Article

Learnable latent embeddings for joint behavioural and neural analysis

https://doi.org/10.1038/s41586-023-06031-6

Received: 30 March 2022

Accepted: 28 March 2023

Published online: 3 May 2023

Open access

Check for updates

Steffen Schneider[1,2], Jin Hwa Lee[1,2] & Mackenzie Weygandt Mathis[1]✉

Mapping behavioural actions to neural activity is a fundamental goal of neuroscience. As our ability to record large neural and behavioural data increases, there is growing interest in modelling neural dynamics during adaptive behaviours to probe neural representations[1–3]. In particular, although neural latent embeddings can reveal underlying correlates of behaviour, we lack nonlinear techniques that can explicitly and flexibly leverage joint behaviour and neural data to uncover neural dynamics[3–5]. Here, we fill this gap with a new encoding method, CEBRA, that jointly uses behavioural and neural data in a (supervised) hypothesis- or (self-supervised) discovery-driven manner to produce both consistent and high-performance latent spaces. We show that consistency can be used as a metric for uncovering meaningful differences, and the inferred latents can be used for decoding. We validate its accuracy and demonstrate our tool's utility for both calcium and electrophysiology datasets, across sensory and motor tasks and in simple or complex behaviours across species. It allows leverage of single- and multi-session datasets for hypothesis testing or can be used label free. Lastly, we show that CEBRA can be used for the mapping of space, uncovering complex kinematic features, for the production of consistent latent spaces across two-photon and Neuropixels data, and can provide rapid, high-accuracy decoding of natural videos from visual cortex.

A central quest in neuroscience is the neural origin of behaviour[1,2]. Nevertheless, we are still limited in both the number of neurons and length of time we can record from behaving animals in a session. Therefore, we need new methods that can combine data across animals and sessions with minimal assumptions, thereby generating interpretable neural embedding spaces[1,3]. Current tools for representation learning are either linear or, if nonlinear, typically rely on generative models and they do not yield consistent embeddings across animals (or repeated runs of the algorithm). Here, we combine recent advances in nonlinear disentangled representation learning and self-supervised learning to develop a new dimensionality reduction method that can be applied jointly to behavioural and neural recordings to show meaningful lower-dimensional neural population dynamics[3–5].

From data visualization (clustering) to discovery of latent spaces that explain neural variance, dimensionality reduction of behaviour or neural data has been impactful in neuroscience. For example, complex three-dimensional (3D) forelimb reaching can be reduced to between only eight and twelve dimensions[6,7], and low-dimensional embeddings show some robust aspects of movements (for example, principal component analysis (PCA)-based manifolds in which the neural state space can easily be constrained and is stable across time[8–10]). Linear methods such as PCA are often used to increase interpretability, but this comes at the cost of performance[1]. Uniform manifold approximation and projection (UMAP)[11] and *t*-distributed stochastic neighbour embedding (*t*-SNE)[12] are excellent nonlinear methods but they lack the ability to explicitly use time information, which is always available in neural recordings, and they are not as directly interpretable as PCA. Nonlinear methods are desirable for use in high-performance decoding but often lack identifiability—the desirable property that true model parameters can be determined, up to a known indeterminacy[13,14]. This is critical because it ensures that the learned representations are uniquely determined and thus facilitates consistency across animals and/or sessions.

There is recent evidence that label-guided variational auto-encoders (VAEs) could improve interpretability[5,15,16]. Namely, by using behavioural variables, such algorithms can learn to project future behaviour onto past neural activity[15], or explicitly to use label priors to shape the embedding[5]. However, these methods still have restrictive explicit assumptions on the underlying statistics of the data and they do not guarantee consistent neural embeddings across animals[5,17,18], which limits both their generalizability and interpretability (and thereby affects accurate decoding across animals).

We address these open challenges with CEBRA, a new self-supervised learning algorithm for obtaining interpretable, consistent embeddings of high-dimensional recordings using auxiliary variables. Our method combines ideas from nonlinear independent component analysis (ICA) with contrastive learning[14,19–21], a powerful self-supervised learning scheme, to generate latent embeddings conditioned on behaviour (auxiliary variables) and/or time. CEBRA uses a new data-sampling scheme to train a neural network encoder with a contrastive optimization objective to shape the embedding space. It can also generate embeddings across multiple subjects and cope with distribution shifts among experimental sessions, subjects and recording modalities.

[1]Brain Mind Institute & Neuro X Institute, École Polytechnique Fédérale de Lausanne, Geneva, Switzerland. [2]These authors contributed equally: Steffen Schneider, Jin Hwa Lee.
✉e-mail: mackenzie@post.harvard.edu

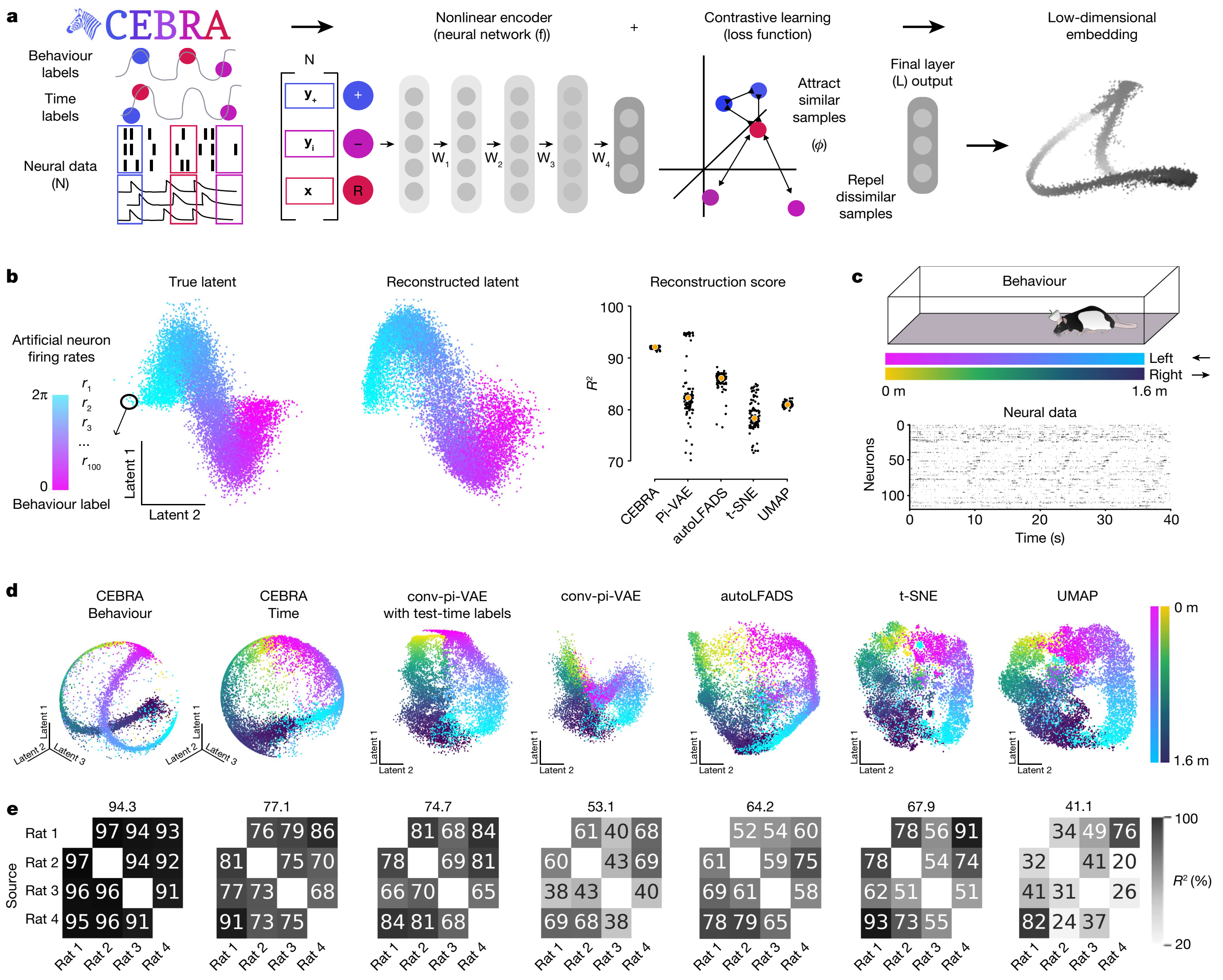


Fig. 1 | Use of CEBRA for consistent and interpretable embeddings. **a**, CEBRA allows for self-supervised, supervised and hybrid approaches for both hypothesis-and discovery-driven analysis. Overview of pipeline: collect data (for example, pairs of behaviour (or time) and neural data (**x**,**y**)), determine positive and negative pairs, train CEBRA and produce embeddings. $W_{1,...4}$ represent the neural network weights. **b**, Left, true 2D latent, where each point is mapped to the spiking rate of 100 neurons. Middle, CEBRA embedding after linear regression to the true latent. Right, reconstruction score is R^2 of linear regression between the true latent and resulting embedding from each method. The 'behaviour label' is a 1D random variable sampled from uniform distribution of [0, 2π] that is assigned to each time bin of synthetic neural data, as visualized by the colour map. The orange line represents the median and each black dot an individual run (n = 100). CEBRA-Behaviour shows a significantly higher reconstruction score compared with pi-VAE, *t*-SNE and UMAP (one-way ANOVA, $F(4, 495) = 251$, $P = 1.12 \times 10^{-117}$ with post hoc Tukey's honest significant difference $P < 0.001$). **c**, Rat hippocampus data derived from ref. 26. Cartoon from scidraw.io. Electrophysiology data were collected while a rat traversed a 1.6 m linear track 'leftwards' or 'rightwards'. **d**, We benchmarked CEBRA against conv-pi-VAE (both with labels and without), autoLFADS, *t*-SNE and unsupervised UMAP. Note: for performance against the original pi-VAE see Extended Data Fig. 1. We plot the three latents (all CEBRA-embedding figures show the first three latents). The dimensionality of the latent space is set to the minimum and equivalent dimension per method (3D for CEBRA and 2D for others) for fair comparison. Note: higher dimensions for CEBRA can yield higher consistency values (Extended Data Fig. 7). **e**, Correlation matrices show R^2 values after fitting a linear model between behaviour-aligned embeddings of pairs of rats, one as the target and the other as the source (mean, n = 10 runs). Parameters were picked by optimization of average run consistency across rats.

Importantly, our method relies on neither data augmentation (as does SimCLR[22]) nor a specific generative model, which would limit its range of use.

Joint behavioural and neural embeddings

We propose a framework for jointly trained latent embeddings. CEBRA leverages user-defined labels (supervised, hypothesis-driven) or time-only labels (self-supervised, discovery-driven; Fig. 1a and Supplementary Note 1) to obtain consistent embeddings of neural activity that can be used for both visualization of data and downstream tasks such as decoding. Specifically, it is an instantiation of nonlinear ICA based on contrastive learning[14]. Contrastive learning is a technique that leverages contrasting samples (positive and negative) against each other to find attributes in common and those that separate them. We can use discrete and continuous variables and/or time to shape the distribution of positive and negative pairs, and then use a nonlinear encoder (here, a convolutional neural network but can be another type of model) trained with a new contrastive learning objective. The encoder features form a low-dimensional embedding of the data (Fig. 1a). Generation of consistent embeddings is highly desirable and closely linked to identifiability in nonlinear ICA[14,23]. Theoretical work has shown that the use of contrastive learning with auxiliary variables is identifiable for bijective neural networks using a noise contrastive estimation (NCE) loss[14], and that with an InfoNCE loss this bijectivity assumption can sometimes be removed[24] (see also our theoretical generalization in Supplementary Note 2). InfoNCE minimization can be viewed as a classification problem such that, given a reference sample, the correct positive sample needs to be distinguished from multiple negative samples.

CEBRA optimizes neural networks $\mathbf{f}$, $\mathbf{f}'$ that map neural activity to an embedding space of a defined dimension (Fig. 1a). Pairs of data $(\mathbf{x}, \mathbf{y})$ are mapped to this embedding space and then compared with a similarity measure $\phi(\cdot,\cdot)$. Abbreviating this process with $\psi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}'(\mathbf{y}))/\tau$ and a temperature hyperparameter, τ, the full criterion for optimization is

$$\underset{\substack{\mathbf{x}\sim p(\mathbf{x}),\, \mathbf{y}_{+}\sim p(\mathbf{y}|\mathbf{x})\\ \mathbf{y}_1,\ldots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}}{\mathbb{E}}\left[-\psi(\mathbf{x}, \mathbf{y}_{+}) + \log \sum_{i=1}^{n} e^{\psi(\mathbf{x}, \mathbf{y}_i)}\right],$$

which, depending on the dataset size, can be optimized with algorithms for either batch or stochastic gradient descent.

In contrast to other contrastive learning algorithms, the positive-pair distribution p and negative-pair distribution q can be systematically designed and allow the use of time, behaviour and other auxiliary information to shape the geometry of the embedding space. If only discrete labels are used, this training scheme is conceptually similar to supervised contrastive learning[21].

CEBRA can leverage continuous behavioural (kinematics, actions) as well as other discrete variables (trial ID, rewards, brain-area ID and so on). Additionally, user-defined information about desired invariances in the embedding is used (across animals, sessions and so on), allowing for flexibility in data analysis. We group this information into task-irrelevant and -relevant variables, and these can be leveraged in different contexts. For example, to investigate trial-to-trial variability or learning across trials, information such as a trial ID would be considered a task-relevant variable. On the contrary, if we aim to build a robust brain machine interface that should be invariant to such short-term changes, we would include trial information as a task-irrelevant variable and obtain an embedding space that no longer carries this information. Crucially, this allows inference of latent embeddings without explicit modelling of the data-generating process (as done in pi-VAE[5] and latent factor analysis via dynamical systems (LFADS)[17]). Omitting the generative model and replacing it by a contrastive learning algorithm facilitates broader applicability without modifications.

Robust and decodable latent embeddings

We first demonstrate that CEBRA significantly outperforms t-SNE, UMAP, automatic LFADS (autoLFADS)[25] and pi-VAE (the latter was shown to outperform PCA, LFADS, demixed PCA and PfLDS (Poisson feed-forward neural network linear dynamical system) on some tasks) in the reconstruction of ground truth synthetic data (one-way analysis of variance (ANOVA), $F(4, 495) = 251$, $P = 1.12 \times 10^{-117}$; Fig. 1b and Extended Data Fig. 1a,b).

We then turned to a hippocampus dataset that was used to benchmark neural embedding algorithms[5,26] (Extended Data Fig. 1c and Supplementary Note 1). Of note, we first significantly improved pi-VAE by the addition of a convolutional neural network (conv-pi-VAE), thereby allowing this model to leverage multiple time steps, and used this for further benchmarking (Extended Data Fig. 1d,e). To test our methods, we first considered the correlation of the resulting embedding space across subjects (does it produce similar latent spaces?), and the correlation across repeated runs of the algorithm (how consistent are the results?). We found that CEBRA significantly outperformed other algorithms in the production of consistent embeddings, and it produced visually informative embeddings (Fig. 1c–e and Extended Data Figs. 2 and 3; for each embedding a single point represents the neural population activity over a specified time bin).

When using CEBRA-Behaviour, the consistency of the resulting embedding space across subjects is significantly higher compared with autoLFADS and conv-pi-VAE, with or without test-time labels (one-way ANOVA $F(25.4)$ $P = 1.92 \times 10^{-16}$; Supplementary Table 1 and Fig. 1d,e). Qualitatively, it can be appreciated that both CEBRA-Behaviour and -Time have similar output embeddings whereas the latents from conv-pi-VAE, either with label priors or without labels, are not consistent (CEBRA does not need test-time labels), suggesting that the label prior strongly shapes the output embedding structure of conv-pi-VAE. We also considered correlations across repeated runs of the algorithm, and found higher consistency and lower variability with CEBRA (Extended Data Fig. 4).

Hypothesis-driven and discovery-driven analyses

Among the advantages of CEBRA are its collective flexibility, limited assumptions, and ability to test hypotheses. For the hippocampus, one can hypothesize that these neurons represent space[27,28] and therefore the behavioural label could be either position or velocity (Fig. 2a). In addition, considering structure in only the behavioural data (with CEBRA) could help refine which behavioural labels to use jointly with neural data (Fig. 2b). Conversely, for the sake of argument, we could have an alternative hypothesis: that the hippocampus does not map space, but simply maps the direction of travel or some other feature. Using the same model but hypothesis free, and using time for selection of contrastive pairs, is also possible, and/or a hybrid thereof (Fig. 2a,b).

We trained hypothesis-guided (supervised), time-only (self-supervised) and hybrid models across a range of input dimensions and embedded the neural latents into a 3D space for visualization. Qualitatively, we find that the position-based model produces a highly smooth embedding that shows the position of the animal—namely, there is a continuous 'loop' of latent dynamics around the track (Fig. 2b). This is consistent with what is known about the hippocampus[26] and shows the topology of the linear track with direction specificity whereas shuffling the labels, which breaks the correlation between neural activity and direction and position, produces an unstructured embedding (Fig. 2b).

CEBRA-Time produces an embedding that more closely resembles that of position (Fig. 2b). This also suggests that time contrastive learning captured the major latent space structure, independent of any label input, reinforcing the idea that CEBRA can serve both discovery- and hypothesis-driven questions (and that running both variants can be informative). The hybrid design, whose goal is to disentangle the latent to subspaces that are relevant to the given behavioural and residual temporal variance and noise, showed a structured embedding space similar to behaviour (Fig. 2b).

To quantify how CEBRA can disentangle which variable had the largest influence on embedding, we tested for encoding position, direction and combinations thereof (Fig. 2c). We find that position plus direction is the most informative label[29] (Fig. 2c and Extended Data Fig. 5a–d). This is evident both in the embedding and the value of the loss function on convergence, which serves as a 'goodness of fit' metric to select the best labels—that is, which label(s) produce the lowest loss at the same point in training (Extended Data Fig. 5e). Note that erroneous (shuffled) labels converge to considerably higher loss values.

To measure performance, we consider how well we could decode behaviour from the embeddings. As an additional baseline we performed linear dimensionality reduction with PCA. We used a k-nearest-neighbour (kNN) decoder for position and direction and measured

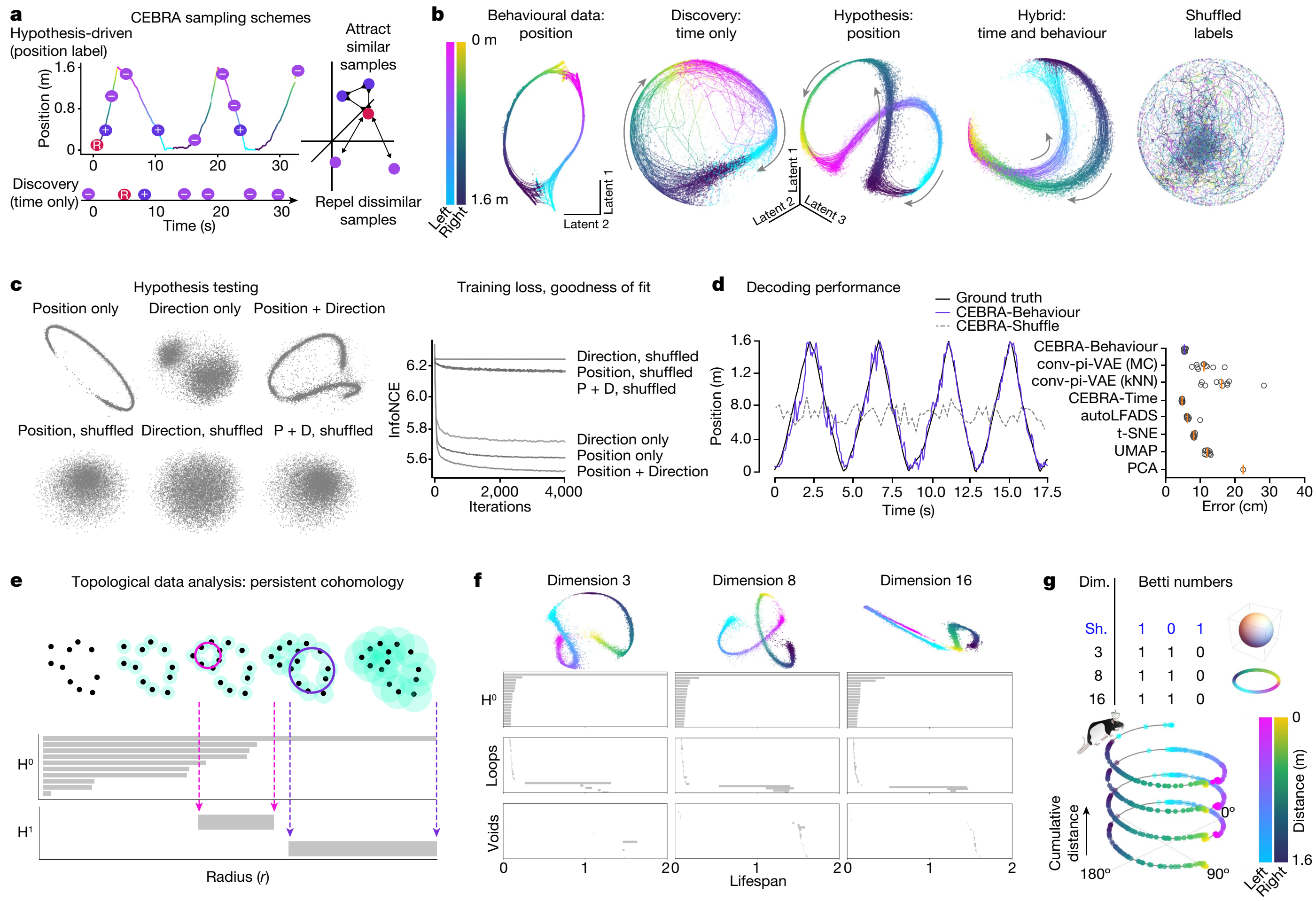


Fig. 2 | Hypothesis- and discovery-driven analysis with CEBRA. a, CEBRA can be used in any of three modes: hypothesis-driven mode, discovery-driven mode, or hybrid mode, which allows for weaker priors on the latent embedding. **b**, Left to right, CEBRA on behavioural data using position as a label, CEBRA-Time, CEBRA-Behaviour (on neural data) with position hypothesis, CEBRA-Hybrid (a five-dimensional space was used, in which 3D is first guided by both behaviour + time and the final 2D is guided by time) and shuffled (erroneous). **c**, Embeddings with position (P) only, direction (D) only and P + D only, and shuffled position only, direction only and P + D only, for hypothesis testing. The loss function can be used as a metric for embedding quality. **d**, Left, we utilized either hypothesis-driven P + D or shuffle (erroneous) to decode the position of the rat, which yielded a large difference in decoding performance: P + D $R^2 = 73.35$ versus −49.90% for shuffled, and median absolute error 5.8 versus 44.7 cm. Purple line represents decoding from the 3D hypothesis-based latent space; dashed line is shuffled. Right, performance across additional methods (orange bars indicate the median of individual runs ($n = 10$), indicated by black circles. Each run is averaged over three splits of the dataset). MC, Monte Carlo. **e**, Schematic showing how persistent cohomology is computed. Each data point is thickened to a ball of gradually expanding radius (r) while tracking the birth and death of 'cycles' in each dimension. Prominent lifespans, indicated by pink and purple arrows, are considered to determine Betti numbers. **f**, Top, visualization of neural embeddings computed with different input dimensions. Bottom, related persistent cohomology lifespan diagrams. **g**, Betti numbers from shuffled embeddings (sh.) and across increasing dimensions (dim.) of CEBRA, and the topology-preserving circular coordinates using the first cocycle from persistent cohomology analysis (Methods).

the reconstruction error. We find that CEBRA-Behaviour has significantly better decoding performance (Fig. 2d and Supplementary Video 1) compared with both pi-VAE and our conv-pi-VAE (one-way ANOVA, $F = 131$, $P = 3.6 \times 10^{-24}$), and also CEBRA-Time compared with unsupervised methods (autoLFADS, t-SNE, UMAP and PCA; one-way ANOVA, $F = 1{,}983$, $P = 6 \times 10^{-50}$; Supplementary Table 2). Zhou and Wei[5] reported a median absolute decoding error of 12 cm error whereas we achieved approximately 5 cm (Fig. 2d). CEBRA therefore allows for high-performance decoding and also ensures consistent embeddings.

Cohomology as a metric for robustness

Although CEBRA can be trained across a range of dimensions, and models can be selected based on decoding, goodness of fit and consistency, we also sought to find a principled approach to verify the robustness of embeddings that might yield insight into neural computations[30,31] (Fig. 2e). We used algebraic topology to measure the persistent cohomology as a comparison in regard to whether learned latent spaces are equivalent. Although it is not required to project embeddings onto a sphere, this has the advantage that there are default Betti numbers (for a d-dimensional uniform embedding, $H^0 = 1, H^1 = 0, \cdots, H^{d-1} = 1$—that is, 1,0,1 for the two-sphere). We used the distance from the unity line (and threshold based on a computed null shuffled distribution in Births versus Deaths to compute Betti numbers; Extended Data Fig. 6). Using CEBRA-Behaviour or -Time we find a ring topology (1,1,0; Fig. 2f), as one would expect from a linear track for place cells. We then computed the Eilenberg–MacLane coordinates for the identified cocycle (H^1) for each model[32,33]—this allowed us to map each time point to topology-preserving coordinates—and indeed we find that the ring topology for the CEBRA models matches space (position) across dimensions (Fig. 2g and Extended Data Fig. 6). Note that this topology differs from

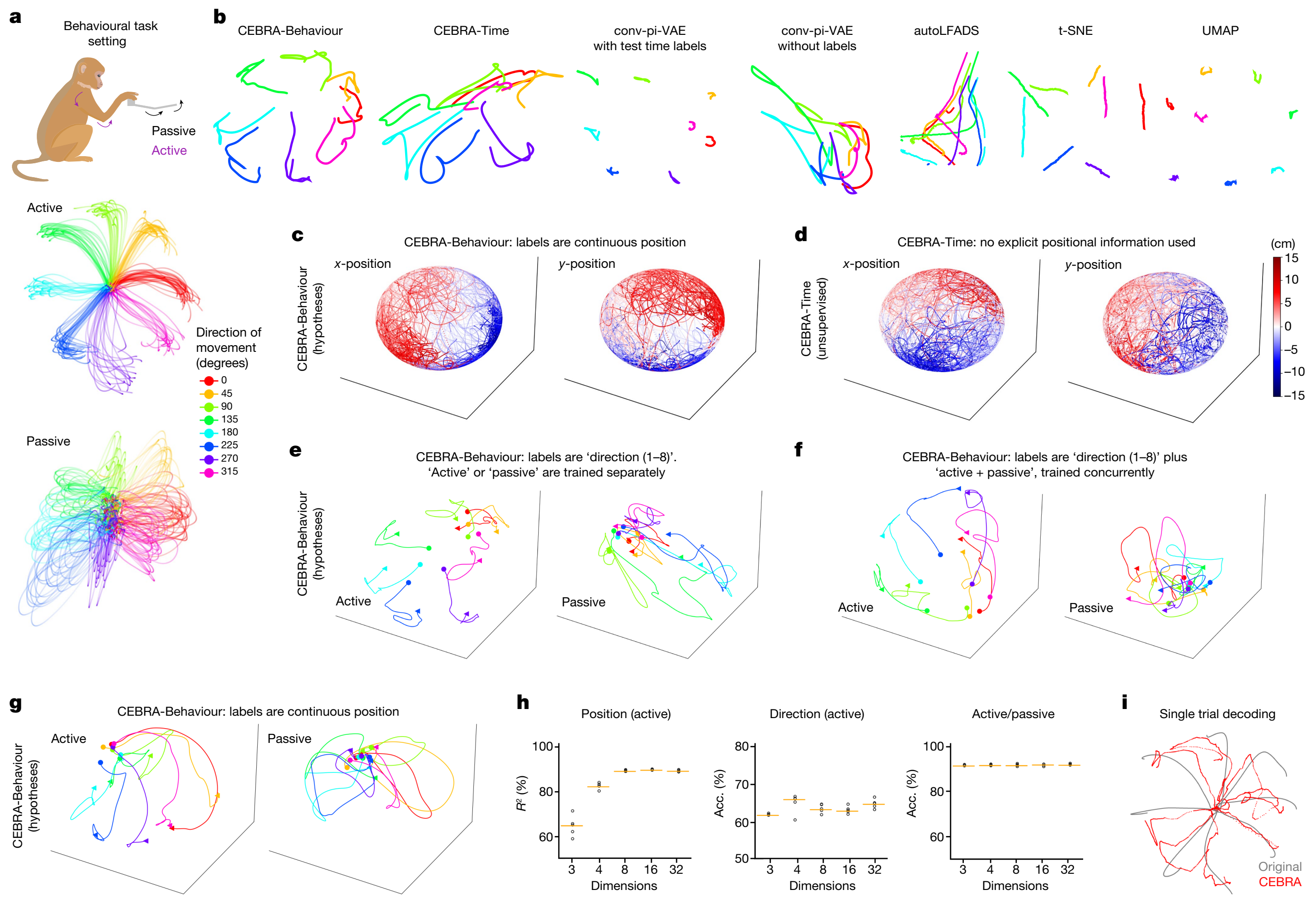


Fig. 3 | Forelimb movement behaviour in a primate. a, The monkey makes either active or passive movements in eight directions. Data derived from area 2 of primary S1 from Chowdhury et al.[34]. Cartoon from scidraw.io. **b**, Embeddings of active trials generated with 4D CEBRA-Behaviour, 4D CEBRA-Time, 4D autoLFADS, 4D conv-pi-VAE variants, 2D *t*-SNE and 2D UMAP. The embeddings of trials ($n = 364$) for each direction are post hoc averaged. **c**, CEBRA-Behaviour trained with (x,y) position of the hand. Left, colour coded to x position; right, colour coded to y position. **d**, CEBRA-Time with no external behaviour variables. Colour coded as in **c**. **e**, CEBRA-Behaviour embedding trained using a 4D latent space, with discrete target direction as behaviour labels, trained and plotted separately for active and passive trials. **f**, CEBRA-Behaviour embedding trained using a 4D latent space, with discrete target direction and active and passive trials as auxiliary labels plotted separately, active versus passive trials. **g**, CEBRA-Behaviour embedding trained with a 4D latent space using active and passive trials with continuous (x,y) position as auxiliary labels plotted separately, active versus passive trials. The trajectory of each direction is averaged across trials ($n = 18$–30 each, per direction) over time. Each trajectory represents 600 ms from −100 ms before the start of the movement. **h**, Left to right, decoding performance of: position using CEBRA-Behaviour trained with (x,y) position (active trials); target direction using CEBRA-Behaviour trained with target direction (active trials); and active versus passive accuracy (Acc.) using CEBRA-Behaviour trained with both active and passive movements. In each case we trained and evaluated five seeds, represented by black dots; orange line represents the median. **i**, Decoded trajectory of hand position using CEBRA-Behaviour trained on active trial with (x,y) position of the hand. The grey line represents a true trajectory and the red line represents a decoded trajectory.

(1,0,1)—that is, Betti numbers for a uniformly covered sphere—which in our setting would indicate a random embedding as found by shuffling (Fig. 2g).

Multi-session, multi-animal CEBRA

CEBRA can also be used to jointly train across sessions and different animals, which can be highly advantageous when there is limited access to simultaneously recorded neurons or when looking for animal-invariant features in the neural data. We trained CEBRA across animals within each multi-animal dataset and find that this joint embedding allows for even more consistent embeddings across subjects (Extended Data Fig. 7a–c; one-sided, paired *t*-tests; Allen data: $t = -5.80$, $P = 5.99 \times 10^{-5}$; hippocampus: $t = -2.22$, $P = 0.024$).

Although consistency increased, it is not a priori clear that decoding from 'pseudosubjects' would be equally good because there could be session- or animal-specific information that is lost in pseudodecoding (because decoding is usually performed within the session). Alternatively, if this joint latent space was as high performance as the single subject, that would suggest that CEBRA is able to produce robust latent spaces across subjects. Indeed, we find no loss in decoding performance (Extended Data Fig. 7c).

It is also possible to rapidly decode from a new session that is unseen during training, which is an attractive setting for brain machine interface deployment. We show that, by pretraining on a subset of the subjects, we can apply and rapidly adapt CEBRA-Behaviour on unseen data (that is, it runs at 50–100 steps s^{-1}, and positional decoding error already decreased by 10 cm after adapting the pretrained network for one step). Lastly, we can achieve a lower error more rapidly compared with training fully on the unseen individual (Extended Data Fig. 7d). Collectively, this shows that CEBRA can rapidly produce high-performance, consistent and robust latent spaces.

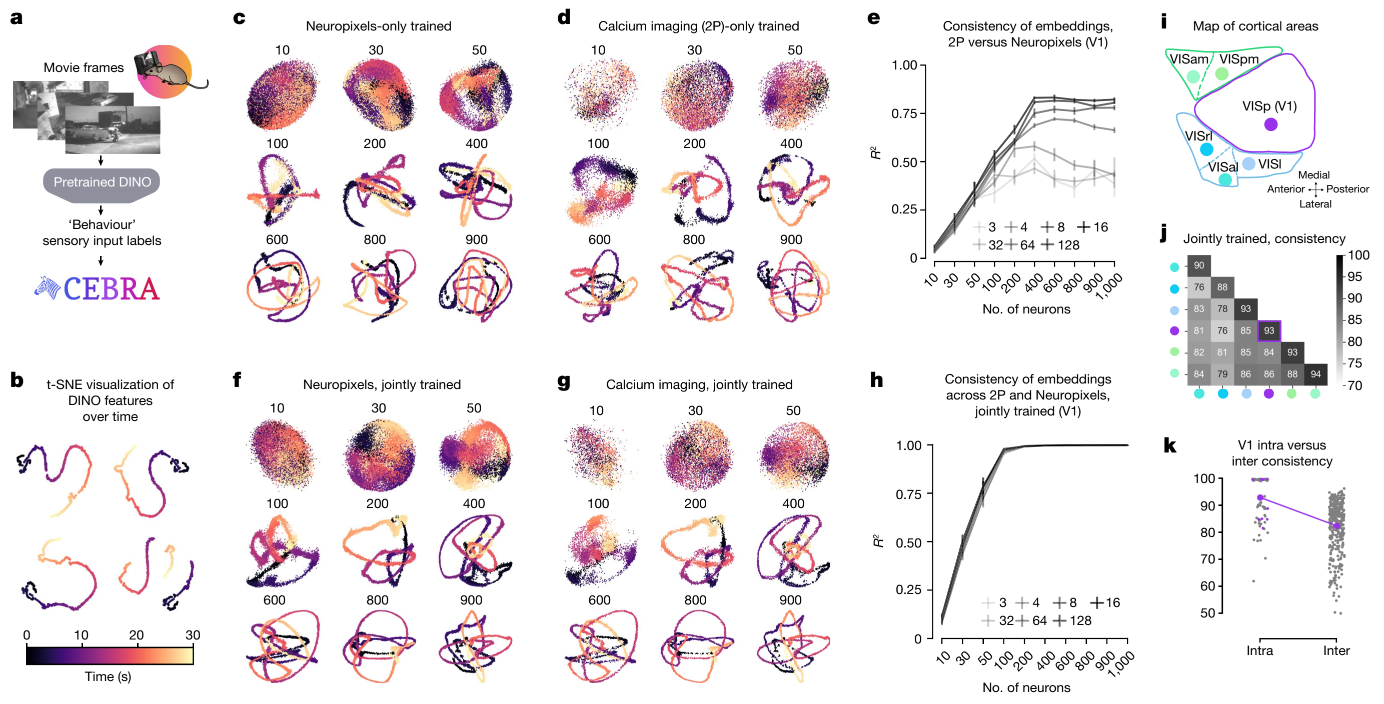


Fig. 4 | Spikes and calcium signalling show similar CEBRA embeddings. **a**, CEBRA-Behaviour can use frame-by-frame video features as a label of sensory input to extract the neural latent space of the visual cortex of mice watching a video. Cartoon from scidraw.io. **b**, *t*-SNE visualization of the DINO features of video frames from four different DINO configurations (latent size, model size), all showing continuous evolution of video frames over time. **c**,**d**, Visualization of trained 8D latent CEBRA-Behaviour embeddings with Neuropixels (NP) data (**c**) or calcium imaging (2P) (**d**). Numbers above each embedding indicate neurons subsampled from the multi-session concatenated dataset. Colour map as in **b**. **e**, Linear consistency between embeddings trained with either calcium imaging or Neuropixels data ($n = 10$–$1{,}000$ neurons, across $n = 5$ shuffles of neural data; mean values ± s.e.m.). **f**,**g**, Visualization of CEBRA-Behaviour embedding (8D) trained jointly with Neuropixels (**f**) and calcium imaging (**g**). Colour map as in **b**. **h**, Linear consistency between embeddings of calcium imaging and Neuropixels trained jointly using a multi-session CEBRA model ($n = 10$–1000 neurons, across $n = 5$ shuffles of neural data; mean values ± s.e.m.). **i**, Diagram of mouse primary visual cortex (V1, VISp) and higher visual areas. **j**, CEBRA-Behaviour 32D model jointly trained with 2P + NP incorporating 400 neurons, followed by measurement of consistency within or across areas (2P versus NP) across two unique sets of disjoint neurons for three seeds and averaged. **k**, Models trained as in **h**, with intra-V1 consistency measurement versus all interarea versus V1 comparison. Purple dots indicate mean of V1 intra-V1 consistency (across $n = 120$ runs) and inter-V1 consistency ($n = 120$ runs). Intra-V1 consistency is significantly higher than interarea consistency (one-sided Welch's *t*-test, $t(12.30) = 4.55$, $P = 0.00019$).

Latent dynamics during a motor task

We next consider an eight-direction 'centre-out' reaching task paired with electrophysiology recordings in primate somatosensory cortex (S1)[34] (Fig. 3a). The monkey performed many active movements, and in a subset of trials experienced randomized bumps that caused passive limb movement. CEBRA produced highly informative visualizations of the data compared with other methods (Fig. 3b), and CEBRA-Behaviour can be used to test the encoding properties of S1. Using either position or time information showed embeddings with clear positional encoding (Fig. 3c,d and Extended Data Fig. 8a–c).

To test how directional information and active versus passive movements influence population dynamics in S1 (refs. 34–36), we trained embedding spaces with directional information and then either separated the trials into active and passive for training (Fig. 3e) or trained jointly and post hoc plotted separately (Fig. 3f). We find striking similarities suggesting that active versus passive strongly influences the neural latent space: the embeddings for active trials show a clear start and stop whereas for passive trials they show a continuous trajectory through the embedding, independently of how they are trained. This finding is confirmed in embeddings that used only the continuous position of the end effector as the behavioural label (Fig. 3g). Notably, direction is a less prominent feature (Fig. 3g) although they are entangled parameters in this task.

As the position and active or passive trial type appear robust in the embeddings, we further explored the decodability of the embeddings. Both position and trial type were readily decodable from 8D+ embeddings with a kNN decoder trained on position only, but directional information was not as decodable (Fig. 3h). Here too, the loss function value is informative for goodness of fit during hypothesis testing (Extended Data Fig. 8d–f). Notably, we could recover the hand trajectory with $R^2 = 88\%$ (concatenated across 26 held-out test trials; Fig. 3i) using a 16D CEBRA-Behaviour model trained on position (Fig. 3i). For comparison, an L1 regression using all neurons achieved $R^2 = 74\%$ and 16D conv-pi-VAE achieved $R^2 = 82\%$. We also tested CEBRA on an additional monkey dataset (mc-maze) presented in the Neural Latent Benchmark[37], in which it achieved state-of-the-art behaviour (velocity) decoding performance (Extended Data Fig. 8).

Consistent embeddings across modalities

Although CEBRA is agnostic to the recording modality of neural data, do different modalities produce similar latent embeddings? Understanding the relationship of calcium signalling and electrophysiology is a debated topic, yet an underlying assumption is that they inherently represent related, yet not identical, information. Although there is a wealth of excellent tools aimed at inferring spike trains from calcium data, currently the pseudo-R^2 of algorithms on paired spiking and calcium data tops out at around 0.6 (ref. 38). Nonetheless, it is clear that recording with either modality has led to similar global conclusions—for example, grid cells can be uncovered in spiking or calcium signals[33,39],

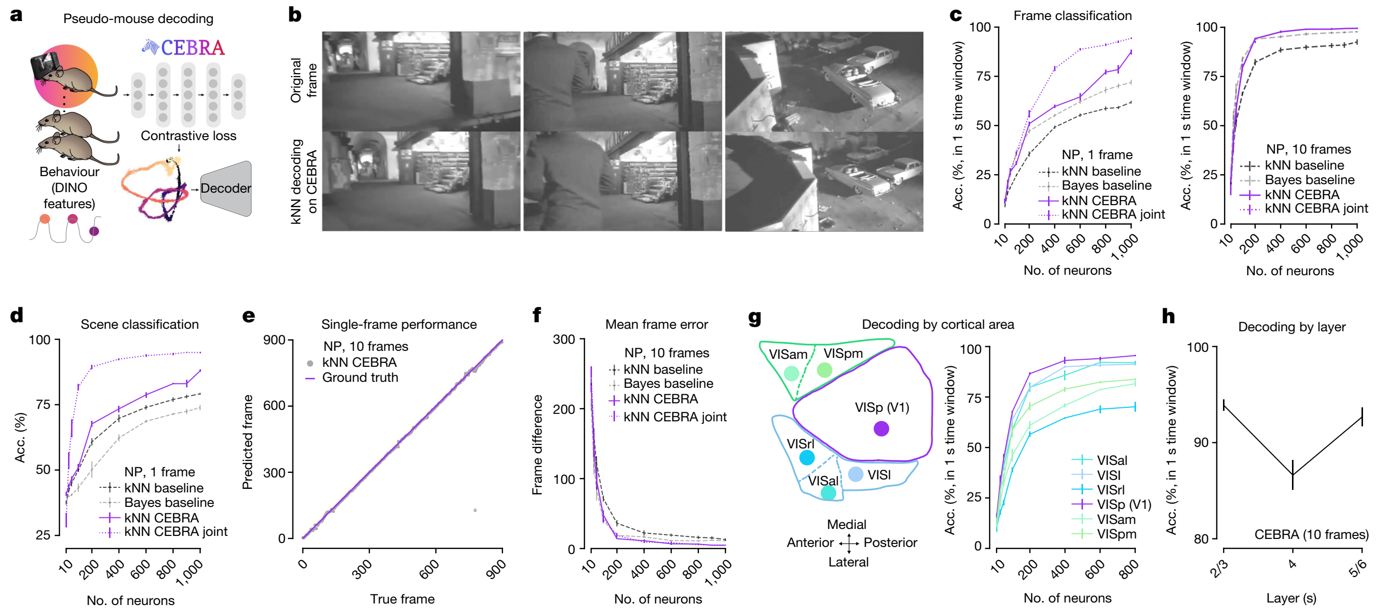


Fig. 5 | Decoding of natural video features from mouse visual cortical areas. a, Schematic of the CEBRA encoder and kNN (or naive Bayes) decoder. **b**, Examples of original frames (top row) and frames decoded from CEBRA embedding of V1 calcium recording using kNN decoding (bottom row). The last repeat among ten was used as the held-out test. **c**, Decoding accuracy measured by considering a predicted frame being within 1 s of the true frame as a correct prediction using CEBRA (NP only), jointly trained (2P + NP) or a baseline population vector plus kNN or naive Bayes decoder using either a one-frame (33 ms) receptive field (left) or ten frames (330 ms) (right); results shown for Neuropixels dataset (V1 data); for each neuron number we have *n* = 5 shuffles, mean ± s.e.m. **d**, Decoding accuracy measured by correct scene prediction using either CEBRA (NP only), jointly trained (2P + NP) or baseline population vector plus kNN or Bayes decoder using a one-frame (33 ms) receptive field (V1 data); *n* = 5 shuffles per neuron number, mean ± s.e.m. **e**, Single-frame ground truth frame ID versus predicted frame ID for Neuropixels using a CEBRA-Behaviour model trained with a 330 ms receptive field (1,000 V1 neurons across mice used). **f**, Mean absolute error of the correct frame index; shown for baseline and CEBRA models as computed in **c**–**e**. **g**, Diagram of the cortical areas considered and decoding performance from CEBRA (NP only), ten-frame receptive field; *n* = 3 shuffles for each area and number of neurons, mean ± s.e.m. **h**, V1 decoding performance versus layer category using 900 neurons with a 330 ms receptive field CEBRA-Behaviour model; *n* = 5 shuffles for each layer, mean ± s.e.m.

reward prediction errors can be found in dopamine neurons across species and recording modalities[40–42], and visual cortex shows orientation tuning across species and modalities[43–45].

We aimed to formally study whether CEBRA could capture the same neural population dynamics either from spikes or calcium imaging. We utilized a dataset from the Allen Brain Observatory where mice passively watched three videos repeatedly. We focused on paired data from ten repeats of 'Natural Movie 1' where neural data were recorded with either Neuropixels (NP) probes or calcium imaging with a two-photon (2P) microscope (from separate mice)[46,47]. Note that, although the data we have considered thus far have goal-driven actions of the animals (such as running down a linear track or reaching for targets), this visual cortex dataset was collected during passive viewing (Fig. 4a).

We used the video features as 'behaviour' labels by extracting high-level visual features from the video on a frame-by-frame basis with DINO, a powerful vision transformer model[48]. These were then used to sample the neural data with feature-labels (Fig. 4b). Next, we used either Neuropixels or 2P data (each with multi-session training) to generate (from 8D to 128D) latent spaces from varying numbers of neurons recorded from primary visual cortex (V1) (Fig. 4c,d). Visualization of CEBRA-Behaviour showed trajectories that smoothly capture the video of either modality with an increasing number of neurons. This is reflected quantitatively in the consistency metric (Fig. 4e). Strikingly, CEBRA-Time efficiently captured the ten repeats of the video (Extended Data Fig. 9), which was not captured by other methods. This result demonstrates that there is a highly consistent latent space independent of the recording method.

Next, we stacked neurons from different mice and modalities and then sampled random subsets of V1 neurons to construct a pseudomouse. We did not find that joint training lowered consistency within modality (Extended Data Fig. 10a,b) and, overall, we found considerable improvement in consistency with joint training (Fig. 4f–h).

Using CEBRA-Behaviour or -Time, we trained models on five higher visual areas and measured consistency with and without joint training, and within or across areas. Our results show that, with joint training, intra-area consistency is higher compared with other areas (Fig. 4i–k), suggesting that CEBRA is not removing biological differences across areas, which have known differences in decodability and feature representations[49,50]. Moreover, we tested within modality and find a similar effect for CEBRA-Behaviour and -Time within recording modality (Extended Data Fig. 10c–f).

Decoding of natural videos from cortex

We performed V1 decoding analysis using CEBRA models that are either joint-modality trained, single-modality trained or with a baseline population vector paired with a simple kNN or naive Bayes decoder. We aimed to determine whether we could decode, on a frame-by-frame basis, the natural video watched by the mice. We used the final video repeat as a held-out test set and nine repeats as the training set. We achieved greater than 95% decoding accuracy, which is significantly better than baseline decoding methods (naive Bayes or kNN) for Neuropixels recordings, and joint-training CEBRA outperformed Neuropixels-only CEBRA-based training (single frame: one-way ANOVA, $F(3,197) = 5.88$, $P = 0.0007$; Supplementary Tables 3–5, Fig. 5a–d and Extended Data Fig. 10g,h). Accuracy was defined by either the fraction of correct frames within a 1 s window or identification of the correct scene. Frame-by-frame results also showed reduced frame ID errors (one-way ANOVA, $F(3,16) = 20.22$, $P = 1.09 \times 10^{-5}$, $n = 1{,}000$ neurons; Supplementary Table 6), which can be seen in Fig. 5e,f, Extended Data Fig. 10i and Supplementary Video 2. The DINO features themselves did not drive performance, because shuffling of features showed poor decoding (Extended Data Fig. 10j).

Lastly, we tested decoding from other higher visual areas using DINO features. Overall, decoding from V1 had the highest performance and VISrl the lowest (Fig. 5g and Extended Data Fig. 10k). Given the high decoding performance of CEBRA, we tested whether there was a particular V1 layer that was most informative. We leveraged CEBRA-Behaviour by training models on each category and found that layers 2/3 and 5/6 showed significantly higher decoding performance compared with layer 4 (one-way ANOVA, $F(2,12) = 9.88$, $P = 0.003$; Fig. 5h). Given the known cortical connectivity, this suggests that the nonthalamic input layers render frame information more explicit, perhaps via feedback or predictive processing.

Discussion

CEBRA is a nonlinear dimensionality reduction method newly developed to explicitly leverage auxiliary (behaviour) labels and/or time to discover latent features in time series data—in this case, latent neural embeddings. The unique property of CEBRA is the extension and generalization of the standard InfoNCE objective by introduction of a variety of different sampling strategies tuned for usage of the algorithm in the experimental sciences and for analysis of time series datasets, and it can also be used for supervised and self-supervised analysis, thereby directly facilitating hypothesis- and discovery-driven science. It produces both consistent embeddings across subjects (thus showing common structure) and can find the dimensionality of neural spaces that are topologically robust. Although there remains a gap in our understanding of how these latent spaces map to neural-level computations, we believe this tool provides an advance in our ability to map behaviour to neural populations. Moreover, because pretrained CEBRA models can be used for decoding in new animals within tens of steps (milliseconds), we can thereby obtain equal or better performance compared with training on the unseen animal alone.

Dimensionality reduction is often tightly linked to data visualization, and here we make an empirical argument that ultimately this is useful only when obtaining consistent results and discovering robust features. Unsupervised *t*-SNE and UMAP are examples of algorithms widely used in life sciences for discovery-based analysis. However, they do not leverage time and, for neural recordings, this is always available and can be used. Even more critical is that concatenation of data from different animals can lead to shifted clusters with *t*-SNE or UMAP due to inherent small changes across animals or in how the data were collected. CEBRA allows the user to remove this unwanted variance and discover robust latents that are invariant to animal ID, sessions or any-other-user-defined nuisance variable. Collectively we believe that CEBRA will become a complement to (or replacement for) these methods such that, at minimum, the structure of time in the neural code is leveraged and robustness is prioritized.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41586-023-06031-6.

1. Urai, A. E., Doiron, B., Leifer, A. M. & Churchland, A. K. Large-scale neural recordings call for new insights to link brain and behavior. *Nat. Neurosci.* **25**, 11–19 (2021).
2. Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A. & Poeppel, D. Neuroscience needs behavior: correcting a reductionist bias. *Neuron* **93**, 480–490 (2017).
3. Jazayeri, M. & Ostojic, S. Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Curr. Opin. Neurobiol.* **70**, 113–120 (2021).
4. Humphries, M. D. Strong and weak principles of neural dimension reduction. *Neuron. Behav. Data Anal. Theory* https://nbdt.scholasticahq.com/article/24619 (2020).
5. Zhou, D., & Wei, X. Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE. *Adv. Neural Inf. Process. Syst.* https://proceedings.neurips.cc//paper/2020/file/510f2318f324cf07fce24c3a4b89c771-Paper.pdf (2020).
6. Vargas-Irwin, C. E. et al. Decoding complete reach and grasp actions from local primary motor cortex populations. *J. Neurosci.* **30**, 9659–9669 (2010).
7. Okorokova, E. V., Goodman, J. M., Hatsopoulos, N. G. & Bensmaia, S. J. Decoding hand kinematics from population responses in sensorimotor cortex during grasping. *J. Neural Eng.* **17**, 046035 (2020).
8. Yu, B. M. et al. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *J. Neurophysiol.* **102**, 614–635 (2008).
9. Churchland, M. et al. Neural population dynamics during reaching. *Nature* **487**, 51–56 (2012).
10. Gallego, J. A. et al. Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nat. Commun.* **9**, 4233 (2018).
11. McInnes, L., Healy, J., Saul, N. & Großberger, L. UMAP: Uniform Manifold Approximation and Projection for dimension reduction. *J. Open Source Softw.* **3**, 861 (2018).
12. Maaten, L. V., Postma, E. O. & Herik, J. V. Dimensionality reduction: a comparative review. *J. Mach. Learn. Res.* **10**, 13 (2009).
13. Roeder, G., Metz, L. & Kingma, D. P. On linear identifiability of learned representations. *Proc. Mach. Learn. Res.* **139**, 9030–9039 (2021).
14. Hyvärinen, A., Sasaki, H. & Turner, R. E. Nonlinear ICA using auxiliary variables and generalized contrastive learning. *Proc. Mach. Learn. Res.* **89**, 859–868 (2019).
15. Sani, O. G., Abbaspourazad, H., Wong, Y. T., Pesaran, B. & Shanechi, M. M. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. *Nat. Neurosci.* **24**, 140–149 (2020).
16. Klindt, D. A. et al. Towards nonlinear disentanglement in natural data with temporal sparse coding. *International Conference on Learning Representations* https://openreview.net/forum?id=EbIDjBynYJ8 (2021).
17. Pandarinath, C. et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nat. Methods* **15**, 805–815 (2017).
18. Prince, L. Y., Bakhtiari, S., Gillon, C. J., & Richards, B. A. Parallel inference of hierarchical latent dynamics in two-photon calcium imaging of neuronal populations. Preprint at https://www.biorxiv.org/content/10.1101/2021.03.05.434105v1 (2021).
19. Gutmann, M. U. & Hyvärinen, A. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.* **13**, 307–361 (2012).
20. Oord, A. V., Li, Y. & Vinyals, O. Representation learning with contrastive predictive coding. Preprint at https://doi.org/10.48550/arXiv.1807.03748 (2018).
21. Khosla, P. et al. Supervised contrastive learning. *Adv. Neural Inf. Process. Syst.* **33**, 18661–18673 (2020).
22. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. E. A simple framework for contrastive learning of visual representations. *Proc. Mach. Learn. Res.* **119**, 1597–1607 (2020).
23. Hälvä, H. et al. Disentangling identifiable features from noisy data with structured nonlinear ICA. *Adv. Neural Inf. Process. Syst.* **34**, 1624–1633 (2021).
24. Zimmermann, R. S., Sharma, Y., Schneider, S., Bethge, M. & Brendel, W. Contrastive learning inverts the data generating process. *Proc. Mach. Learn. Res.* **139**, 12979–12990 (2021).
25. Keshtkaran, M. R. et al. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *Nat. Methods* **19**, 1572–1577 (2022).
26. Grosmark, A. D. & Buzsáki, G. Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. *Science* **351**, 1440–1443 (2016).
27. Huxter, J. R., Burgess, N. & O'Keefe, J. Independent rate and temporal coding in hippocampal pyramidal cells. *Nature* **425**, 828–832 (2003).
28. Moser, E. I., Kropff, E. & Moser, M. Place cells, grid cells, and the brain's spatial representation system. *Annu. Rev. Neurosci.* **31**, 69–89 (2008).
29. Dombeck, D. A., Harvey, C. D., Tian, L., Looger, L. L. & Tank, D. W. Functional imaging of hippocampal place cells at cellular resolution during virtual navigation. *Nat. Neurosci.* **13**, 1433–1440 (2010).
30. Curto, C. What can topology tell us about the neural code? *Bull. Am. Math. Soc* **54**, 63–78 (2016).
31. Chaudhuri, R., Gerçek, B., Pandey, B., Peyrache, A. & Fiete, I. R. The intrinsic attractor manifold and population dynamics of a canonical cognitive circuit across waking and sleep. *Nat. Neurosci.* **22**, 1512–1520 (2019).
32. Silva, V. D., Morozov, D. & Vejdemo-Johansson, M. Persistent cohomology and circular coordinates. *Discrete Comput. Geom.* **45**, 737–759 (2009).
33. Gardner, R. J. et al. Toroidal topology of population activity in grid cells. *Nature* **602**, 123–128 (2022).
34. Chowdhury, R. H., Glaser, J. I. & Miller, L. E. Area 2 of primary somatosensory cortex encodes kinematics of the whole arm. *eLife* **9**, e48198 (2019).
35. Prud'homme, M. J. & Kalaska, J. F. Proprioceptive activity in primate primary somatosensory cortex during active arm reaching movements. *J. Neurophysiol.* **72**, 2280–2301 (1994).
36. London, B. M. & Miller, L. E. Responses of somatosensory area 2 neurons to actively and passively generated limb movements. *J. Neurophysiol.* **109**, 1505–1513 (2013).
37. Pei, F. et al. Neural Latents Benchmark '21: Evaluating latent variable models of neural population activity. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* https://openreview.net/forum?id=KVMS3fl4Rsv (2021).
38. Berens, P. et al. Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS Comput. Biol.* **14**, e1006157 (2018).
39. Hafting, T., Fyhn, M., Molden, S., Moser, M. & Moser, E. I. Microstructure of a spatial map in the entorhinal cortex. *Nature* **436**, 801–806 (2005).
40. Schultz, W., Dayan, P. & Montague, P. R. A neural substrate of prediction and reward. *Science* **275**, 1593–1599 (1997).

41. Cohen, J. Y., Haesler, S., Vong, L., Lowell, B. B. & Uchida, N. Neuron-type specific signals for reward and punishment in the ventral tegmental area. *Nature* **482**, 85–88 (2011).
42. Menegas, W. et al. Dopamine neurons projecting to the posterior striatum form an anatomically distinct subclass. *eLife* **4**, e10032 (2015).
43. Hubel, D. H. & Wiesel, T. N. Ferrier lecture – functional architecture of macaque monkey visual cortex. *Proc. R. Soc. Lond. B Biol. Sci.* **198**, 1–59 (1977).
44. Niell, C. M., Stryker, M. P. & Keck, W. M. Highly selective receptive fields in mouse visual cortex. *J. Neurosci.* **28**, 7520–7536 (2008).
45. Ringach, D. L. et al. Spatial clustering of tuning in mouse primary visual cortex. *Nat. Commun.* **7**, 12270 (2016).
46. de Vries, S. E. et al. A large-scale standardized physiological survey reveals functional organization of the mouse visual cortex. *Nat. Neurosci.* **23**, 138–151 (2019).
47. Siegle, J. H. et al. Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature* **592**, 86–92 (2021).
48. Caron, M. et al. Emerging properties in self-supervised vision transformers. *IEEE/CVF International Conference on Computer Vision* 9630–9640 (2021).
49. Esfahany, K., Siergiej, I., Zhao, Y. & Park, I. M. Organization of neural population code in mouse visual system. *eNeuro* **5**, ENEURO.0414-17.2018 (2018).
50. Jin, M. & Glickfeld, L. L. Mouse higher visual areas provide both distributed and specialized contributions to visually guided behaviors. *Curr. Biol.* **30**, 4682–4692 (2020).

Methods

Datasets

Artificial spiking dataset. The synthetic spiking data used for benchmarking in Fig. 1 were adopted from Zhou and Wei[5]. The continuous 1D behaviour variable $c \in [0, 2\pi)$ was sampled uniformly in the interval $[0, 2\pi)$. The true 2D latent variable $\mathbf{z} \in \mathbb{R}^2$ was then sampled from a Gaussian distribution $\mathcal{N}(\mu(c), \Sigma(c))$ with mean $\mu(c) = (c, 2\mathrm{sinc})^\top$ and covariance $\Sigma(c) = \mathrm{diag}(0.6 - 0.3\ |\mathrm{sinc}|, 0.3\ |\mathrm{sinc}|)$. After sampling, the 2D latent variable $\mathbf{z}$ was mapped to the spiking rates of 100 neurons by the application of four randomly initialized RealNVP[51] blocks. Poisson noise was then applied to map firing rates onto spike counts. The final dataset consisted of 1.5×10^4 data points for 100 neurons ([number of samples, number of neurons]) and was split into train (80%) and validation (20%) sets. We quantified consistency across the entire dataset for all methods. Additional synthetic data, presented in Extended Data Fig. 1, were generated by varying noise distribution in the above generative process. Beside Poisson noise, we used additive truncated ([0,1000]) Gaussian noise with s.d. = 1 and additive uniform noise defined in [0,2], which was applied to the spiking rate. We also adapted Poisson spiking by simulating neurons with a refractory period. For this, we scaled the spiking rates to an average of 110 Hz. We sampled interspike intervals from an exponential distribution with the given rate and added a refractory period of 10 ms.

Rat hippocampus dataset. We used the dataset presented in Grosmark and Buzsáki[26]. In brief, bilaterally implanted silicon probes recorded multicellular electrophysiological data from CA1 hippocampus areas from each of four male Long–Evans rats. During a given session, each rat independently ran on a 1.6-m-long linear track where they were rewarded with water at each end of the track. The numbers of recorded putative pyramidal neurons for each rat ranged between 48 and 120. Here, we processed the data as in Zhou and Wei[5]. Specifically, the spikes were binned into 25 ms time windows. The position and running direction (left or right) of the rat were encoded into a 3D vector, which consisted of the continuous position value and two binary values indicating right or left direction. Recordings from each rat were parsed into trials (a round trip from one end of the track as a trial) and then split into train, validation and test sets with a $k = 3$ nested cross-validation scheme for the decoding task.

Macaque dataset. We used the dataset presented in Chowdhury et al.[34] In brief, electrophysiological recordings were performed in Area 2 of somatosensory cortex (S1) in a rhesus macaque (monkey) during a centre-out reaching task with a manipulandum. Specifically, the monkey performed an eight-direction reaching task in which on 50% of trials it actively made centre-out movements to a presented target. The remaining trials were 'passive' trials in which an unexpected 2 Newton force bump was given to the manipulandum towards one of the eight target directions during a holding period. The trials were aligned as in Pei et al.[37], and we used the data for −100 and 500 ms from movement onset. We used 1 ms time bins and convolved the data using a Gaussian kernel with s.d. = 40 ms.

Mouse visual cortex datasets. We utilized the Allen Institute two-photon calcium imaging and Neuropixels data recorded from five mouse visual and higher visual cortical areas (VISp, VISl, VISal, VISam, VISpm and VISrl) during presentation of a monochrome video with 30 Hz frame rate, as presented previously[46,47,52]. For calcium imaging (2P) we used the processed dataset from Vries et al.[46] with a sampling rate of 30 Hz, aligned to the video frames. We considered the recordings from excitatory neurons (Emx1-IRES-Cre, Slc17a7-IRES2-Cre, Cux2-CreERT2, Rorb-IRES2-Cre, Scnn1a-Tg3-Cre, Nr5a1-Cre, Rbp4-Cre_KL100, Fezf2-CreER and Tlx3-Cre_PL56) in the 'Visual Coding-2P' dataset. Ten repeats of the first video (Movie 1) were shown in all session types (A, B and C) for each mouse and we used those neurons that were recorded in all three session types, found via cell registration[46]. The Neuropixels recordings were obtained from the 'Brain Observatory 1.1' dataset[47]. We used the preprocessed spike timings and binned them to a sampling frequency of 120 Hz, aligned with the video timestamps (exactly four bins aligned with each frame). The dataset contains recordings for ten repeats, and we used the same video (Movie 1) that was used for the 2P recordings. For analysis of consistency across the visual cortical areas we used a disjoint set of neurons for each seed, to avoid higher intraconsistency due to overlapping neuron identities. We made three disjoint sets of neurons by considering only neurons from session A (for 2P data) and nonoverlapping random sampling for each seed.

CEBRA model framework

Notation. We will use $\mathbf{x}$,$\mathbf{y}$ as general placeholder variables and denote the multidimensional, time-varying signal as $\mathbf{s}_t$, parameterized by time t. The multidimensional, continuous context variable $\mathbf{c}_t$ contains additional information about the experimental condition and additional recordings, similar to the discrete categorical variable k_t.

The exact composition of $\mathbf{s}$, $\mathbf{c}$ and k depends on the experimental context. CEBRA is agnostic to exact signal types; with the default parameterizations, $\mathbf{s}_t$ and $\mathbf{c}_t$ can have up to an order of hundreds or thousands of dimensions. For even higher-dimensional datasets (for example, raw video, audio and so on) other optimized deep learning tools can be used for feature extraction before the application of CEBRA.

Applicable problem setup. We refer to $\mathbf{x} \in X$ as the reference sample and to $\mathbf{y} \in Y$ as a corresponding positive or negative sample. Together, $(\mathbf{x}, \mathbf{y})$ form a positive or negative pair based on the distribution from which $\mathbf{y}$ is sampled. We denote the distribution and density function of $\mathbf{x}$ as $p(\mathbf{x})$, the conditional distribution and density of the positive sample $\mathbf{y}$ given $\mathbf{x}$ as $p(\mathbf{y}|\mathbf{x})$ and the conditional distribution and density of the negative sample $\mathbf{y}$ given $\mathbf{x}$ as $q(\mathbf{y}|\mathbf{x})$.

After sampling—and irrespective of whether we are considering a positive or negative pair—samples $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{y} \in \mathbb{R}^{D'}$ are encoded by feature extractors $\mathbf{f}: X \mapsto Z$ and $\mathbf{f}': Y \mapsto Z$. The feature extractors map both samples from signal space $X \subseteq \mathbb{R}^D$, $Y \subseteq \mathbb{R}^{D'}$ into a common embedding space $Z \subseteq \mathbb{R}^E$. The design and parameterization of the feature extractor are chosen by the user of the algorithm. Note that spaces X and Y and their corresponding feature extractors can be the same (which is the case for single-session experiments in this work), but that this is not a strict requirement within the CEBRA framework (for example, in multi-session training across animals or modalities, X and Y are selected as signals from different mice or modalities, respectively). It is also possible to include the context variable (for example, behaviour) into X, or to set $\mathbf{x}$ to the context variable and $\mathbf{y}$ to the signal variable.

Given two encoded samples, a similarity measure $\phi: Z \times Z \mapsto \mathbb{R}$ assigns a score to a pair of embeddings. The similarity measure needs to assign a higher score to more similar pairs of points, and to have an upper bound. For this work we consider the dot product between normalized feature vectors, $\phi(\mathbf{z}, \mathbf{z}') = \mathbf{z}^\top \mathbf{z}'/\tau$, in most analyses (latents on a hypersphere) or the negative mean squared error, $\phi(\mathbf{z}, \mathbf{z}') = -\|\mathbf{z} - \mathbf{z}'\|^2/\tau$ (latents in Euclidean space). Both metrics can be scaled by a temperature parameter τ that is either fixed or jointly learned with the network. Other L_p norms and other similarity metrics, or even a trainable neural network (a so-called projection head commonly used in contrastive learning algorithms[14,22]), are possible choices within the CEBRA software package. The exact choice of ϕ shapes the properties of the embedding space and encodes assumptions about distributions p and q.

The technique requires paired data recordings—for example, as is common in aligned time series. The signal $\mathbf{s}_t$, continuous context $\mathbf{c}_t$ and discrete context k_t are synced in their time point t. How the reference, positive and negative samples are constructed from these available signals is a configuration choice made by the algorithm user, and depends on the scientific question under investigation.

Optimization. Given the feature encoders $\mathbf{f}$ and $\mathbf{f}'$ for the different sample types, as well as the similarity measure ϕ, we introduce the shorthand $\psi(\mathbf{x},\mathbf{y}) = \phi(\mathbf{f}(\mathbf{x}), \mathbf{f}'(\mathbf{y}))$. The objective function can then be compactly written as:

$$\int_{\mathbf{x}\in \mathrm{X}} \mathrm{d}\mathbf{x}\mathrm{p}(\mathbf{x}) \left[-\int_{\mathbf{y}\in \mathrm{Y}} \mathrm{d}\mathbf{y}\mathrm{p}(\mathbf{y}|\mathbf{x})\psi(\mathbf{x},\mathbf{y}) + \log \int_{\mathbf{y}\in \mathrm{Y}} \mathrm{d}\mathbf{y}\mathrm{q}(\mathbf{y}|\mathbf{x})\mathrm{e}^{\psi(\mathbf{x},\mathbf{y})} \right]. \quad (1)$$

We approximate this objective by drawing a single positive example $\mathbf{y}_+$, and multiple negative examples $\mathbf{y}_i$ from the distributions outlined above, and minimize the loss function

$$\mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim p(\mathbf{x}),\, \mathbf{y}_+\sim p(\mathbf{y}|\mathbf{x}) \\ \mathbf{y}_1,\ldots,\mathbf{y}_n\sim q(\mathbf{y}|\mathbf{x})}} \left[-\psi(\mathbf{x},\mathbf{y}_+) + \log \sum_{i=1}^{n} e^{\psi(\mathbf{x},\mathbf{y}_i)} \right], \quad (2)$$

with a gradient-based optimization algorithm. The number of negative samples is a hyperparameter of the algorithm, and larger batch sizes are generally preferable.

For sufficiently small datasets, as used in this paper, both positive and negative samples are drawn from all available samples in the dataset. This is in contrast to the common practice in many contrastive learning frameworks in which a minibatch of samples is drawn first, which are then grouped into positive and negative pairs. Allowing access to the whole dataset to form pairs gives a better approximation of the respective distributions $p(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{y}|\mathbf{x})$, and considerably improves the quality of the obtained embeddings. If the dataset is sufficiently small to fit into the memory, CEBRA can be optimized with batch gradient descent—that is, using the whole dataset at each optimizer step.

Goodness of fit. Comparing the loss value—at both absolute and relative values across models at the same point in training time— can be used to determine goodness of fit. In practical terms, this means that one can find which hypothesis best fits one's data, in the case of using CEBRA-Behaviour. Specifically, let us denote the objective in equation(1) as L_{asympt} and its approximation in equation (2) with a batch size of n as L_n. In the limit of many samples, the objective converges up to a constant, $L_{\text{asympt}} = \lim_{n\to\infty}[L_n - \log n]$ (Supplementaty Note 2 and ref. 53).

The objective also has two trivial solutions: the first is obtained for a constant $\psi(\mathbf{x},\mathbf{y}) = \psi$, which yields a value of $L_n = \log n$. This solution can be obtained when the labels are not related to the signal (e.g., with shuffled labels). It is typically not obtained during regular training because the network is initialized randomly, causing the initial embedding points to be randomly distributed in space.

If the embedding points are distributed uniformly in space and ϕ is selected such that $\mathbb{E}[\phi(\mathbf{x},\mathbf{y})] = 0$, we will also get a value that is approximately $L_n = \log n$. This value can be readily estimated by computing $\phi(\mathbf{u},\mathbf{v})$ for randomly distributed points.

The minimizer of equation (1) is also clearly defined as $-D_{\mathrm{KL}}(p\|q)$ and depends on the positive and negative distribution. For discovery-driven (time contrastive) learning, this value is impossible to estimate because it would require access to the underlying conditional distribution of the latents. However, for training with predefined positive and negative distributions, this quantity can be again numerically estimated.

Interesting values of the loss function when fitting a CEBRA model are therefore

$$-D_{\mathrm{KL}}(p\|q) \le L_n - \log n \le 0 \quad (3)$$

where $L_n - \log n$ is the goodness of fit (lower is better) of the CEBRA model. Note that the metric is independent of the batch size used for training.

Sampling. Selection of the sampling scheme is CEBRA's key feature in regard to adapting embedding spaces to different datasets and recording setups. The conditional distributions $p(\mathbf{y}|\mathbf{x})$ for positive samples and $q(\mathbf{y}|\mathbf{x})$ for negative samples, as well as the marginal distribution $p(\mathbf{x})$ for reference samples, are specified by the user. CEBRA offers a set of predefined sampling techniques but customized variants can be specified to implement additional, domain-specific distributions. This form of training allows the use of context variables to shape the properties of the embedding space, as outlined in the graphical model in Supplementary Note 1.

Through the choice of sampling technique, various use cases can be built into the algorithm. For instance, by forcing positive and negative distributions to sample uniformly across a factor, the model will become invariant to this factor because its inclusion would yield a suboptimal value of the objective function.

When considering different sampling mechanisms we distinguish between single- and multi-session datasets: a single-session dataset consists of samples $\mathbf{s}_t$ associated to one or more context variables $\mathbf{c}_t$ and/or k_t. These context variables allow imposition of the structure on the marginal and conditional distribution used for obtaining the embedding. Multi-session datasets consist of multiple, single-session datasets. The dimension of context variables $\mathbf{c}_t$ and/or k_t must be shared across all sessions whereas the dimension of the signal $\mathbf{s}_t$ can vary. In such a setting, CEBRA allows learning of a shared embedding space for signals from all sessions.

For single-session datasets, sampling is done in two steps. First, based on a specified 'index' (the user-defined context variable $\mathbf{c}_t$ and/or k_t), locations t are sampled for reference, positive and negative samples. The algorithm differentiates between categorical (k) and continuous ($\mathbf{c}$) variables for this purpose.

In the simplest case, negative sampling (q) returns a random sample from the empirical distribution by returning a randomly chosen index from the dataset. Optionally, with a categorical context variable $k_t \in [K]$, negative sampling can be performed to approximate a uniform distribution of samples over this context variable. If this is performed for both negative and positive samples, the resulting embedding will become invariant with respect to the variable k_t. Sampling is performed in this case by computing the cumulative histogram of k_t and sampling uniformly over k using the transformation theorem for probability densities.

For positive pairs, different options exist based on the availability of continuous and discrete context variables. For a discrete context variable $k_t \in [K]$ with K possible values, sampling from the conditional distribution is done by filtering the whole dataset for the value k_t of the reference sample, and uniformly selecting a positive sample with the same value. For a continuous context variable $\mathbf{c}_t$ we can use a set of time offsets Δ to specify the distribution. Given the time offsets, the empirical distribution $P(\mathbf{c}_{t+\tau}|\mathbf{c}_t)$ for a particular choice of $\tau \in \Delta$ can be computed from the dataset: we build up a set $D = \{t \in [T], \tau \in \Delta : \mathbf{c}_{t+\tau} - \mathbf{c}_t\}$, sample a $\mathbf{d}$ uniformly from D and obtain the sample that is closest to the reference sample's context variable modified by this distance ($\mathbf{c} + \mathbf{d}$) from the dataset. It is possible to combine a continuous variable $\mathbf{c}_t$ with a categorical variable k_t for mixed sampling. On top of the continual sampling step above, it is ensured that both samples in the positive pair share the same value of k_t.

It is crucial that the context samples **c** and the norm used in the algorithm match in some way; for simple context variables with predictable conditional distributions (for example, a 1D or 2D position of a moving animal, which can most probably be well described by a Gaussian conditional distribution based on the previous sample), the positive sample distribution can also be specified directly, for example, as a normal distribution centred around $\mathbf{c}_t$. An additional alternative is to use CEBRA also to preprocess the original context samples **c** and use the embedded context samples with the metric used for CEBRA training. This scheme is especially useful for higher-dimensional behavioural data, or even for complex inputs such as video.

We next consider the multi-session case in which signals $\mathbf{s}_t^{(i)} \in \mathbb{R}^{n_i}$ come from N different sessions $i \in [N]$ with session-dependent dimensionality n_i. Importantly, the corresponding continuous context variables $\mathbf{c}_t^{(i)} \in \mathbb{R}^m$ share the same dimensionality m, which makes it possible to relate samples across sessions. The multi-session setup is similar to mixed-session sampling (if we treat the session ID as a categorical variable $k_t^{(i)} := i$ for all time steps t in session i). The conditional distribution for both negative and positive pairs is uniformly sampled across sessions, irrespective of session length. Multi-session mixed or discrete sampling can be implemented analogously.

CEBRA is sufficiently flexible to incorporate more specialized sampling schemes beyond those outlined above. For instance, mixed single-session sampling could be extended additionally to incorporate a dimension to which the algorithm should become invariant; this would add an additional step of uniform sampling with regard to this desired discrete variable (for example, via ancestral sampling).

Choice of reference, positive and negative samples. Depending on the exact application, the contrastive learning step can be performed by explicitly including or excluding the context variable. The reference sample **x** can contain information from the signal $\mathbf{s}_t$, but also from the experimental conditions, behavioural recordings or other context variables. The positive and negative samples **y** are set to the signal variable $\mathbf{s}_t$.

Theoretical guarantees for linear identifiability of CEBRA models. Identifiability describes the property of an algorithm to give a consistent estimate for the model parameters given that the data distributions match. We here apply the relaxed notion of linear identifiability that was previously discussed and used[13,14]. After training two encoder models **f** and **f′**, the models are linearly identifiable if $\mathbf{f}(\mathbf{x}) = \mathbf{L}\mathrm{f}(\mathbf{x})$, where **L** is a linear map.

When applying CEBRA, three cases are of potential interest. (1) When applying discovery-driven CEBRA, will two models estimated on comparable experimental data agree in their inferred representation? (2) Under which assumptions about the data will we be able to discover the true latent distribution? (3) In the hypothesis-driven or hybrid application of CEBRA, is the algorithm guaranteed to give a meaningful (nonstandard) latent space when we can find signal within the data?

For the first case, we note that the CEBRA objective with a cosine similarity metric follows the canonical discriminative form for which Roeder et al.[13] showed linear identifiability: for sufficiently diverse datasets, two CEBRA models trained to convergence on the same dataset will be consistent up to linear transformations. Note that the consistency of CEBRA is independent of the exact data distribution: it is merely required that the embeddings of reference samples across multiple positive pairs, and the embeddings of negative samples across multiple negative pairs, vary in sufficiently numerous linearly independent directions. Alternatively, we can derive linear identifiability from assumptions about data distribution: if the ground truth latents are sufficiently diverse (that is, vary in all latent directions under distributions p and q), and the model is sufficiently parameterized to fit the data, we will also obtain consistency up to a linear transformation. See Supplementary Note 2 for a full formal discussion and proof.

For the second case, additional assumptions are required regarding the exact form of data-generating distributions. Within the scope of this work we consider ground truth latents distributed on the hypersphere or Euclidean space. The metric then needs to match assumptions about the variation of ground truth latents over time. In discovery-driven CEBRA, using the dot product as the similarity measure then encodes the assumption that latents vary according to a von Mises–Fisher distribution whereas the (negative) mean squared error encodes an assumption that latents vary according to a normal distribution. More broadly, if we assume that the latents have a uniform marginal distribution (which can be ensured by designing unbiased experiments), the similarity measure should be chosen as the log-likelihood of conditional distribution over time. In this case, CEBRA identifies the latents up to an affine transformation (in the most general case).

This result also explains the empirically high performance of CEBRA for decoding applications: if trained for decoding (using the variable to decode for informing the conditional distribution), it is trivial to select matching conditional distributions because both quantities are directly selected by the user. CEBRA then 'identifies' the context variable up to an affine transformation.

For the third case, we are interested in hypothesis-testing capabilities. We can show that if a mapping exists between the context variable and the signal space, CEBRA will recover this relationship and yield a meaningful embedding, which is also decodable. However, if such a mapping does not exist we can show that CEBRA will not learn a structured embedding.

CEBRA models

We chose $X = Y$ as the neural signal, with varying levels of recorded neurons and channels based on the dataset. We used three types of encoder model based on the required receptive field: a receptive field of one sample was used for the synthetic dataset experiments (Fig. 1b) and a receptive field of ten samples in all other experiments (rat, monkey, mouse) except for the Neuropixels dataset, in which a receptive field of 40 samples was used due to the fourfold higher sampling rate of the dataset.

All feature encoders were parameterized by the number of neurons (input dimension), a hidden dimension used to control model size and capacity, as well as by their output (embedding) dimension. For the model with the receptive field of one, a four-layer MLP was used. The first and second layers map their respective inputs to the hidden dimension whereas the third introduces a bottleneck and maps to half the hidden dimension. The final layer maps to the requested output dimension. For the model with a receptive field of ten, a convolutional network with five time-convolutional layers was used. The first layer had a kernel size of two, and the next three had a kernel size of three and used skip connections. The final layer had a kernel size of three and mapped hidden dimensions to the output dimension. For the model with receptive field 40, we first preprocessed the signal by concatenating a 2× downsampled version of the signal with a learnable downsample operation implemented as a convolutional layer with kernel size four and stride two, directly followed (without activation function between) by another convolutional layer with kernel size three and stride two. After these first layers, the signal was subsampled by a factor of four. Afterwards, similar to the receptive field ten model, we applied three layers with kernel size three and skip connections and a final layer with kernel size three. In all models, Gaussian error linear unit activation functions[54] were applied after each layer except the last. The feature vector was normalized after the last layer unless a mean squared error-based similarity metric was used (as shown in Extended Data Fig. 8).

Our implementation of the InfoNCE criterion received a minibatch (or the full dataset) of size $n \times d$ for each of the reference, positive and negative samples. n dot-product similarities were computed between reference and positive samples and $n \times n$ dot-product similarities were computed between reference and negative samples. Similarities were scaled with the inverse of the temperature parameter τ:

```
from torch import einsum, logsumexp, no_grad

def info_nce(ref, pos, neg, tau = 1.0):
    pos_dist = einsum("nd,nd->n", ref, pos)/tau
    neg_dist = einsum("nd,md->nm", ref, neg)/tau
    with no_grad():
        c, _ = neg_dist.max(dim=1)
    pos_dist = pos_dist – c.detach()
    neg_dist = neg_dist – c.detach()
    pos_loss = –pos_dist.mean()
    neg_loss = logsumexp(neg_dist, dim = 1).mean()
    return pos_loss + neg_loss
```

Alternatively, a learnable temperature can be used. For a numerically stable implementation we store the log inverse temperature $\alpha = -\log\tau$ as a parameter of loss function. At each step we scale the distances in loss function with $\min(\exp\alpha,\ 1/\tau_{\min})$. The additional parameter $\tau_{\min}$ is a lower bound on the temperature. The inverse temperature used for scaling the distances in the loss will hence lie in $(0, 1/\tau_{\min}]$.

CEBRA model parameters used. In the main figures we have used the default parameters (https://cebra.ai/docs/api.html) for fitting CEBRA unless otherwise stated in the text (such as dimension, which varied and is noted in figure legends), or below:

Synthetic data: model_architecture= 'offset1-model-mse', conditional= 'delta', delta=0.1, distance= 'euclidean', batch_size=512, learning_rate=1e-4.

Rat hippocampus neural data: model_architecture= 'offset10-model', time_offsets=10, batch_size=512.

Rat behavioural data: model_architecture= 'offset10-model-mse', distance= 'euclidean', time_offsets=10, batch_size=512.

Primate S1 neural data: model_architecture= 'offset10-model', time_offsets=10, batch_size=512.

Allen datasets (2P): model_architecture= 'offset10-model', time_offsets=10, batch_size=512.

Allen datasets (NP): model_architecture= 'offset40-model-4x-subsample', time_offsets=10, batch_size=512.

CEBRA API and example usage. The Python implementation of CEBRA is written in PyTorch[55] and NumPy[56] and provides an application programming interface (API) that is fully compatible with scikit-learn[57], a package commonly used for machine learning. This allows the use of scikit-learn tools for hyperparameter selection and downstream processing of the embeddings—for example, decoding. CEBRA can be used as a dropin replacement in existing data pipelines for algorithms such as *t*-SNE, UMAP, PCA or FastICA. Both CPU and GPU implementations are available.

Using the previously introduced notations, suppose we have a dataset containing signals $\mathbf{s}_t$, continuous context variables $\mathbf{c}_t$ and discrete context variables k_t for all time steps t,

```
import numpy as np
N = 500
s = np.zeros((N, 55), dtype = float)
k = np.zeros((N,), dtype = int)
c = np.zeros((N, 10), dtype = float)
```

along with a second session of data,

```
s2 = np.zeros((N, 75), dtype = float)
c2 = np.zeros((N, 10), dtype = float)
assert c2.shape[1] == c.shape[1]:
```

note that both the number of samples and the dimension in $\mathbf{s}'$ does not need to match $\mathbf{s}$. Session alignment leverages the fact that the second dimensions of $\mathbf{c}$ and $\mathbf{c}'$ match. With this dataset in place, different variants of CEBRA can be applied as follows:

```
import cebra
model = cebra.CEBRA
    (output_dimension=8,
    num_hidden_units=32,
    batch_size=1024,
    learning_rate=3e-4,
    max_iterations=1000)
```

The training mode to use is determined automatically based on what combination of data is passed to the algorithm:

```
# time contrastive learning
model.fit(s)
# discrete behaviour contrastive learning
model.fit(s, k)
# continuous behaviour contrastive learning
model.fit(s, c)
# mixed behaviour contrastive learning
model.fit(s, c, k)
# multi-session training
model.fit([s, s2], [c, c2])
# adapt to new session
model.fit(s, c)
model.fit(s2, c2, adapt = True)
```

Because CEBRA is a parametric method training a neural network internally, it is possible to embed new data points after fitting the model:

```
s_test = np.zeros((N, 55), dtype=float)
# obtain and plot embedding
z = model.transform(s_test)
plt.scatter(z[:, 0], z[:, 1])
plt.show()
```

Besides this simple-to-use API for end users, our implementation of CEBRA is a modular software library that includes a plugin system, allowing more advanced users to readily add additional model implementations, similarity functions, datasets and data loaders and distributions for sampling positive and negative pairs.

Consistency of embeddings across runs, subjects, sessions, recording modalities and areas

To measure the consistency of the embeddings we used the R^2 score of linear regression (including an intercept) between embeddings from different subjects (or sessions). Secondly, pi-VAE, which we benchmarked and improved (Extended Data Fig. 1), demonstrated a theoretical guarantee that it can reconstruct the true latent space up to an affine transformation. Across runs, we measured the R^2 score of linear regression between embeddings across ten runs of the algorithms, yielding 90 comparisons. These runs were done with the same hyperparameters, model and training setup.

For the rat hippocampus data, the numbers of neurons recorded were different across subjects. The behaviour setting was the same: the rats moved along a 1.6-meter-long track and, for analysis, behaviour data were binned into 100 bins of equal size for each direction (leftwards, rightwards). We computed averaged feature vectors for each bin by averaging all normalized CEBRA embeddings for a given bin and renormalized the average to lie on the hypersphere. If a bin did not contain any sample, it was filled by samples from the two adjacent bins. CEBRA was trained with latent dimension three (the minimum) such that it was constrained to lie only on a two-sphere (making this '3D' space equivalent to a 2D Euclidean space). All other methods were trained with two latent dimensions in Euclidean space. Note that $n + 1$ dimensions of CEBRA are equivalent to n dimensions of other methods that we compared, because the feature space of CEBRA is normalized (that is, the feature vectors are normalized to have unit length).

For Allen visual data in which the number of behavioural data points is the same across different sessions (that is, fixed length of video stimuli), we directly computed the R^2 score of linear regression between embeddings from different sessions and modalities. We surveyed three, four, eight, 32, 64 and 128 latent dimensions with CEBRA.

To compare the consistency of embeddings between or within the areas considered, we computed intra- and interarea consistency within the same recording modality (2P or NP). Within the same modality we sampled 400 neurons from each area. We trained one CEBRA model per area and computed linear consistency between all pairs of embeddings. For intra-area comparison we sampled an additional 400 disjoint neurons. For each area we trained two CEBRA models on these two sets of neurons and computed their linear consistency. We repeated this process three times.

For comparisons across modalities (2P and NP) we sampled 400 neurons from each modality (which are disjoint, as above, because one set was sampled from 2P recordings and the other from NP recordings). We trained a multi-session CEBRA model with one encoder for 2P and one for NP in the same embedding space. For intra-area comparison we computed linear consistency between NP and 2P decoders from the same area. For interarea comparison we computed linear consistency between the NP encoder from one area and the 2P encoder from another and again considered all combinations of areas. We repeated this process three times.

For comparison of single- and multi-session training (Extended Data Fig. 7) we computed embeddings using encoder models with eight, 16, ..., 128 hidden units to vary the model size, and benchmarked eight, 16, ..., 128 latent dimensions. Hyperparameters, except for number of optimization steps, were selected according to either validation set decoding R^2 (rat) or accuracy (Allen). Consistency was reported as the point in training at which position decoding error was less than 7 cm for the first rat in the hippocampus dataset, and a decoding accuracy of 60% in the Allen dataset. For single-session training, four embeddings were trained independently on each individual animal whereas for multi-session training the embeddings were trained jointly on all sessions. For multi-session training, the same number of samples was drawn from each session to learn an embedding invariant to the session ID. The consistency versus decoding error trade-off (Extended Data Fig. 7c) was reported as the average consistency across all 12 comparisons (Extended Data Fig. 7b) versus average decoding performance across all rats and data splits.

Model comparisons

pi-VAE parameter selection and modifications to pi-VAE. Because the original implementation of pi-VAE used a single time bin spiking rate as an input, we therefore modified their code to allow for larger time bin inputs and found that time window input with a receptive field of ten time bins (250 ms) gave higher consistency across subjects and better preserved the qualitative structure of the embedding (thereby outperforming the results presented by Zhou and Wei[5]; Extended Data Fig. 1). To do this we used the same encoder neural network architecture as that for CEBRA and modified the decoder to a 2D output (we call our modified version conv-pi-VAE). Note, we used this modified pi-VAE for all experiments except for the synthetic setting, for which there is no time dimension and thus the original implementation is sufficient.

The original implementation reported a median absolute error of 12 cm for rat 1 (the individual considered most in that work), and our implementation of time-windowed input with ten bins resulted in a median absolute error of 11 cm (Fig. 2). For hyperparameters we tested different epochs between 600 (the published value used) and 1,000, and learning rate between 1.0×10^{-6} and 5.0×10^{-4} via a grid search. We fixed hyperparameters as those that gave the highest consistency across subjects, which were training epochs of 1,000 and learning rate 2.5×10^{-4}. All other hyperparameters were retained as in the original implementation[5]. Note that the original paper demonstrated that pi-VAE is fairly robust across different hyperparameters. For decoding (Fig. 2) we considered both a simple kNN decoder (that we use for CEBRA) and the computationally more expensive Monte Carlo sampling method originally proposed for pi-VAE[5]. Our implementation of conv-pi-VAE can be found at https://github.com/AdaptiveMotorControlLab/CEBRA.

autoLFADS parameter selection. AutoLFADS[25] includes a hyperparameter selection and tuning protocol, which we used, and we also used the original implementation (https://github.com/snel-repo/autolfads-tf2/, https://github.com/neurallatents/nlb_tools/tree/main/examples/baselines/autolfads). For the rat hippocampus dataset we chopped the continuous spiking rate (25 ms bin size) into 250-ms-long segments with 225 ms overlap between segments to match the training setup for CEBRA, UMAP, *t*-SNE and pi-VAE. We used population-based training (PBT) for hyperparameter searches and constrained the search range to default values given in the original script (initial learning rate between 1.0×10^{-5} and 5.0×10^{-3}, dropout rate 0–0.6, coordinated dropout rate 0.01–0.70, L2 generator weight between 1.0×10^{-4} and 1.0, L2 controller weight between 1.0×10^{-4} and 1.0, KL controller weight between 1.0×10^{-6} and 1.0×10^{-4} and KL initial condition weight between 1.0×10^{-6} and 1.0×10^{-3}). The negative log-likelihood metric was used to select the best hyperparameters. Each generation of PBT consisted of 25 training epochs and we trained for a maximum of 5,000 epochs of batch size 100 while executing early stopping after awaiting 50 epochs. The PBT search was done using 20 parallel workers on each rat.

UMAP parameter selection. For UMAP[11], following the parameter guide (umap-learn.readthedocs.io/), we focused on tuning the number of neighbours (*n_neighbors*) and minimum distance (*min_dist*). The *n_components* parameter was fixed to 2 and we used a cosine metric to make a fair comparison with CEBRA, which also used the cosine distance metric for learning. We performed a grid search with 100 total hyperparameter values in the range [2, 200] for *n_neighbors* and in the range [0.0001, 0.99] for *min_dist*. The highest consistency across runs in the rat hippocampus dataset was achieved with *min_dist* of 0.0001 and *n_neighbors* of 24. For the other datasets in Extended Data Fig. 3 we used the default value of *n_neighbors* as 15 and *min_dist* as 0.1.

***t*-SNE parameter selection.** For *t*-SNE[12] we used the implementation in openTSNE[58]. We performed a sweep on *perplexity* in the range [5, 50] and *early_exaggeration* in the range [12, 32] following the parameter guide, while fixing *n_components* as 2 and used a cosine metric for fair comparison with UMAP and CEBRA. We used PCA initialization to improve the run consistency of *t*-SNE[59]. The highest consistency across runs in the rat hippocampus dataset was achieved with *perplexity* of ten and *early_exaggeration* of 16.44. For the other datasets in Extended Data Fig. 3 we used the default value for *perplexity* of 30 and for *early_exaggeration* of 12.

Decoding analysis

We primarily used a simple kNN algorithm, a nonparametric supervised learning method, as a decoding method for CEBRA. We used the implementation in scikit-learn[57]. We used a kNN regressor for continuous value regression and a kNN classifier for discrete label classification. For embeddings obtained with cosine metrics we used cosine distance metrics for kNN, and Euclidean distance metrics for those obtained in Euclidean space.

For the rat hippocampus data a kNN regressor, as implemented in scikit-learn[57], was used to decode position and a kNN classifier to decode direction. The number of neighbours was searched over the range [1, 4, 9, 16, 25] and we used the cosine distance metric. We used the R^2 score of predicted position and direction vector on the validation set as a metric to choose the best *n_neighbours* parameter. We report the median absolute error for positional decoding on the test set. For pi-VAE, we additionally evaluated decoding quality using the originally proposed decoding method based on Monte Carlo sampling, with the settings from the original article[5]. For autoLFADS, use of their default Ridge regression decoder[25] performed worse than our kNN decoder, which is why we reported all results for the kNN decoder. Note that UMAP, *t*-SNE and CEBRA-Time were trained using the full dataset without label information when learning the embedding, and we used the above split only for training and cross-validation of the decoder.

For direction decoding within the monkey dataset we used a Ridge classifier[57] as a baseline. The regularization hyperparameter was searched over $[10^{-6}, 10^{2}]$. For CEBRA we used a kNN classifier for decoding direction with *k* searched over the range [1, 2500]. For conv-pi-VAE we searched for the best learning rate over $[1.0 \times 10^{-5}, 1.0 \times 10^{-3}]$. For position decoding we used Lasso[57] as a baseline. The regularization hyperparameter was searched over $[10^{-6}, 10^{2}]$. For conv-pi-VAE we used 600 epochs and searched for the best learning rates over $[5 \times 10^{-4}, 2.5 \times 10^{-4}, 0.125 \times 10^{-4}, 5 \times 10^{-5}]$ via a grid of (x,y) space in 1 cm bins for each axis as the sampling process for decoding. For CEBRA we used kNN regression, and the number of neighbours *k* was again searched over [1, 2500].

For the Allen Institute datasets we performed decoding (frame number or scene classification) for each frame from Video 1. Here we used a kNN classifier[57] with a population vector kNN as a baseline, similar to the decoding of orientation grating performed in ref. 46. For CEBRA we used the same kNN classifier method as on CEBRA features. In both cases the number of neighbours, *k*, was searched over a range [1, 100] in an exponential fashion. We used neural data recorded during the first eight repeats as the train set, the ninth repeat for validation in choosing the hyperparameter and the last repeat as the test set to report decoding accuracy. We also used a Gaussian naive Bayes decoder[57] to test linear decoding from the CEBRA model and neural population vector. Here we assumed uniform priors over frame number and searched over the range $[10^{-10}, 10^{3}]$ in an exponential manner for the *var_smoothing* hyperparameter.

For layer-specific decoding we used data from excitatory neurons in area VISp: layers 2/3 [Emx1-IRES-Cre, Slc17a7-IRES2-Cre]; layer 4 [Cux2-CreERT2, Rorb-IRES2-Cre, Scnn1a-Tg3-Cre]; and layers 5/6 [Nr5a1-Cre, Rbp4-Cre_KL100, Fezf2-CreER, Tlx3-Cre_PL56, Ntrsr1-cre].

Neural Latents Benchmark. We tested CEBRA on the *mc-maze* 20 ms task from the Neural Latents Benchmark[37] (https://eval.ai/web/challenges/challenge-page/1256/leaderboard/3183). We trained the offset10-model with 48 output dimensions and [128, 256, 512] hidden units, as presented throughout the paper. We trained, in total, 48 models by additionally varying the temperature in [0.0001, 0.004] and time offsets from {1, 2}. We performed smoothing of input neural data using a Gaussian kernel with 50 ms s.d. Lastly, we took 45 embeddings from the trained models picked by the validation score, aligned the embeddings (using the Procrustes method[60]) and averaged them.

Topological analysis

For the persistent cohomology analysis we utilized ripser.py[61]. For the hippocampus dataset we used 1,000 randomly sampled points from CEBRA-Behaviour trained with temperature 1, time offset 10 and minibatch size 512 for 10,000 training steps on the full dataset and then analysed up to 2D cohomology. Maximum distance considered for filtration was set to infinity. To determine the number of cocycles in each cohomology dimension with a significant lifespan we trained 500 CEBRA embeddings with shuffled labels, similar to the approach taken in ref. 33. We took the maximum lifespan of each dimension across these 500 runs as a threshold to determine robust Betti numbers, then surveyed the Betti numbers of CEBRA embeddings across three, eight, 16, 32 and 64 latent dimensions.

Next we used DREiMac[62] to obtain topology-preserving circular coordinates (radial angle) of the first cocycle (H^1) from the persistent cohomology analysis. Similar to above, we used 1,000 randomly sampled points from the CEBRA-Behaviour models of embedding dimensions 3, 8, 16, 32 and 64.

Behaviour embeddings for video datasets

High-dimensional inputs, such as videos, need further preprocessing for effective use with CEBRA. First we used the recently presented DINO model[48] to embed video frames into a 768D feature space. Specifically we used the pretrained ViT/8 vision transformer model, which was trained by a self-supervised learning objective on the ImageNet database. This model is particularly well suited for video analysis and among the state-of-the-art models available for embedding natural images into a space appropriate for a kNN search[48], a desired property when making the dataset compatible with CEBRA. We obtained a normalized feature vector for each video frame, which was then used as the continuous behaviour variable for all further CEBRA experiments.

For scene labels, three individuals labelled each video frame using eight candidate descriptive labels allowing multilabel classes. We took the majority vote of these three individuals to determine the label of each frame. In the case of multilabels we considered this as a new class label. The above procedure resulted in ten classes of frame annotation.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

Hippocampus dataset: https://crcns.org/data-sets/hc/hc-11/about-hc-11, and we used the preprocessing script from https://github.com/zhd96/pi-vae/blob/main/code/rat_preprocess_data.py. Primate dataset: https://gui.dandiarchive.org/#/dandiset/000127. Allen Institute dataset: Neuropixels data are available at https://allensdk.readthedocs.io/en/latest/visual_coding_neuropixels.html. The preprocessed 2P recordings are available at https://github.com/zivlab/visual_drift/tree/main/data.

Code availability

Code: https://github.com/AdaptiveMotorControlLab/CEBRA. Code used to reproduce the figures: https://github.com/AdaptiveMotorControlLab/CEBRA-figures.

51. Dinh, L., Sohl-Dickstein, J. N. & Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations* https://openreview.net/pdf?id=HkpbnH9lx (2017).
52. Deitch, D., Rubin, A. & Ziv, Y. Representational drift in the mouse visual cortex. *Curr. Biol.* **31**, 4327–4339 (2021).
53. Wang, T. & Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. *Proc. Mach. Learn. Res.* **119**, 9929–9939 (2020).
54. Hendrycks, D. & Gimpel, K. Gaussian error linear units (GELUs). Preprint at https://doi.org/10.48550/arXiv.1606.08415 (2016).
55. Paszke, A. et al. PyTorch: an imperative style, high-performance deep learning library. *Neural Inf. Process. Syst.* **32**, 8024–8035 (2019).
56. Walt, S. V., Colbert, S. C. & Varoquaux, G. The NumPy array: a structure for efficient numerical computation. *Comput. Sci. Eng.* **13**, 22–30 (2011).
57. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
58. Policar, P. G., Stražar, M. & Zupan, B. openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. Preprint at *bioRxiv* https://doi.org/10.1101/731877 (2019).
59. Kobak, D. & Linderman, G. C. Initialization is critical for preserving global data structure in both *t*-SNE and UMAP. *Nat. Biotechnol.* **39**, 156–157 (2021).

60. Schönemann, P. H. A generalized solution of the orthogonal procrustes problem. *Psychometrika* **31**, 1–10 (1966).
61. Tralie, C. J., Saul, N. & Bar-On, R. Ripser.py: a lean persistent homology library for Python. *J. Open Source Softw.* **3**, 925 (2018).
62. Tralie, C. J., Mease, T. & Perea, J. Dreimac: dimension reduction with Eilenberg–Maclane coordinates. GitHub https://github.com/ctralie/DREiMac/tree/cdd6d02ba53c3597a931db9da478fd198d6ed00f (2018).

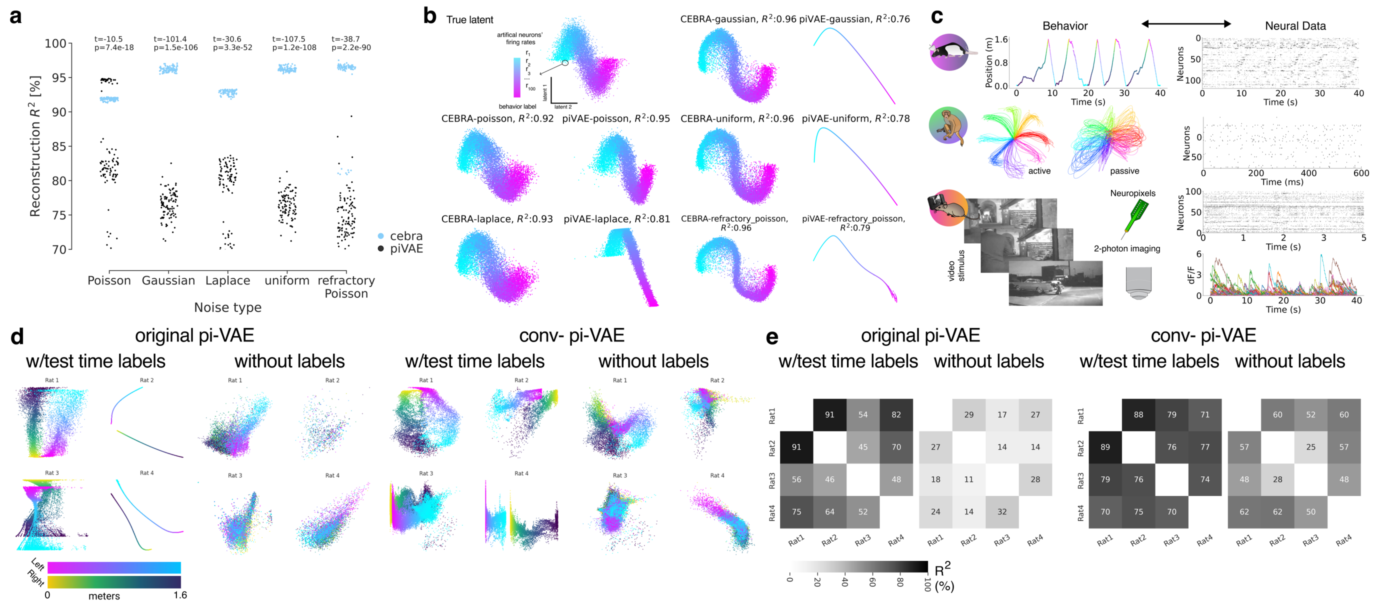
Acknowledgements Funding was provided by SNSF grant no. 310030_201057, The Vallee Foundation and a Novartis Foundation for Medical-Biological Research Young Investigator Grant to M.W.M., a Google PhD Fellowship to S.S. and the German Academic Exchange Service to J.H.L. S.S. acknowledges the IMPRS-IS Tübingen and ELLIS PhD programme, and J.H.L. thanks the TUM Programme in Neuroengineering. M.W.M. is the Bertarelli Foundation Chair of Integrative Neuroscience. We thank L. Eck for help with theory; A. Mathis for advice on topological data analysis; M. Bethge, R. Zimmerman, D. Paiton and W. Brendel for general discussions; C. Benquet, R. González and J. Lauer for feedback on code; G. Kane for advice on statistical analysis; and M. Frey, D. Kobak and J. Macke for comments on the manuscript. The funders had no role in the conceptualization, design, data collection, analysis, decision to publish or preparation of the manuscript.

Author contributions M.W.M. and S.S. conceptualized the project. S.S., J.H.L. and M.W.M. were responsible for the methodology. S.S., J.H.L. and M.W.M. were responsible for the software. S.S. was responsible for the theory. S.S. and J.H.L. undertook the formal analysis. S.S. and J.H.L. did the investigation. J.H.L., S.S. and M.W.M. were responsible for data curation. M.W.M. wrote the original draft. M.W.M., S.S. and J.H.L. wrote and edited the final version of the article.

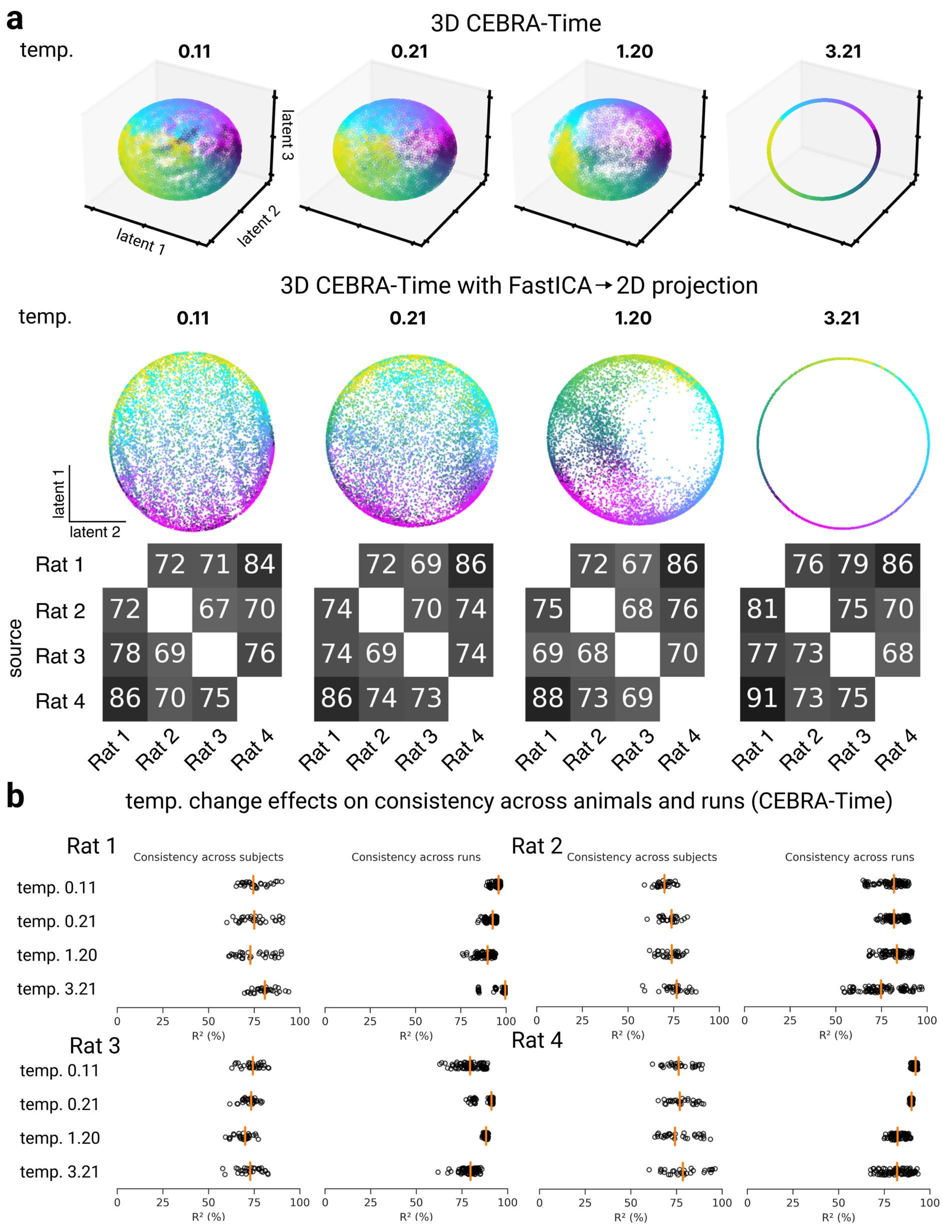
Funding Open access funding provided by EPFL Lausanne.

Competing interests S.S. and M.W.M. have filed a patent pertaining to the method presented in this work (filing no. 63/302,670). The remaining author declares no competing interests.

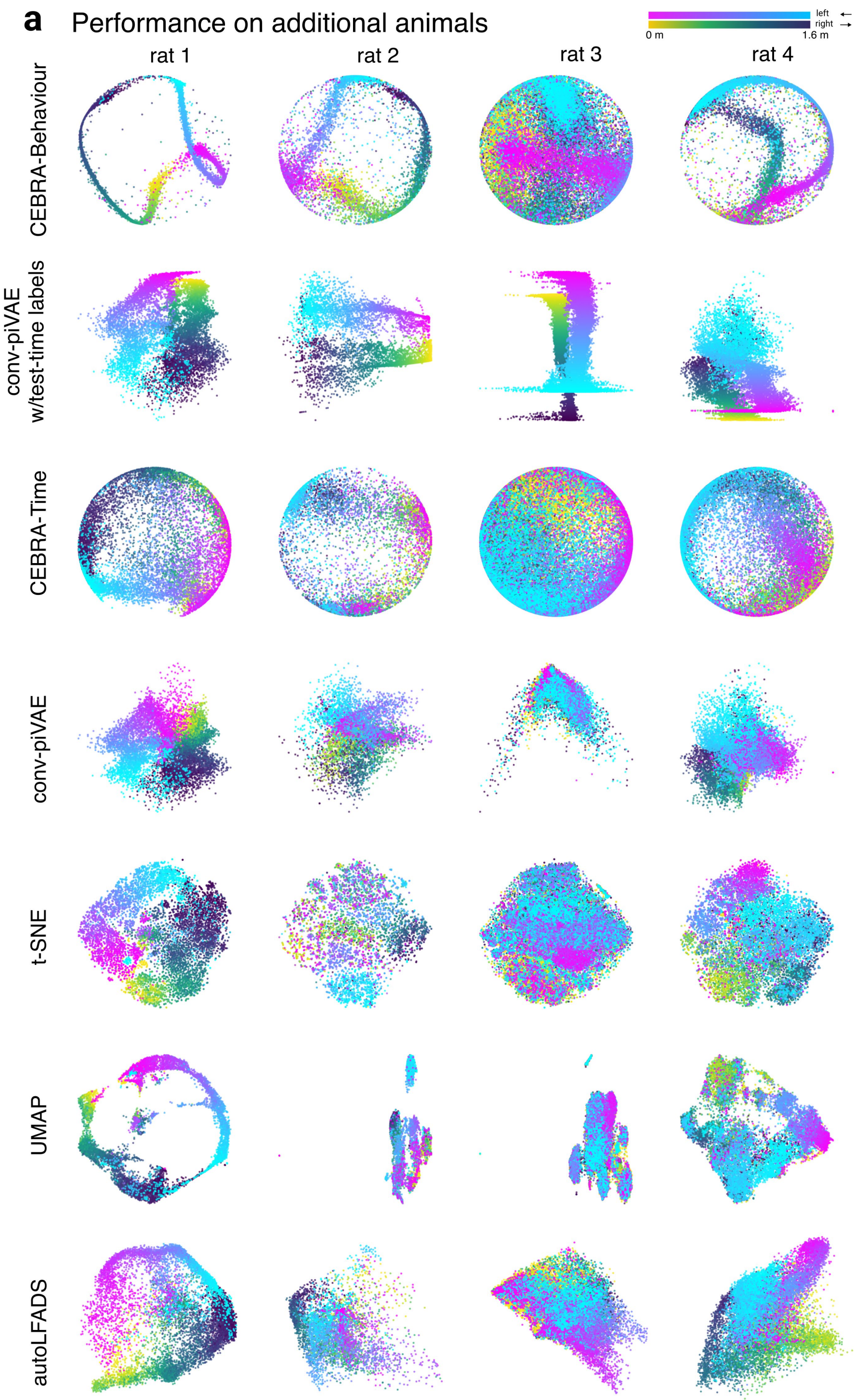
Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41586-023-06031-6.
Correspondence and requests for materials should be addressed to Mackenzie Weygandt Mathis.
Peer review information *Nature* thanks Anne Churchland, Benjamin Dunn and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



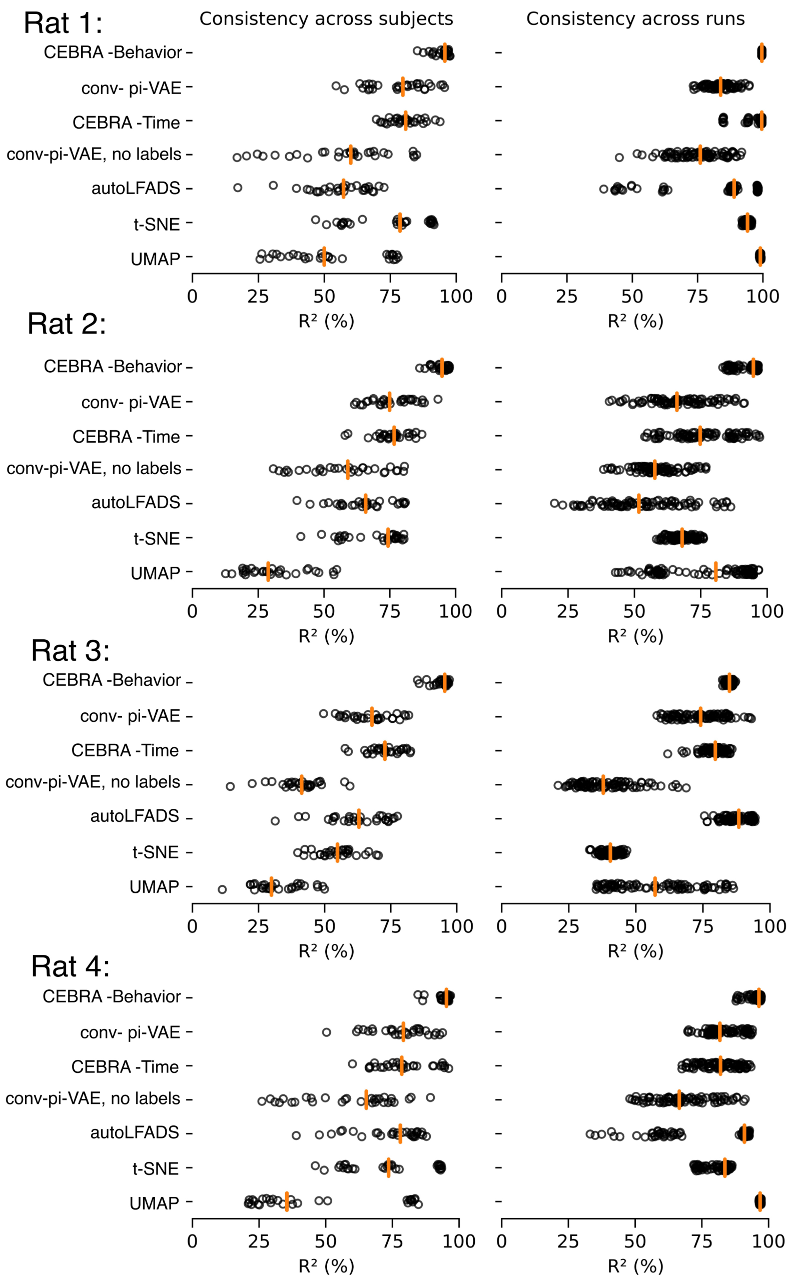
Extended Data Fig. 1 | Overview of datasets, synthetic data, & original pi-VAE implementation vs. modified conv-pi-VAE. a, We generated synthetic datasets similar to Fig. 1b with additional variations in the noise distributions in the generative process. We benchmarked the reconstruction score of the true latent using CEBRA and pi-VAE (n = 100 seeds) on the generated synthetic datasets. CEBRA showed higher and less variable reconstruction scores than pi-VAE in all noise types (one-sided Welch's t-test, corrected using the Holm-Bonferroni method, t and p-values indicated on the plot). **b**, Example visualization of the reconstructed latents from CEBRA and pi-VAE on different synthetic dataset types. **c**, We benchmarked and demonstrate the abilities of CEBRA on four datasets. Rat-based electrophysiology data[26], where the animal transverse a 1.6 meter linear track "leftwards" or "rightwards". Two mouse-based datasets: one 2-photon calcium imaging passively viewing dataset[46], and one with the same stimulus but recorded with Neuropixels[47]. A monkey-based electrophysiology dataset of centre out reaching from Chowdhury et al.[34], and processed to trial data as in ref. 52. **d**, Conv-pi-VAE showed improved performance, both with labels (Wilcoxon signed-rank test, $P = 0.0341$) and without labels Wilcoxon signed-rank test, $P = 0.0005$). Example runs/embeddings the consistency across rats, with **e**, consistency across rats, from target to source, as computed in Fig. 1. Cartoon animals are adapted from scidraw.io.



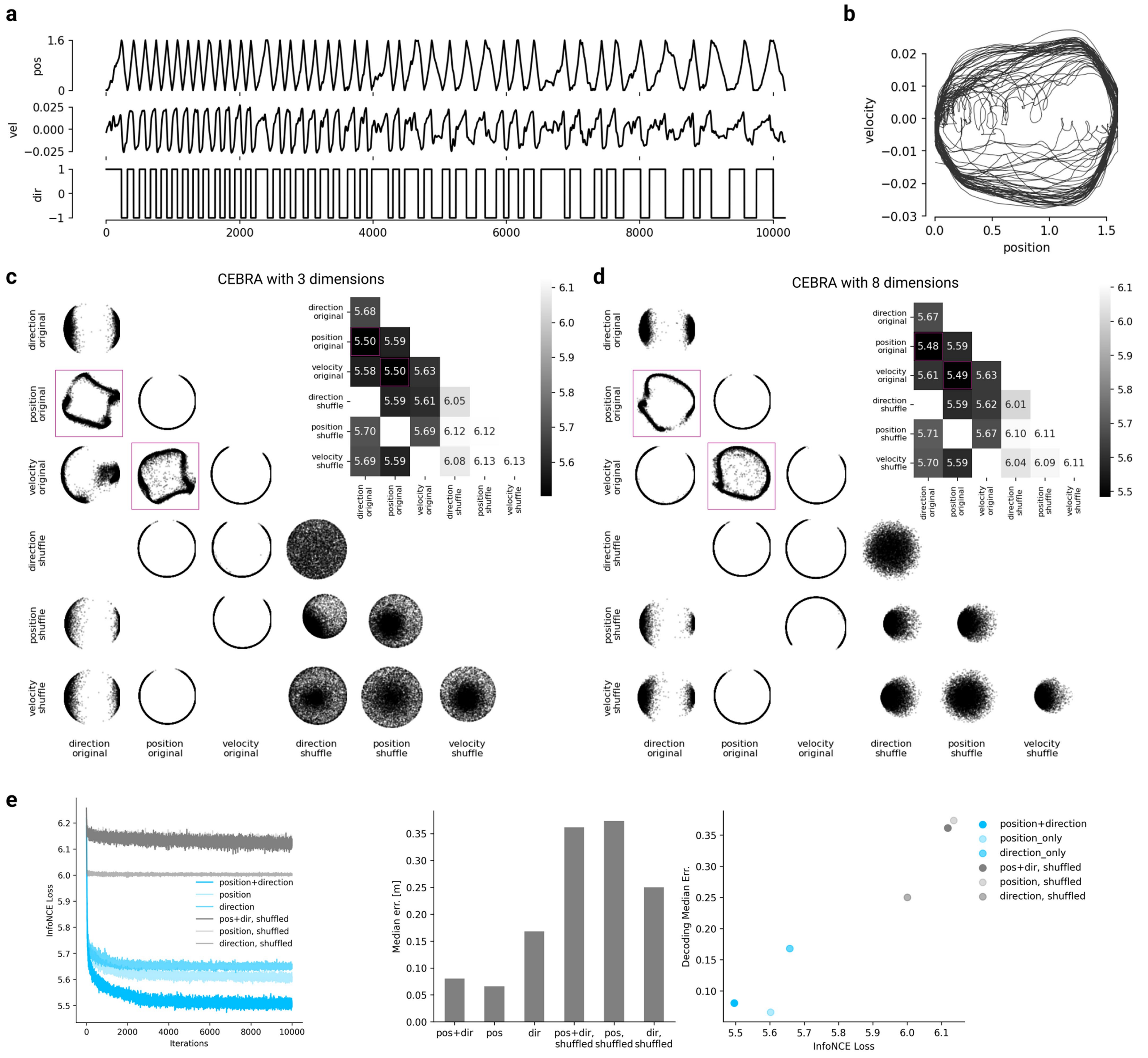
Extended Data Fig. 2 | Hyperparameter changes on visualization and consistency. a, Temperature has the largest effect on visualization (vs. consistency) of the embedding as shown by a range from 0.1 to 3.21 (highest consistency for rat 1), as can be appreciated in 3D (top) and post FastICA into a 2D embedding (middle). Bottom row shows the corresponding change on mean consistency, and in **b**, the variance can be noted. Orange line denotes the median and black dots are individual runs (subject consistency: 10 runs with 3 comparisons per rat; run consistency: 10 runs, each compared to 9 remaining runs).



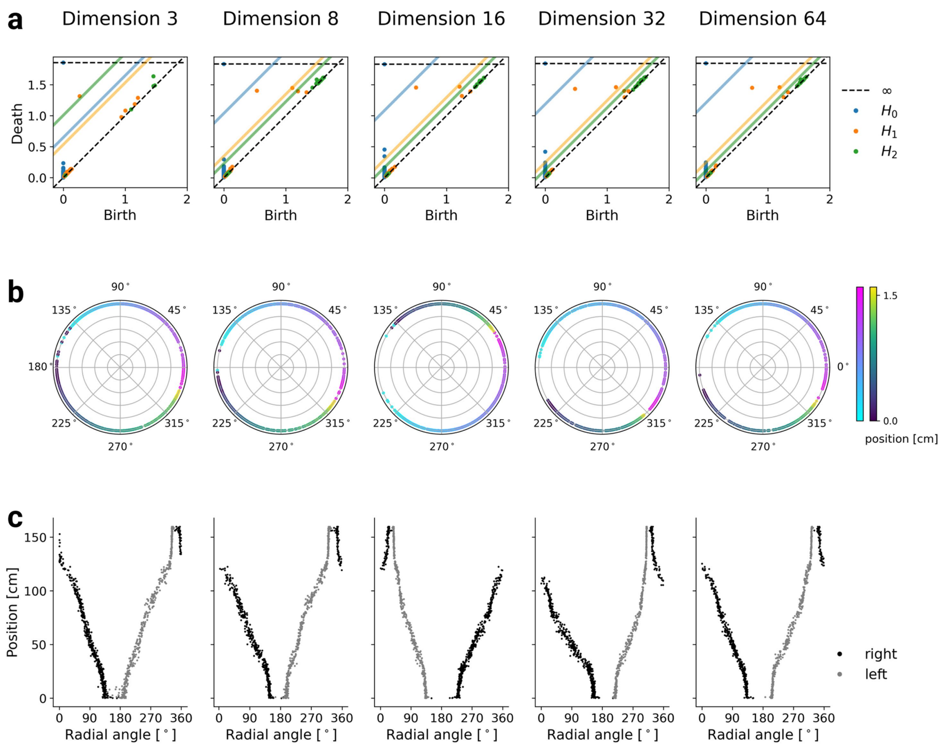
Extended Data Fig. 3 | CEBRA produced consistent, highly decodable embeddings. a, Additional rat data shown for all algorithms we benchmarked (see Methods). For CEBRA-Behaviour, we used temperature 1, time offset 10, batch size 512 and 10k training steps. For CEBRA-Time, we used temperature 2.25, time offset 10, batch size 512 and 4k training steps. For UMAP, we used the cosine metric and *min_dist* of 0.99 and *n_neighbors* of 31. For t-SNE we used cosine metric and *perplexity* of 29. For conv-pi-VAE, we trained 1000 epochs with learning rate 2.5×10^{-4}. For autoLFADS we used the in-built ray-tune framework for finding optimal hyperparameters. CEBRA was trained with output latent 3D (the minimum) and all other methods were trained with a 2D latent.



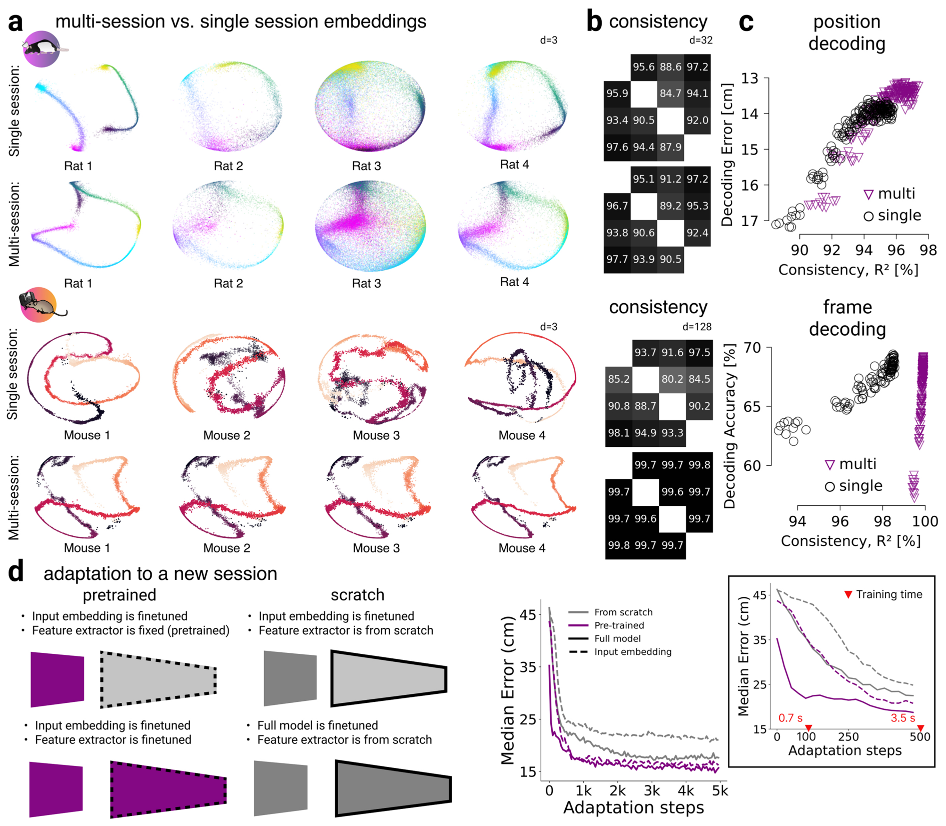
Extended Data Fig. 4 | Additional metrics used for benchmarking consistency. Comparisons of all algorithms along different metrics for rats 1, 2, 3, 4. The orange line is median across n = 10 runs, black circles denote individual runs. Each run is the average over three non-overlapping test splits.



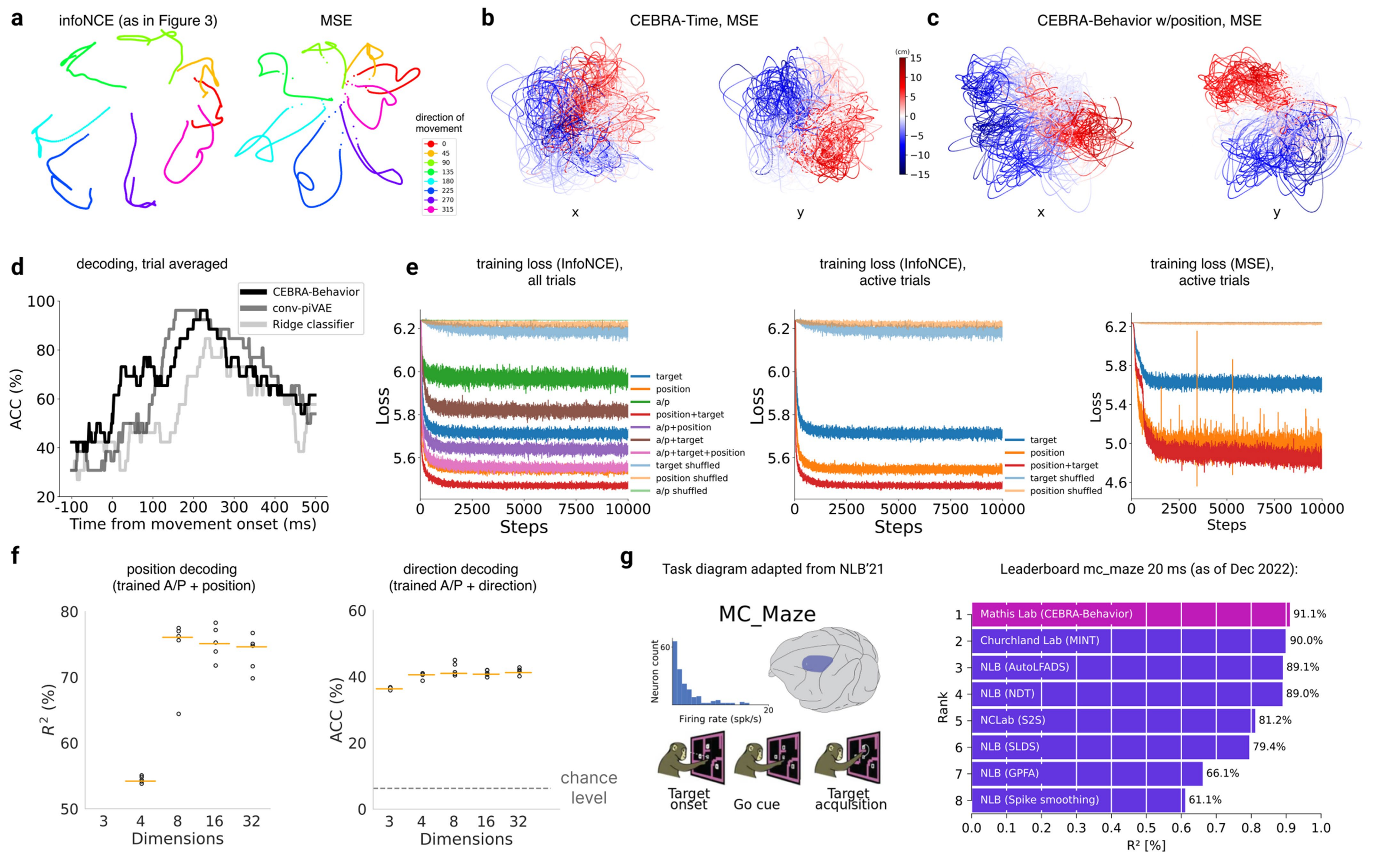
Extended Data Fig. 5 | Hypothesis testing with CEBRA. a, Example data from a hippocampus recording session (rat 1). We tested possible relationships between three experimental variables (rat location, velocity, movement direction) and the neural recordings (120 neurons, not shown). **b**, Relationship between velocity and position. **c**, We trained CEBRA with three-dimensional outputs on every single experimental variable (main diagonal) and every combination of two variables. All variables are treated as 'continuous' in this experiment. We compared original to shuffled variables (shuffling is done by permuting all samples over the time dimension) as a control. We projected the original three-dimensional space onto the first principal components. We show the minimum value of the InfoNCE loss on the trained embedding for all combinations in the confusion matrix (lower number is better). Either velocity or direction, paired with position information is needed for maximum structure in the embedding (highlighted, coloured), yielding lowest InfoNCE error. **d**, Using an eight-dimensional CEBRA embedding did not qualitatively alter the results. We again report the first two principal components as well as InfoNCE training error upon convergence, and find non-trivial embeddings with lowest training error for combinations of direction/velocity and position. **e**, The InfoNCE metric can serve as the goodness of fit metric, both for hypothesis testing and identifying decodable embeddings. We trained CEBRA in discovery-driven mode with 32 latent dimensions. We compared the InfoNCE loss (left, middle) between various hypotheses. Low InfoNCE was correlated with low decoding error (right).



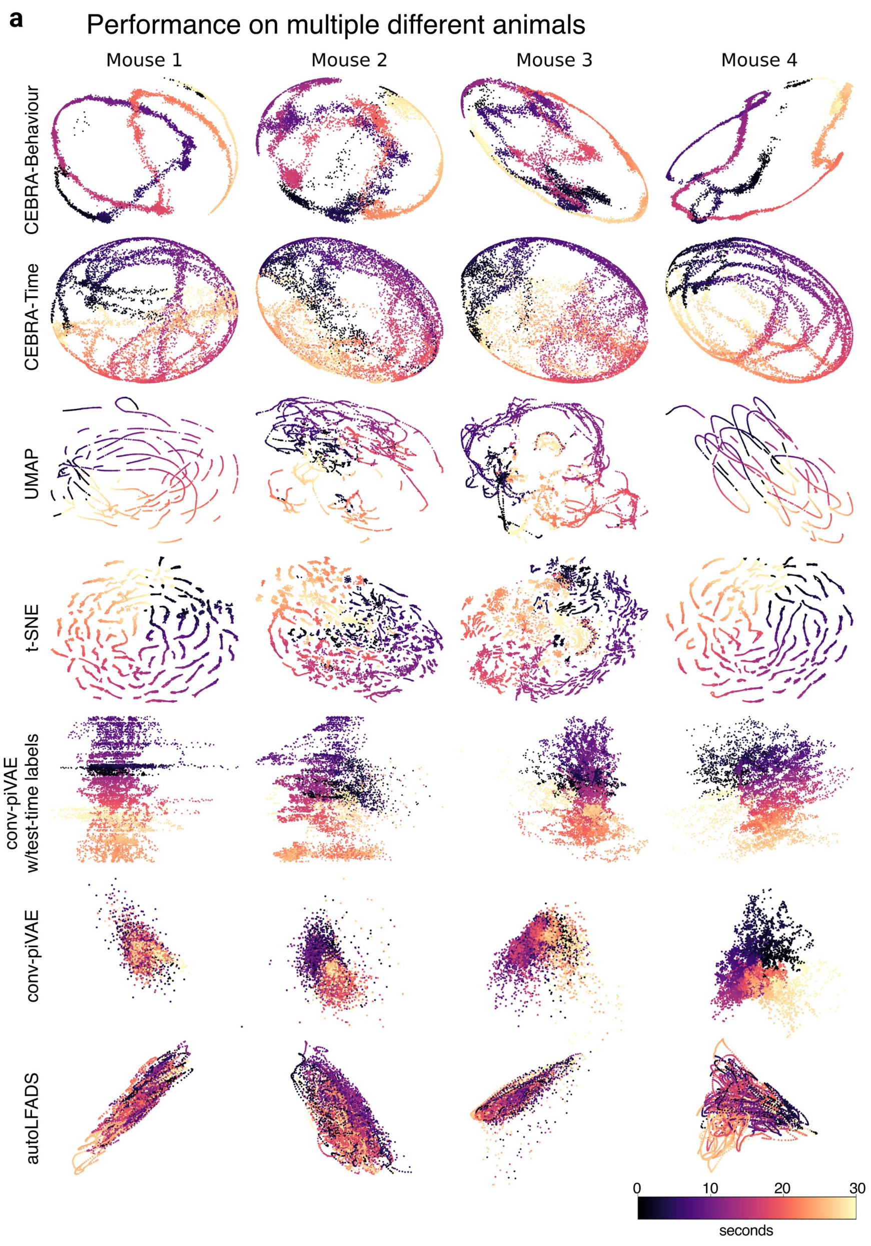
Extended Data Fig. 6 | Persistence across dimensions. a, For each dimension of CEBRA-Behaviour embedding from the rat hippocampus dataset Betti numbers were computed by applying persistent cohomology. The coloured dots are lifespans observed in hypothesis based CEBRA-Behaviour. To rule out noisy lifespans, we set a threshold (coloured diagonal lines) as maximum lifespan based on 500 seeds of shuffled-CEBRA embedding for each dimension. **b**, The topology preserving circular coordinates using the first co-cycle from persistent cohomology analysis on the CEBRA embedding of each dimension is shown (see Methods). The colours indicate position and direction of the rat at the corresponding CEBRA embedding points. **c**, The radial angle of each embedding point obtained from **b** and the corresponding position and direction of the rat.



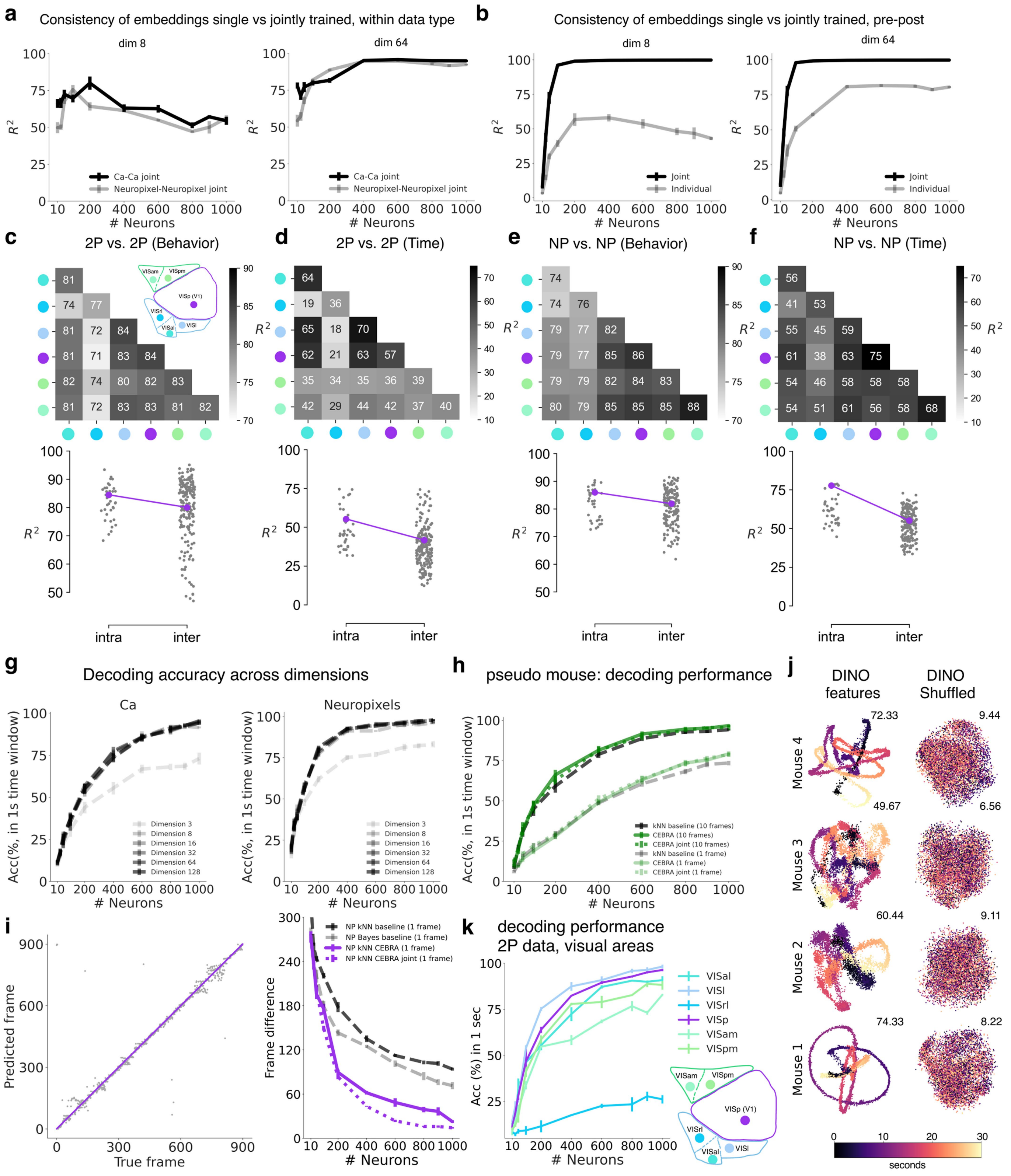
Extended Data Fig. 7 | Multi-session training and rapid decoding. **a**, Top: hippocampus dataset, single animal vs. multi-animal training shows an increase in consistency across animals. Bottom: same for Allen dataset, 4 mice. **b**, Consistency matrix single vs. multi-session training for hippocampus (32D embedding) and Allen data (128D embedding) respectively. Consistency is reported at the point in training where the average position decoding error is less than 14 cm (corresponds to 7 cm error for rat 1), and a decoding accuracy of 60% on the Allen dataset. **c**, Comparison of decoding metrics for single or multi-session training at various consistency levels (averaged across all 12 comparisons). Models were trained for 5,000 (single) or 10,000 (multi-session) steps with a 0.003 learning rate; batch size was 7,200 samples per session. Multi-session training requires longer training or higher learning rates to obtain the same accuracy due to the 4-fold larger batch size, but converges to same decoding accuracy. We plot points at intervals of 500 steps ($n = 10$ seeds); training progresses from lower right to upper left corner within both plots. **d**, We demonstrate that we could also adapt to an unseen dataset; here, 3 rats were used for pretraining, and rat 4 was used as a held-out test. The grey lines indicate models trained from scratch (random initialization). We also tested fine-tuning only the input embedding (first layer) or the full model, as the diagram, left, describes. We measured the average time (mean ± s.d.) to adapt 100 steps (0.65 ± 0.13 s) and 500 steps (3.07 ± 0.61 s) on 40 repeated experiments.



Extended Data Fig. 8 | Somatosensory cortex decoding from primate recordings. a, We compare CEBRA-Behaviour with the cosine similarity and embeddings on the sphere reproduced from Fig. 3b (left) against CEBRA-Behaviour trained with the MSE loss and unnormalized embeddings. The embeddings of trials ($n = 364$) of each direction were post-hoc averaged. **b**, CEBRA-Behaviour trained with *x*,*y* position of the hand. Left panel is colour-coded to changes in *x* position and right panel is color-coded to changes in *y* position. **c**, CEBRA-Time without any external auxiliary variables. As in **b**, left and right are colour-coded to *x* and *y* position, respectively. **d**, Decoding performance of target direction using CEBRA-Behaviour, conv-pi-VAE and a linear classifier. CEBRA-Behaviour shows significantly higher decoding performance than the linear classifier (one-way ANOVA, $F(2,75) = 3.37$, $P < 0.05$ with Post Hoc Tukey significant difference $P < 0.05$). **e**, Loss (InfoNCE) vs. training iteration for CEBRA-Behaviour with position, direction, active or passive, and position+direction labels (and shuffled labels) for all trials (left) or only active trials (right), or active trials with a MSE loss. **f**, Additional decoding performance results on position and direction-trained CEBRA models with all trial types. For each case, we trained and evaluated 5 seeds represented by black dots and the orange line represents the median. **g**, Results on the mc-maze 20 ms benchmark.



Extended Data Fig. 9 | CEBRA produces consistent, highly decodable embeddings. a, Additional 4 sessions with the most neurons in the Allen visual dataset calcium recording shown for all algorithms we benchmarked (see Methods). For CEBRA-Behaviour and CEBRA-Time, we used 3D, temperature 1, time offset 10, batch size 512 and 10k training steps. For UMAP, we used a cosine metric and *n_neighbors* 15 and *min_dist* 0.1. For t-SNE, we used a cosine metric and *perplexity 30*. For conv-pi-VAE, we trained with 600 epochs, a batch size of 200 and a learning rate 5×10^{-4}. autoLFADS was trained with ray-tune parameter selection and the resulting factors were transformed with PCA to generate the visualization. All methods used 10-time-bins input. CEBRA was trained with 3D latent and all other methods were obtained with an equivalent 2D latent dimension. To align for visualization, we aligned to mouse 1, except for conv-pi-VAE without labels and for autoLFADS, which visually looked best when aligned to mouse 4.



Extended Data Fig. 10 | See next page for caption.

Extended Data Fig. 10 | Spikes and calcium signalling reveal similar embeddings. a, Consistency between the single and jointly trained embeddings. **b**, Consistency of embeddings from two recording modalities, when a single modality was trained independently and/or jointly trained. CEBRA can find 'common latents' even without joint training. Data is also presented in Fig. 4e, h, but here plotted together to show improvement with joint training; for **a** and **b**, for each neuron number we have $n = 5$ shuffles, mean ± s.e.m. **c**-**f**, Consistency across modalities and areas for CEBRA-Behaviour and -Time (as computed in Fig. 4i–k). The purple dots indicate mean of intra-V1 scores and inter-V1 scores (inter-V1 vs intra-V1 one-sided Welch's t-test; 2P (Behaviour): $t(10.6) = 1.52$, $P = 0.081$, 2P (Time): $t(44.3) = 4.26$, $P = 0.0005$, NP (Behaviour): $t(11.6) = 2.83$, $P = 0.0085$, NP (Time): $t(8.9) = 15.51$, $P < 0.00001$) **g**, CEBRA + kNN decoding performance (see Methods) of CEBRA embeddings of different output embedding dimensions, from calcium (2P) data or Neuropixels (NP), as denoted; for each neuron number we have $n = 5$ shuffles, mean ± s.e.m. **h**, Decoding accuracy measured by considering predicted frame being within 1 s difference to true frame using CEBRA (2P only), jointly trained (2P+NP), or a baseline population vector kNN decoder (using time window 33 ms (single frame), or 330 ms (10 frame receptive field)); for each neuron number we have $n = 5$ shuffles, mean ± s.e.m. (**i**): Single frame performance and quantification using CEBRA 1 frame receptive field (NP data), or baseline models, $n = 900$ video frames. **j**, CEBRA-Behaviour used the DINO features as auxiliary labels and DINO-shuffled used the shuffled DINO features. We shuffled the frame order of DINO features within a repeat. Same shuffled order was use for all repeats. Colour code is frame number from the movie. The prediction is considered as true if the predicted frame is within 1 s from the true frame, and the accuracy (%) is noted next to the embedding. For mouse ID 1–4: 337, 353, 397, 475 neurons were recorded, respectively. **k**, Decoding performance from 2P data from different visual cortical areas from different layers using a 10-frame-window, 128D CEBRA-Behaviour model using DINO features; for each neuron number we have $n = 5$ shuffles, mean± s.e.m.

nature portfolio

Corresponding author(s):	Mackenzie Mathis
Last updated by author(s):	2023-Jan-27

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	no software was used for data collection.
Data analysis	Python >3.8, cebra==0.0.1 and 0.0.2, torch>=1.8.2, scikit-learn>=1.0.2, umap-learn==0.5.2, numpy==1.21.5

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Hippocampus dataset: https://crcns.org/data-sets/hc/hc-11/about-hc-11 and we used the pre-processing script from https://github.com/zhd96/pi-vae/blob/main/code/rat_preprocess_data.py. Primate dataset: https://gui. dandiarchive.org/#/dandiset/000127. Allen Institute dataset: Neuropixels data are at https://allensdk.readthedocs. io/en/latest/visual_coding_neuropixels.html. The pre-processed 2P recordings are available at https://github. com/zivlab/visual_drift/tree/main/data

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	No sample size method was used to predetermine sample size. We ran 100-120 seeds for synthetic datasets, with is larger than or similar to Zhou, D., & Wei, X. (2020). For neural datasets, we used the same sample size as in the original papers, ranging from 1 to 4 animals; see Grosmark & Buzsáki (2016), Chowdhury et al (2019), and for the Allen datasets see de Vries et al (2019) & Siegle et al (2021), and ran n=5 seeds on each neural subgrouping, which is greater or similar to machine learning papers, such as Roeder et al. (2020).
Data exclusions	We did not collect original data, thus did not exclude any data or model runs.
Replication	Each metric was run at least 5-100 time and hyperparameters were cross validated.
Randomization	No randomization of animals was possible or needed (no control vs. treatment groupings); models were randomly seeded and no data was excluded.
Blinding	We did not collect the data therefore blinding was not possible or required.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study	
☒	☐	Antibodies
☒	☐	Eukaryotic cell lines
☒	☐	Palaeontology and archaeology
☒	☐	Animals and other organisms
☒	☐	Clinical data
☒	☐	Dual use research of concern

Methods

n/a	Involved in the study	
☒	☐	ChIP-seq
☒	☐	Flow cytometry
☒	☐	MRI-based neuroimaging