons from shallow, overcomplete, sparse encoder-decoder (SED) models [32,33] that can be directly assessed for interpretability, an approach similar to those that have had many successess in related fields [34–40]. Additionally, while we showcase NLDisco on spike data, it can be readily used with virtually any other neural recording modality as the only data requirement is any predefined spatiotemporal binning.

2 Methods: The NLDisco pipeline

The NLDisco pipeline transforms high-dimensional neural data into a set of interpretable latents in four primary stages, the first three of which can be fully automated (Figure 1).

[1] A "feature" is a sufficiently interpretable latent. We illustrate this in Results.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Data on the Brain & Mind.

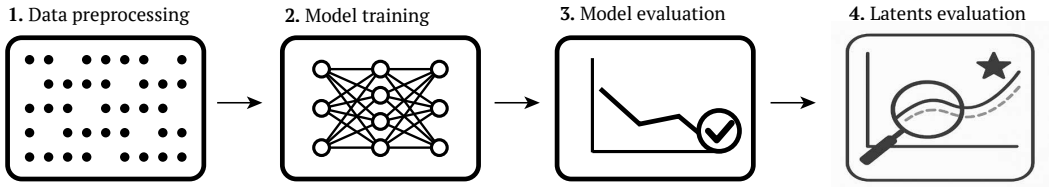

Figure 1: The NLDisco pipeline.
The NLDisco pipeline has 4 stages: 1) Spatiotemporal binning and processing of neural data; 2) Training a model; 3) Evaluating the model; 4) Evaluating the model's latents for feature interpretability.

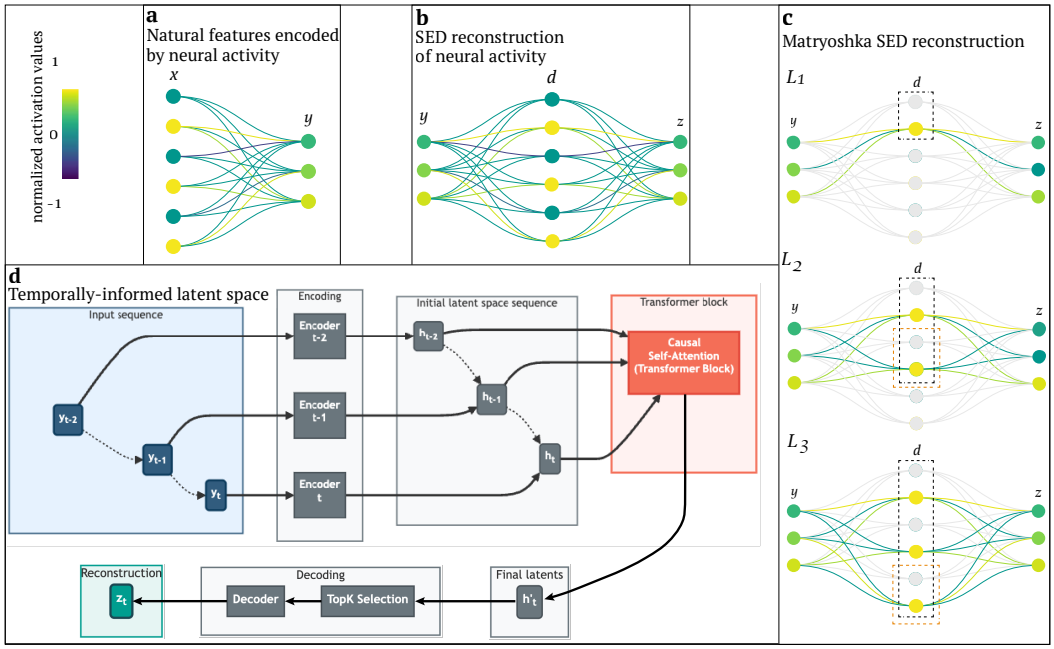

Figure 2: Model architecture considerations
(**a**) Natural, "real-world" features x are encoded by neural activity y. In this example, three active features are simultaneously represented by the joint activity of three neurons. (**b**) A SED reconstructs neural activity z based on y via sparse dictionary elements d. When training is successful, d corresponds to x: sparse dictionary elements (i.e. model neurons) represent natural features. If z tries to recreate y exactly ($\hat{y}$), the model is an autoencoder; in other scenarios (e.g. z is separate but dependent on or related to y) it is a transcoder. (**c**) A Matryoshka SED segments the latent space into multiple nested levels, each of which attempts to do a full reconstruction of the target neural activity. The black boxes indicate the latents involved in a single level, while the light-red boxes indicate the additional latents used at lower-levels. In this example, k=1 for top-k selection of latents to recruit for reconstruction at each level (the yellow neuron within each light-red box). Latents in the highest-level (L_1) will often correspond to high-level features (e.g. a round object), while latents exclusive to the lowest-level (L_3) will often correspond to low-level features (e.g. a basketball). (**d**) Incorporation of a transformer block with sequence input allows imbuing the latents with temporal information corresponding to the evolution of the sequence, which can lead to improved reconstruction and interpretability. Each sample in the input sequence is transformed by the same encoding dictionary matrix in parallel to yield a latent space sequence. Causal self-attention is then performed on this sequence of latents via a single, small multi-head transformer block, yielding a final latent space. The latents are sparsified via top-k and transformed by the decoding dictionary matrix to yield the neural data reconstruction, as in the single, non-sequential input case.

The first stage preproceses neural data for model training, including utilities for binning and normalizing the data, and can work directly with output from popular spikesorters such as Kilosort[41].

The second stage trains a novel SED architecture to reconstruct target neural data (Algorithm 1). Sparsity in the model's latent space encourages a monosemantic dictionary, where each hidden layer neuron corresponds to a single neural representation that can be judged for interpretability. The model supports both autoencoder and transcoder configurations (Figure 2a,b). We incorporate a Matryoshka architecture to learn multi-scale features[42] and batch top-k selection[43] to control sparsity dynamically (Figure 2c). Optionally, a self-attention layer can integrate temporal history (Figure 2d).

The third and fourth stages involve model and latent evaluation, respectively. Model quality is evaluated via metrics inspired by SAEBench[44], focusing on the trade-off between reconstruction fidelity and dictionary sparsity (Figure S2). Finally, latents are evaluated for interpretability, commonly by visualizing activation patterns and quantifying decoding performance for variables of interest, amongst other approaches (see Results).

3 Results

We evaluated NLDisco on simulated rat hippocampal spike data in a navigation task, and real macaque motor cortex spike data in an active reaching task, demonstrating the pipeline's ability to recover ground-truth features and reveal meaningful representations.

3.1 Simulated rat hippocampal spike data in a navigation task

Following Theodoni et al.[45], we simulated the activity of hippocampal neurons during a navigation task in a virtual linear environment (see Simulated rat spike data in a navigation task for full simulation details). The virtual rat moved along a 1-m linear track with stochastic velocity. We modeled four place cells with heterogeneous fields, and to determine if NLDisco could discover task-relevant features within larger neural population activity, added activity from 96 "noise" neurons independent of position.

After training an SED on this simulated data, we found that while some latents captured random noise fluctuations, others successfully learned spatial signals. Two classes of spatial representations emerged: single-place-cell-like features and multi-place-cell-like features (Figure 3), demonstrating NLDisco's ability to isolate distinct functional representations.

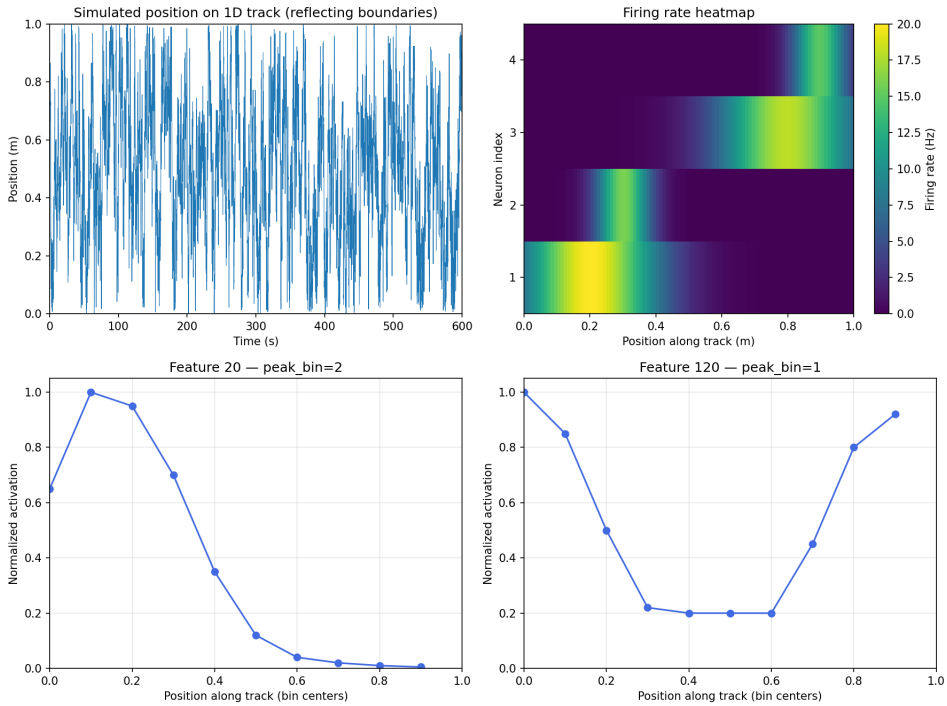

Figure 3: Features identified from simulated place cell activity
Simulated place cell activity and learned latent representations on a 1D linear track. Top-left: the simulated positional trajectory. Top-right: the firing rate tuning curves for four place cells. Bottom-left: single-place-cell-like feature firing for an early position in the track. Bottom-right: multi-place-cell-like feature firing for the beginning and end of the track.

Figure 4: Features identified in the Churchland dataset
(a) Example of SED features tuned to a continuous behavioural variable: hand velocity. The dashed line marks 0.5, corresponding to equal activation probability inside and outside a bin (no selectivity), while values above 0.5 indicate condition-specific tuning. Features shown are selective for low, high, and "extreme" velocities (active at both ends of the range). **(b)** Example of an SED feature tuned to a discrete environmental variable: the hit target being on the right side of the workspace (positive x-coordinate). Selectivity scores are computed as in (a), with the dashed line again marking the non-selective baseline at 0.5. **(c)** Mean z-scores of biological neurons when the right-side target feature (latent 169) is active, showing which units in the recorded population are systematically co-active with the latent.

3.2 Macaque motor cortex spike data in an active reaching task

To test NLDisco on real neural data, we used the churchland_shenoy_neural_2012 dataset[46] prepared by Brainsets[47], in which monkeys perform a reaching task under a variety of different conditions. Recordings contain roughly 200 neurons per monkey alongside behavioural measures like hand position, velocity, acceleration, and gaze position.

To find features, SED latents were automatically mapped to metadata variables of interest using a (Selectivity score). This approach revealed a wide range of features - for instance, latents tuned to different ranges of hand velocity (Figure 4a). Strikingly, these feature types emerged independently across subjects and generalized to unseen sessions. Additionally, we also identified features aligned with environmental variables, such as the hit target position (Figure 4b). Finally, beyond mapping latents to behavioral or task variables, NLDisco allows tracing features back to their underlying source units (Figure 4c), highlighting both correspondence and compositional interpretability.

4 Discussion

In this work we introduced NLDisco, an end-to-end pipeline for discovering interpretable neural latents. NLDisco offers several key advantages over other neural LVM approaches, including interpretability of sparse, multi-scale features, scalability to large datasets, and flexibility across different recording modalities. The pipeline is particularly well-suited for exploratory analyses of high-dimensional neural data. We acknowledge, however, several limitations: model performance can be sensitive to hyperparameters, discovered relationships are correlational rather than causal, and final assessment of a latent's interpretability is inherently subjective. Future work will focus on applying the pipeline to examine neural representations across space (brain regions) and time, as well as to other recording modalities such as calcium imaging. Additionally, we plan to leverage temporal integration via latent self-attention to enhance both reconstruction accuracy from and interpretability of dynamic, evolving features. By releasing NLDisco as an open-source package with interactive tutorials, we aim to facilitate neuroscientific discovery.

References

[1] Nicholas A. Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, Susu Chen, Jennifer Colonell, Richard J. Gardner, Bill Karsh, Fabian Kloosterman, Dimitar Kostadinov, Carolina Mora-Lopez, John O'Callaghan, Junchol Park, Jan Putzeys, Britton Sauerbrei, Rik J. J. van Daal, Abraham Z. Vollan, Shiwei Wang, Marleen Welkenhuysen, Zhiwen Ye, Joshua T. Dudman, Barundeb Dutta, Adam W. Hantman, Kenneth D. Harris, Albert K. Lee, Edvard I. Moser, John O'Keefe, Alfonso Renart, Karel Svoboda, Michael Häusser, Sebastian Haesler, Matteo Carandini, and Timothy D. Harris. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539):eabf4588, 2021. doi: 10.1126/science.abf4588. URL https://doi.org/10.1126/science.abf4588.

[2] Bogdan C. Raducanu, Refet Firat Yazicioglu, Carolina Mora López, Marco Ballini, Jan Putzeys, Shiwei Wang, Andrei Alexandru, Véronique Rochus, Marleen Welkenhuysen, Nick Van Helleputte, Silke Musa, Robert Puers, Fabian Kloosterman, Chris Van Hoof, Richárd Fiáth, István Ulbert, and Srinjoy Mitra. Time multiplexed active neural probe with 1356 parallel recording sites. *Sensors*, 17(10):2388, 2017. doi: 10.3390/s17102388. URL https://doi.org/10.3390/s17102388.

[3] Denise J Cai, Daniel Aharoni, Tristan Shuman, Justin Shobe, Jeremy Biane, Weilin Song, Brandon Wei, Michael Veshkini, Mimi La-Vu, Jerry Lou, Sergio E Flores, Isaac Kim, Yoshitake Sano, Miou Zhou, Karsten Baumgaertel, Ayal Lavi, Masakazu Kamata, Mark Tuszynski, Mark Mayford, Peyman Golshani, and Alcino J Silva. A shared neural ensemble links distinct contextual memories encoded close in time. *Nature*, 534(7605):115–118, 2016. doi: 10.1038/nature17955. URL https://pubmed.ncbi.nlm.nih.gov/27251287/.

[4] Vincent Villette, Mariya Chavarha, Ivan Dimov, Jonathan Bradley, Lagnajeet Pradhan, Benjamin Mathieu, Stephen W. Evans, Simon Chamberland, Dongqing Shi, Renzhi Yang, Benjamin B. Kim, Annick Ayon, Abdelali Jalil, François St-Pierre, Mark J. Schnitzer, Guo-Qiang Bi, Katalin Tóth, Jun B. Ding, Stéphane Dieudonné, and Michael Z. Lin. Ultrafast two-photon imaging of a high-gain voltage indicator in awake behaving mice. *Cell*, 179(7):1590–1608.e23, 2019. doi: 10.1016/j.cell.2019.11.004. URL https://doi.org/10.1016/j.cell.2019.11.004.

[5] Dimitre G Ouzounov, Tianyu Wang, Mengran Wang, Danielle D Feng, Nicholas G Horton, Jean C Cruz-Hernández, Yu-Ting Cheng, Jacob Reimer, Andreas S Tolias, Nozomi Nishimura, and Chris Xu. In vivo three-photon imaging of activity of gcamp6-labeled neurons deep in intact mouse brain. *Nature Methods*, 14(4):388–390, 2017. doi: 10.1038/nmeth.4183. URL https://doi.org/10.1038/nmeth.4183.

[6] Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, 10(5):413–420, 2013. doi: 10.1038/nmeth.2434. URL https://doi.org/10.1038/nmeth.2434.

[7] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

[8] Mark D Humphries. Strong and weak principles of neural dimension reduction. *Neurons, Behavior, Data analysis, and Theory*, 5(2), 2021. ISSN 2690-2664. doi: 10.51628/001c.24619. URL http://dx.doi.org/10.51628/001c.24619.

[9] Yue Song, T. Anderson Keller, Yisong Yue, Pietro Perona, and Max Welling. Langevin flows for modeling neural latent dynamics. *arXiv preprint arXiv:2507.11531*, 2025. URL https://arxiv.org/abs/2507.11531.

[10] Steffen Schneider, Jin H Lee, and Mackenzie W Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617(7960):360–368, 2023.

[11] Trung Le and Eli Shlizerman. Stndt: Modeling neural population activity with spatiotemporal transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17926–17939. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/72163d1c3c1726f1c29157d06e9e93c1-Paper-Conference.pdf.

[12] Mohammad R Keshtkaran and Chethan Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *Nature Methods*, 19 (12):1598–1608, 2022.

[13] Byron M Yu, John P Cunningham, Gokhan Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of population spike trains. *Journal of Neurophysiology*, 102(1):614–635, 2009.

[14] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, 2011.

[15] Yuanjun Gao, Evan Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[16] Ryan J Low, Sam Lewallen, Dmitriy Aronov, Rachel Nevers, and David W Tank. Probing variability in a cognitive map using manifold inference from neural dynamics. *bioRxiv*, page 418939, 2018.

[17] Kristopher T Jensen, Ta-Chu Kao, Marco Tripodi, and Guillaume Hennequin. Manifold gplvms for discovering non-euclidean latent structure in neural data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Daniel Hernandez, Antonio Khalil Moretti, Ziqiang Wei, Shreya Saxena, John P Cunningham, and Liam Paninski. Nonlinear evolution via spatially-dependent linear dynamics for electrophysiology and calcium data (vind). *arXiv preprint arXiv:1811.02459*, 2020. v3, 2020.

[19] Timothy Doyeon Kim, Thomas Zhihao Luo, Jonathan W Pillow, and Carlos D Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 5568–5578. PMLR, 2021.

[20] Cole Hurwitz, Akash Srivastava, Kai Xu, Justin Jude, Matthew G Perich, Lee E Miller, and Matthias H Hennig. Targeted neural dynamical modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Mathieu Schimel, Ta-Chu Kao, Kristopher T Jensen, and Guillaume Hennequin. ilqr-vae: Control-based learning of input-driven dynamics with applications to neural data. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

[22] Nina Kudryashova, Matthew G Perich, Lee E Miller, and Matthias H Hennig. Ctrl-tndm: Decoding feedback-driven movement corrections from motor cortex neurons. In *Computational and Systems Neuroscience (Cosyne) Abstracts*, 2023.

[23] Joel Ye, Jennifer L Collinger, Leila Wehbe, and Robert Gaunt. Neural data transformer 2: Multi-context pretraining for neural spiking activity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[24] Rabia Gondur, Usama Bin Sikandar, Evan Schaffer, Mikio Christian Aoi, and Stephen L Keeley. Multi-modal gaussian process variational autoencoders for neural and behavioral data. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[25] Arthur Pellegrino, Heike Stein, and N. Alex Cayco-Gajic. Dimensionality reduction beyond neural subspaces with slice tensor component analysis. *Nature Neuroscience*, 27(6):1199–1210, 2024.

[26] Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Dissociative and prioritized modeling of behaviorally relevant neural dynamics using recurrent neural networks. *Nature Neuroscience*, 27(10):2033–2045, 2024.

[27] Marten Pals, Vineeth Ramaswamy, Léo Grinsztajn, Maxime Le Helley, and Etienne Roesch. Inferring stochastic low-rank recurrent neural networks with variational sequential monte carlo. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[28] Yizhe Zhang and colleagues. Towards a "universal translator" for neural dynamics at scale. *arXiv preprint arXiv:2407.14668*, 2024. URL https://arxiv.org/abs/2407.14668.

[29] Tom M George, Pierre Glaser, Kim Stachenfeld, Caswell Barry, and Claudia Clopath. Simpl: Scalable and hassle-free optimisation of neural representations from behaviour. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[30] Sean M Perkins, Elom A Amematsro, John P Cunningham, Qi Wang, and Mark M Churchland. An emerging view of neural geometry in motor cortex supports high-performance decoding. *eLife (Reviewed Preprint, Version of Record)*, 2025. doi: 10.7554/eLife.89421. URL https://elifesciences.org/reviewed-preprints/89421.

[31] Valentin Schmutz, Ali Haydaroğlu, Shuqi Wang, Yixiao Feng, and Kenneth D Harris. High-dimensional neuronal activity from low-dimensional latent dynamics: a solvable model. *bioRxiv*, 2025. URL https://www.biorxiv.org/content/10.1101/2025.06.03.657632v1. Introduces the Neural Cross-Encoder (NCE).

[32] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997. doi: 10.1016/S0042-6989(97)00169-7. URL https://doi.org/10.1016/S0042-6989(97)00169-7.

[33] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103, Helsinki, Finland, 2008. ACM. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

[34] Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

[35] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv*, 2023. doi: 10.48550/arXiv.2309.08600. URL https://arxiv.org/abs/2309.08600.

[36] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L. Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html#appendix-feature-ablations.

[37] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

[38] Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *arXiv*, 2024. doi: 10.48550/arXiv.2406.11944. URL https://arxiv.org/abs/2406.11944.

[39] Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

[40] Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

[41] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv*, 2016. doi: 10.1101/061481. URL https://www.biorxiv.org/content/early/2016/06/30/061481.

[42] Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv*, 2025. URL https://arxiv.org/abs/2503.17547.

[43] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv*, 2024. URL https://arxiv.org/abs/2412.06410.

[44] Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *arXiv*, 2025. URL https://arxiv.org/abs/2503.09532.

[45] Panagiota Theodoni, Bernat Rovira, Yingxue Wang, and Alex Roxin. Theta-modulation drives the emergence of connectivity patterns underlying replay in a network model of place cells. *eLife*, 7:e37388, 2018. doi: 10.7554/eLife.37388. URL https://doi.org/10.7554/eLife.37388.

[46] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[47] Brainsets Project. churchland_shenoy_neural_2012 dataset documentation. https://brainsets.readthedocs.io/en/latest/glossary/brainsets.html#churchland-shenoy-neural-2012, 2022. Accessed: 2025-09-23.

[48] Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019. doi: 10.21105/joss.00747. URL https://doi.org/10.21105/joss.00747.

[49] D. Campagner, J. Bhagat, G. Lopes, L. Calcaterra, A. G. Pouget, A. Almeida, T. T. Nguyen, C. H. Lo, T. Ryan, B. Cruz, F. J. Carvalho, Z. Li, A. Erskine, J. Rapela, O. Folsz, M. Marin, J. Ahn, S. Nierwetberg, S. C. Lenzi, J. D. S. Reggiani, and SGEN group – SWC GCNU Experimental Neuroethology Group. Aeon: an open-source platform to study the neural basis of ethological behaviours over naturalistic timescales. *bioRxiv*, 2025. doi: 10.1101/2025.07.31.664513. URL https://www.biorxiv.org/content/early/2025/08/01/2025.07.31.664513.

[50] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[51] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[52] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(May):1457–1469, 2004.

[53] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994. doi: 10.1016/0165-1684(94)90029-9. URL https://doi.org/10.1016/0165-1684(94)90029-9.

[54] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Acknowledgements

We thank the Bogue Fellowship, University College London, the Sainsbury Wellcome Centre and the Gatsby Charitable Foundation for the funding that made this project possible. Additionally, we thank Anthropic for hosting and collaboration with the Bogue Fellowship. JB was supported by a Bogue Fellowship during the initial stages of this work, and a studentship from the Gatsby Charitable Foundation and the Wellcome Trust thereafter. AP was supported by funding from the Gatsby Charitable Foundation and the Wellcome Trust.

We thank Jack Lindsey for initial guidance and mentorship during the development of the NLDisco pipeline. We thank Trenton Bricken for facilitating the collaboration with Jack and Anthropic. We thank Tiago Branco for pragmatic feedback on applying NLDisco to neural datasets, and support with the Bogue Fellowship application. We thank Aaditya Singh, Kevin Miller, Laurence Freeman, Will Dorrell, Jin Lee, Pierre Glaser, Tom George, Tom Mrsic-Flogel, Jeff Erlich, and Ann Duan for helpful conversations.

5 Appendix

5.1 Software and data availability

The code repository containing the NLDisco implementation alongside tutorial notebooks and data access instructions to replicate the results in this paper is available at `https://github.com/jkbhagatio/nldisco`.

The software is available under the permissive MIT license for general use.

5.2 On interpretable latents

Many neural LVM approaches focus on learning structure in a low-dimensional latent space. However, this structure can be inherently complex and difficult to interpret. NLDisco bypasses this challenge by learning individual latents directly.

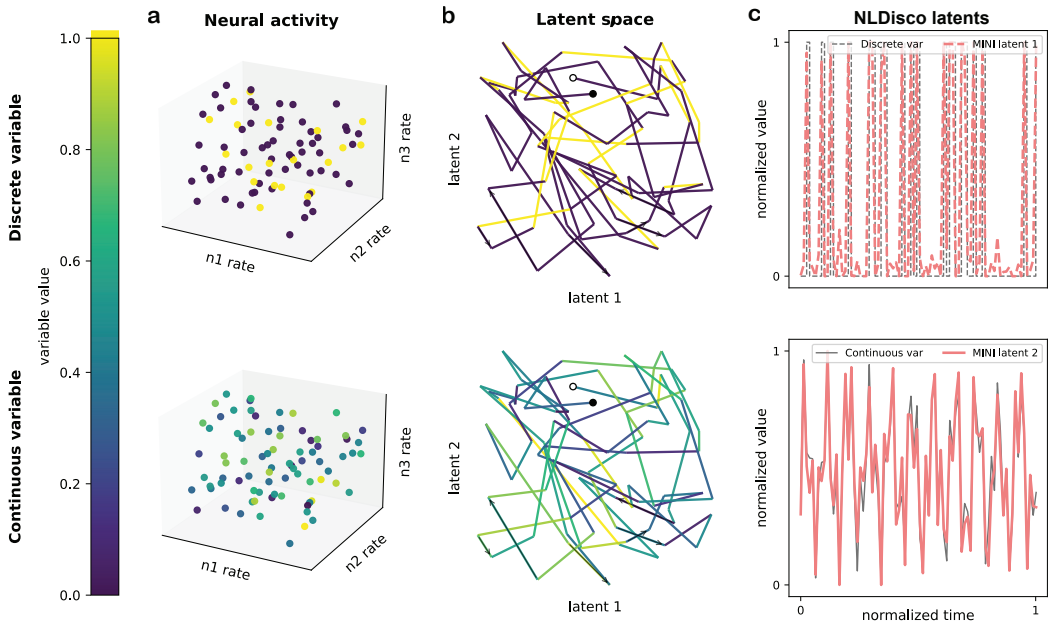

Figure S1: Interpretable latents in a complex latent space.
This toy example highlights the utility of NLDisco. The two rows show two different variables (top: discrete; bottom: continuous), each uniquely encoded by the same underlying neural activity made up of three neurons' firing rates. The viridis colorbar shows the variables' values as a function of this neural activity. (**a**) Each point in the scatterplots represents a moment in time. (**b**) A projection of this activity into a 2D latent space creates tangled trajectories where variable states (e.g. 'on' and 'off' in the discrete case, and 'high' and 'low' in the continuous case) are not easily distnguished. The start and end points of the trajectories are marked by white and black dots, respectively, while arrows indicate trajectory direction. (**c**) In contrast to the tangled latent space, NLDisco finds individual latents corresponding to each variable, demonstrating the potential for improved interpretability of neural representations.

5.3 Additional pipeline details

5.3.1 Model training

Some notes on Model training procedure:

Dead Latent Resurrection. A common failure mode in training SEDs is "latent death," where dictionary elements cease to activate for any input. We address this with an auxiliary loss designed to revive dead latents. We monitor feature activation frequencies and identify a set of dead latents $\mathcal{D}$. These latents are then trained via an auxiliary MSE loss to reconstruct the residual error of the primary model ($\mathbf{x}_{\text{target}} - \hat{\mathbf{x}}_L$). The gradients from this auxiliary loss are exclusively applied to the

Algorithm 1 Model training procedure

Data definitions:
  Input neural data: $\mathbf{Y} \in \mathbb{R}^{B \times S \times N_{\text{in}}}$ (Batch, Sequence length, Input neural units)
  Target neural data: $\mathbf{Z} \in \mathbb{R}^{B \times N_{\text{out}}}$ (Batch, Output neural units)
Model definitions:
  Encoder: $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{D_{\max} \times S \times N_{\text{in}}}$, $\mathbf{b}_{\text{enc}} \in \mathbb{R}^{D_{\max}}$ (Hidden layer neurons)
  Decoder: $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{N_{\text{out}} \times D_{\max}}$, $\mathbf{b}_{\text{dec}} \in \mathbb{R}^{N_{\text{out}}}$
  Transformer block (optional) $\theta_{\mathbf{t}}$: $\{\mathbf{W}_{Q,K,V} \in \mathbb{R}^{D_{\max} \times D_{\max}}, \dots\}$
  Matryoshka levels: $\{\mathbf{D_l}\}_{\mathbf{l=1}}^{\mathbf{L}}$
  Weights each level's reconstruction loss (optional): $\{\lambda_{\mathbf{l}}\}_{\mathbf{l=1}}^{\mathbf{L}}$
  Auxiliary loss weight: γ

procedure TRAIN_STEP($\mathbf{Y}, \mathbf{Z}$)

▷ — *Forward Pass* —
  $\mathbf{A} \leftarrow \text{ReLU}(\mathbf{Y}\mathbf{W}_{\text{enc}}^T + \mathbf{b}_{\text{enc}})$ ▷ Get encoder activations

  if $S > 1$ **then**
    $\mathbf{A} \leftarrow \text{SelfAttention}(\mathbf{A}; \theta_{\text{attn}})$ ▷ Temporally integrate over latent space
  end if

  $\mathcal{L}_{\text{recon}} \leftarrow 0$
  for $l = 1$ to L **do** ▷ For each level...
    $\hat{\mathbf{A}}_l \leftarrow S_{\text{topk}}(\mathbf{A}_{:,:D_l})$ ▷ Sparsify latents with batch top-k
    $\hat{\mathbf{Z}}_l \leftarrow \text{ReLU}(\hat{\mathbf{A}}_l \mathbf{W}_{\text{dec},:,:D_l}^T + \mathbf{b}_{\text{dec}})$ ▷ Reconstruct target (decoder activations)
    $\mathcal{L}_{\text{recon}} \leftarrow \mathcal{L}_{\text{recon}} + \lambda_l \cdot \text{MSLE}(\mathbf{Z}, \hat{\mathbf{Z}}_l)$ ▷ Get reconstruction loss
  end for

▷ — *Auxiliary Loss for Dead Latent Resurrection* —
  $\mathcal{D} \leftarrow \text{GetDeadLatents}(\mathbf{A})$
  if any($\mathcal{D}$) **then** ▷ Only compute if dead latents exist
    $\mathbf{R} \leftarrow \mathbf{Z} - \hat{\mathbf{Z}}_L$ ▷ Get reconstruction residual
    $\hat{\mathbf{R}} \leftarrow \text{ReLU}((\mathbf{A} \odot \mathcal{D})\mathbf{W}_{\text{dec}}^T + \mathbf{b}_{\text{dec}})$ ▷ Reconstruct residual from dead latents
    $\mathcal{L}_{\text{aux}} \leftarrow \text{MSE}(\mathbf{R}, \hat{\mathbf{R}})$ ▷ Get auxiliary loss
  else
    $\mathcal{L}_{\text{aux}} \leftarrow 0$
  end if

  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{recon}} + \gamma \mathcal{L}_{\text{aux}}$ ▷ Total loss: reconstruction + auxiliary

▷ — *Backward Pass & Parameter Update* —
  $\mathbf{g} \leftarrow \nabla_\theta \mathcal{L}_{\text{total}}$ ▷ Compute gradients, masking $\nabla \mathcal{L}_{\text{aux}}$ to dead latents
  $\theta \leftarrow \text{Update}(\theta, \mathbf{g})$ ▷ Update model parameters
end procedure

parameters of the dead latents, thereby encouraging them to learn useful representations without disrupting the training of active latents.

5.3.2 Model architecture

The core of our pipeline is an overcomplete sparse encoder-decoder model with a single hidden layer with ReLU activations. While we experimented with more complex architectures, such as multi-layer decoders, we found they often made interpreting the resulting latents more difficult without significant reconstruction gains. The Matryoshka architecture is particularly useful for datasets where features are expected to exist at multiple scales, as it helps mitigate "feature absorption" where general features are partially subsumed by more specific ones[42]. The transformer block is most beneficial when analyzing data with meaningful temporal dynamics, as it allows the model to learn features that evolve over a sequence rather than just instantaneous neural patterns. We found that combining these

components with a batch top-k operator provided a robust and flexible framework that consistently outperformed L1-based sparsity penalties and yielded more interpretable features.

5.3.3 Model evaluation

Model evaluation centers on the trade-off between reconstruction fidelity and latent dictionary sparsity. High reconstruction fidelity is trivial to achieve with a large and dense code but undermines the goal of finding disentangled, interpretable latents. NLDisco therefore first reviews dictionary quality via latent sparsity metrics (Figure S2 top). The mean L0 norm measures the average number of active latents per sample, set indirectly by the batch top-k parameter, while the latent activity density visualizes the firing distribution across all latents. Together these metrics reveal whether the model has learned an efficient code that uses many latents intermittently while avoiding common pitfalls such as "representational collapse" into too many constantly active latents and/or widespread "feature death", in which a large portion of the dictionary never activates. These sparsity metrics are complemented by a couple of reconstruction fidelity metrics: variance explained (R^2) and cosine similarity of reconstruction-to-target neural activity across both spatial and temporal dimensions (Figure S2 bottom). High scores in these metrics confirm that the sparse code is a sufficient representation that has captured salient information present in the target neural data.

For a deeper diagnosis, several additional optional metrics are available. To check whether explanatory power is well-distributed, a variance attribution analysis quantifies the overall reconstruction variance explained by each individual latent. And to check if temporal information present in the target neural data is preserved by the latent reconstruction, a spectral frequency analysis compares the reconstruction's frequency content to that of the target neural data, given a specified duration.

5.3.4 Latents evaluation

In this paper, we used prepared dashboards to visualize the activity of latents in the context of the simulated spatial navigation task and the in vivo motor control reaching task. In general when working with new datasets, we recommend spending time designing similar simple, bespoke dashboards.

5.3.5 Hyperparameter sweeps

Effective hyperparameter tuning is crucial for balancing reconstruction fidelity with latent interpretability. We typically sweep over key architectural and optimizer parameters. Architectural choices center on the Matryoshka configuration: `dsed_topk_map` controls the sparsity at each hierarchical level, while `dsed_loss_x_map` sets the relative weight of each level's reconstruction in the total loss. Tuning these helps discover a multi-scale dictionary that matches the complexity of the neural data. We also tune `dead_latent_window`, which determines when an inactive latent is revived via an auxiliary residual reconstruction loss. This prevents "feature death" by giving silent latents a chance to learn useful representations. We also sweep standard optimizer parameters like learning and decay rates.

While optimal settings are data-dependent, we offer some general heuristics. The total number of latents can be initially set relative to the number of training examples (e.g.$n\_examples/10$) and adjusted downwards based on reconstruction performance. The top-k value for a given level should reflect a reasonable estimate of how many features might be co-active in a given time window. Based on our experiments, we find that a cosine learning rate schedule and the MSLE loss function often outperform static learning rates and MSE loss, respectively, and can be used as robust defaults. These targeted sweeps and informed defaults streamline the search for a model that yields a sparse, accurate, and interpretable set of latents.

5.4 Additional results

5.4.1 Simulated rat spike data in a navigation task

The virtual rat moved along a 1-m linear track, with its velocity sampled from an Ornstein-Uhlenbeck (OU) process:

$$dv_t = \theta\,(\mu - v_t)\,dt + \sigma\,dW_t$$

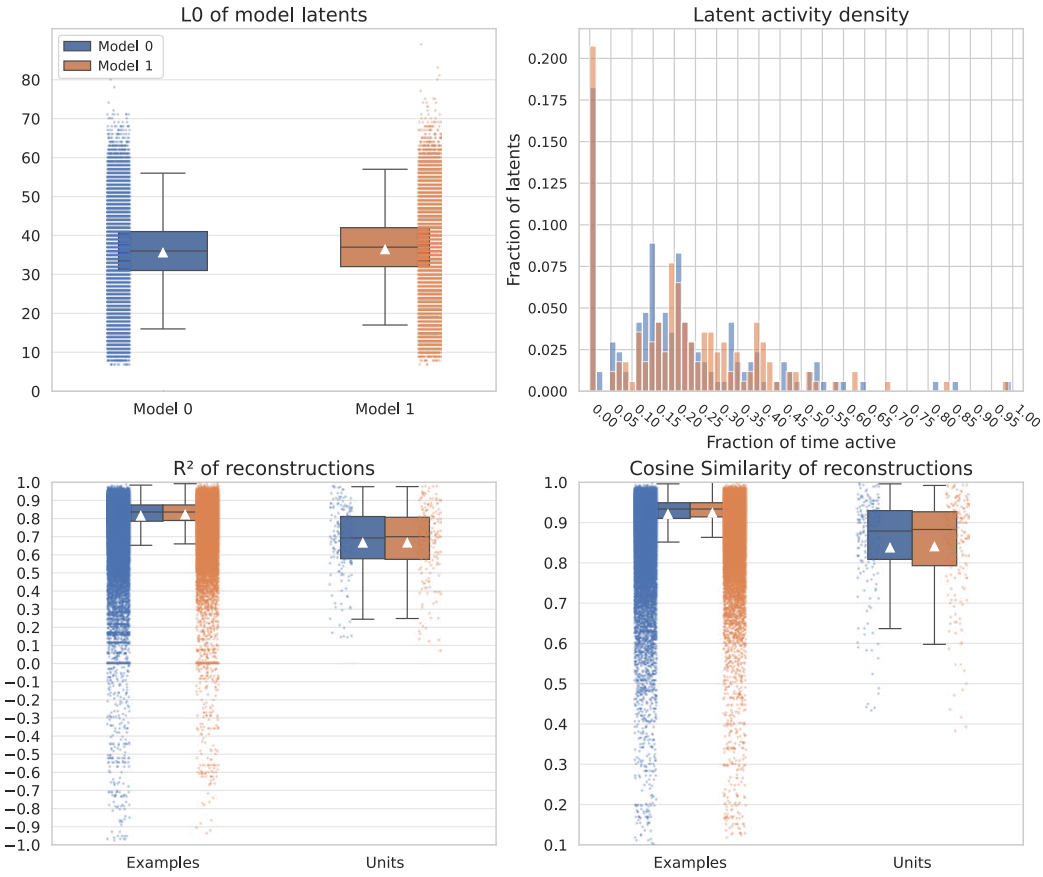

Figure S2: Model evaluation metrics.
In this example SAE model evaluation, a three-level Matryoshka architecture with a total of 320 latents was used, with top-k selections of 12 $\cdot B$, 24 $\cdot B$, and 36 $\cdot B$ active latents per batch size B (1024). The recording had a total of 200 units, and the model was trained on binned 182,868 50 ms binned spike counts. The model exhibits an approximately normal distribution for the mean latent L0 norm, few dead or tonically active latents, and high neural reconstruction fidelity as measured by the R^2 and cosine similarity metrics across both units and examples, all indicators of healthy training.

with parameters $\theta = 1.0$ (mean reversion), $\mu = 0.0$ m/s (long-run mean), $\sigma = 0.4$ (volatility), and $v_0 = 0.0$ (initial velocity). The simulation lasted 10 minutes (600 s) at a temporal resolution of 100 Hz.

Place fields were modeled as Gaussian functions of position. Two cells had broad fields ($\sigma = 20$ cm) centered at 200 cm and 800 cm, while two had narrow fields ($\sigma = 10$ cm) at 300 cm and 900 cm. Peak firing rates were set to 20, 16, 18, and 15 Hz respectively, with a 0.1 Hz baseline.

The 96 noise neurons had stable mean firing rates sampled uniformly between 1 and 5 Hz. The resulting spike count matrix was used to train an SED with 128 latents and a sparsity constraint of 4 active latents per timebin.

5.4.2 Macaque motor cortex spike data in a reaching task

5.5 Selectivity score

Units are mapped to metadata variables through the calculation of a selectivity score. For a latent l and condition c (a variable/value combination, e.g. velocity between 0 and 1):

$$\text{activation\_frac\_during} = \frac{\#\{\text{activations of } l \text{ in examples with } c\}}{\#\{\text{examples with } c\}} \tag{1}$$

$$\text{activation\_frac\_outside} = \frac{\#\{\text{activations of } u \text{ in examples without } c\}}{\#\{\text{examples without } c\}} \tag{2}$$

$$\text{selectivity\_score} = \frac{\text{activation\_frac\_during}}{\text{activation\_frac\_during} + \text{activation\_frac\_outside}} \tag{3}$$

A selectivity score of 0.5 corresponds to no preference, meaning the unit is equally active inside and outside the condition. Scores above 0.5 indicate condition-specific tuning, with larger values reflecting stronger selectivity. Scores below 0.5 indicate the unit is more active outside the condition.

5.6 Methods Comparison

Here we highlight 15 features of neural LVM methods and create a table displaying how NLDisco and other relevant methods compare against these features.

These features are:

- *Requires multimodal data*: Whether the method requires multimodal data (e.g. video data, or various forms of behavioral data, in addition to neural data).
- *Requires trial-structured data*: Whether the method requires trial-structured data.
- *Supports multimodal data*: Whether the method can incorporate multimodal data, even if not required.
- *Supports trial-structured data*: Whether the method can use trial-structured data, even if not required.
- *Learns sparse, easily identifiable latents*: Whether the method by default learns sparse, easily identifiable latents.
- *Learns hierarchical latents*: Whether the method by default learns latents that are hierarchically organized (e.g. whether the method can learn one latent that corresponds to a particular behavior, and another, sparser latent that corresponds to a sub-behavior of the first)
- *Learns temporally precise latents*: Whether the method by default learns latents that are time-locked to neural events at the resolution of the desired event, potentially down to single-spike precision.
- *Learns a continuously-valued latent space*: Whether the method by default learns latents that change smoothly as a function of changes in the input neural data.
- *Can use temporal dynamics to update the latent space*: Whether the method can use temporal dynamics (e.g. neural data or latent space history) to update the latent space.
- *Imposes a prior on the latent space*: Whether the method imposes a predfined structure on the latent space (e.g. geometric constraints like orthogonality of latents, or distributional assumptions like a Gaussian latents).
- *Uses nonlinear dynamics*: Whether the method learns latents from nonlinear neural dynamics.
- *Can be used as a generative model*: Whether the method can be used to generate new neural data samples from the learned latent space.
- *Enforces neural data reconstruction*: Whether the method needs to perform neural data reconstruction when learning latents.
- *Has approximate linear time scaling*: Whether the method has linear time scaling with respect to the number of data points in an example and in the dataset.
- *Requires significant hyperparameter tuning*: Whether the method requires significant hyperparameter tuning to learn interpretable latents.

Light-green text indicates that the method has an ideal implementation of the feature, while dark-red text indicates a shortcoming.

Feature	**NLDisco**	LangevinFlow[9]	CEBRA[10]	ST-NDT[11]	AutoLFADS[12]	UMAP[50]	t-SNE[51]	sparseNMF[52]	ICA[53]	PCA[54]
Requires multimodal data	No	No	No	No	No	No	No	No	No	No
Requires trial-structured data	No	No	No	No	No^	No	No	No	No	No
Supports multimodal data	No†	No	Yes	No	No	No	No	No	No	No
Supports trial-structured data	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Learns sparse, easily identifiable latents	Yes	No	No	No	No	No	No	Yes	No	No
Learns hierarchical latents	Yes	No	No	No	No	No	No	No	No	No
Learns temporally precise latents	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	No
Learns a continuously-valued latent space	No‡	Yes	Yes	Yes	Yes	Yes	Yes	No‡	Yes	Yes
Can use temporal dynamics to update the latent space	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
Imposes a prior on the latent space	No	Yes	No	No	Yes	No	No	No	Yes	Yes
Uses nonlinear dynamics	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
Can be used as a generative model	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
Enforces neural data reconstruction	Yes	Yes	No	Yes	Yes	No	No	Yes	No	No
Has approximate linear time scaling	Yes#	No	Yes	No	No	Yes	No	No	Yes	Yes
Requires significant hyperparameter tuning	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No

Table 1: Comparison of NLDisco with other neural LVM methods.

†: Not a fundamental limitation – could be added as a feature
‡: Enforced sparsity can cause step-like jumps in the data-to-latents mapping
#: In the standard implementation, without a transformer layer in the decoder
^: Can bin data to create "pseudo"-trials