Deep Graph Pose: a semi-supervised deep graphical model for improved animal pose tracking

Anqi Wu[1*] **E. Kelly Buchanan**[1*] **Matthew Whiteway**[1] **Michael Schartner**[2]

Guido Meijer[3] **Jean-Paul Noel**[4] **Erica Rodriguez**[1] **Claire Everett**[1]

Amy Norovich[1] **Evan Schaffer**[1] **Neeli Mishra**[1] **C. Daniel Salzman**[1]

Dora Angelaki[4] **Andrés Bendesky**[1] **The International Brain Laboratory**[5]

John Cunningham[1] **Liam Paninski**[1]

[1] Columbia University, New York, USA
{aw3236, ekb2154, mw3323, er2934, cpe2108, aln2128, ess2129, nm2786
cds2005, ab4463, jpc2181, lmp2107}@columbia.edu
[2] University of Geneva, Geneva, Switzerland
Michael.Schartner@unige.ch
[3] The Champalimaud Centre for the Unknown, Lisbon, Portugal
guido.meijer@research.fchampalimaud.org
[4] New York University, New York, USA
{jpn5, da93}@nyu.edu
[5] University of California, Berkeley, California, USA
info@internationalbrainlab.org

Abstract

Noninvasive behavioral tracking of animals is crucial for many scientific investigations. Recent transfer learning approaches for behavioral tracking have considerably advanced the state of the art. Typically these methods treat each video frame and each object to be tracked independently. In this work, we improve on these methods (particularly in the regime of few training labels) by leveraging the rich spatiotemporal structures pervasive in behavioral video — specifically, the spatial statistics imposed by physical constraints (e.g., paw to elbow distance), and the temporal statistics imposed by smoothness from frame to frame. We propose a probabilistic graphical model built on top of deep neural networks, Deep Graph Pose (DGP), to leverage these useful spatial and temporal constraints, and develop an efficient structured variational approach to perform inference in this model. The resulting semi-supervised model exploits both labeled and unlabeled frames to achieve significantly more accurate and robust tracking while requiring users to label fewer training frames. In turn, these tracking improvements enhance performance on downstream applications, including robust unsupervised segmentation of behavioral "syllables," and estimation of interpretable "disentangled" low-dimensional representations of the full behavioral video.

[*] equal contribution

1 Introduction

Animal pose estimation (APE) is a critical scientific task, with applications in ethology, psychology, neuroscience, and other fields. Recent work in neuroscience, for example, has emphasized the degree to which neural activity throughout the brain is correlated with movement [1, 2, 3]; i.e., to understand the brains of behaving animals we need to extract as much information as possible from behavioral video recordings. State of the art APE methods, such as Deep Lab Cut (DLC) [4], DeepPoseKit (DPK) [5], and LEAP [6], have transferred tools from the human pose estimation (HPE) deep learning literature to the APE setting [7, 8], opening up an exciting array of new applications and new scientific questions to be addressed.

However, even with these advances in place, hundreds of labels may still be needed to achieve tracking at the desired level of precision and reliability. Providing these labels requires significant user effort, particularly in the common case that users want to track multiple objects per frame (e.g., all the fingers on a hand or paw). Unlike HPE algorithms [9], APE algorithms are applied to a wide variety of different body structures (e.g., fish, flies, mice, or cheetahs) [10], compounding the effort required for collecting labeled datasets. Moreover, even with hundreds of labels, users still often see occasional "glitches" in the output (i.e., frames where tracking is briefly lost), which typically interfere with downstream analyses of the extracted behavior.

To reduce the prevalence of these "glitches," several state of the art human pose estimation algorithms have employed graphical models as regularizers to enforce spatial or temporal constraints [11, 12, 13, 14, 15]. Most of these methods use neural network regularizers [16, 17, 18], exploiting the abundance of large labeled training datasets to prevent overfitting - something not feasible in the APE setting, where training data is much more scarce. Works such as [13, 14] instead combine undirected graphical models with deep neural networks, assign tracked locations discrete values, and use discrete message passing algorithms during the inference step. More recently, [19] employs a convolutional neural network (CNN) with a fully connected conditional random field (CRF) model to learn spatial relationships from the data and avoid specifying certain tree structured models for the CRF, but this method does not exploit temporal information. By incorporating temporal information, the model would become a CRF with latent variables, which requires a completely different inference method. Meanwhile, among state of the art APE algorithms, DLC, DPK, and LEAP do not incorporate any prior temporal structural knowledge into their model, and therefore treat each video frame as independent samples during training. DPK incorporates spatial "part affinity fields" [20] to predict the targets as well as the edges among those targets, but estimating these terms again requires large labeled training sets.

To improve APE performance in the sparse-labeled-data regime, we propose a probabilistic graphical model built on top of deep neural networks, Deep Graph Pose (DGP), to leverage both spatial and temporal constraints, and develop an efficient structured variational approach to perform inference in this model. DGP models the targets as continuous random variables (unlike [13, 14]), resulting in a semi-supervised model that takes advantage of both labeled and unlabeled frames to achieve significantly more accurate and robust tracking, using fewer labels. Finally, we demonstrate that these tracking improvements enhance performance in downstream applications, including robust unsupervised segmentation of behavioral "syllables," and estimation of interpretable low-dimensional representations of the full behavioral video.

2 Model

The graphical model of DGP is summarized in Figure 1. We observe frames x_t indexed by t, along with a small subset of labeled markers $y_{t,j}$ (where j indexes the different targets we would like to track). The target locations $y_{t,j}$ on most frames are unlabeled, but we have several sources of information to constrain these latent variables: temporal smoothness constraints between the targets $y_{t,j}$ and $y_{t+1,j}$, which we capture with potentials ϕ_t; spatial constraints between the targets $y_{t,i}$ and $y_{t,j}$, which we model with spatial potentials ϕ_s; and information from the image x_t, modeled by ϕ_n.

We use the notation ϕ_n to indicate that this potential is parameterized by a neural network. A number of architectures could potentially be employed for ϕ_n [6, 5]; we chose to adapt the architecture used in DLC [4] here. We use a simple quadratic potential ϕ_t to impose temporal smoothness:

$$\phi_t^j(y_{t,j}, y_{t+1,j}) \quad = \quad \frac{1}{2}w_t^j||y_{t,j} - y_{t+1,j}||^2, \tag{1}$$

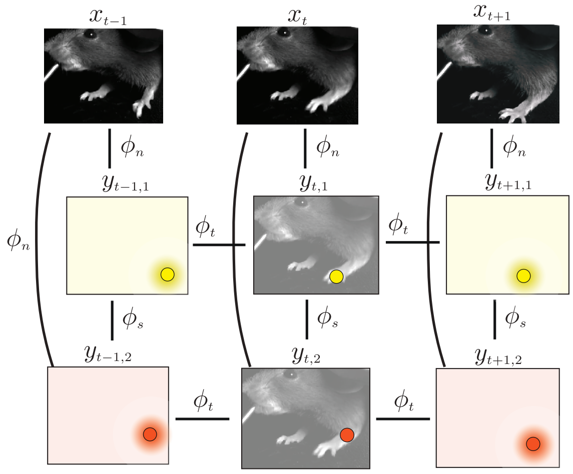

Figure 1: **Deep Graph Pose (DGP) model**. DGP leverages observed (labeled) and hidden information to infer the locations of unobserved targets via graph semi-supervised inference. At each time t, we observe the frame x_t. We want to track multiple targets in each frame (in this case, the paw and elbow). We also observe the labels of the two targets in some frames (in this example, in the t-th frame), denoted as $y_{t,1}$ and $y_{t,2}$ (colored circles at t). The hidden variables are the unobserved targets (indicated with colored circles in the colored background in frames $t-1$ and $t+1$ here).

which penalizes the distance between targets in consecutive frames; the weights w_t^j in general may depend on the target index j, but are constant in time.

The spatial potential ϕ_s is more dataset-dependent and can be chosen depending on the constraints that the markers should satisfy. Typical examples include a soft constraint that the paw marker should never be greater than some distance from the elbow marker, or the nose might always be within some radius of a waterspout with a fixed position. Again, we use a simple quadratic potential to encode these soft constraints:

$$\phi_s^{ij}(y_{t,i}, y_{t,j}) \quad = \quad \frac{1}{2} w_s^{ij} ||y_{t,i} - y_{t,j}||^2, \tag{2}$$

which penalizes the distance between "connected" targets $y_{t,i}$ and $y_{t,j}$ (where the user can pre-specify pairs of connected targets that should have neighboring locations in the frame, e.g. paw and elbow).

We want to "let the data speak" and avoid oversmoothing, so the penalty weights w_s and w_t should be small. In practice we found that w_t^j could be fixed to a small constant value (independent of dataset and target index j) with good results; similarly, setting $w_s^{ij} = c/d_{ij}$, where d_{ij} is a rough estimate of the average distance (in pixels) between targets i and j and $c > 0$ is a small scalar (again independent of dataset and target indices i, j), led to robust results without any need to fit extra parameters.

We summarize the parameter vector as $\beta = \{\theta, w_t, w_s\}$, where θ denotes the neural net parameters in ϕ_n. Given β, the joint probability distribution over targets y is

$$p(y|x, \beta) = \frac{1}{Z(x, \beta)} \exp\left(-\sum_{t=1}^{T}\sum_{j=1}^{J} \phi_n^j(y_{t,j}, x_t) - \sum_{t=1}^{T-1}\sum_{j=1}^{J} \phi_t^j(y_{t,j}, y_{t+1,j}) - \sum_{t=1}^{T}\sum_{i,j\in\mathcal{E}} \phi_s^{ij}(y_{t,i}, y_{t,j}) \right), \tag{3}$$

where $\mathcal{E}$ denotes the edge set of constrained targets (i.e., the pairs of markers i, j with a nonzero potential function), $Z(x, \beta) = \int p(y|x, \beta)dy$ is the normalizing constant marginalizing out y, T denotes the total number of frames, and J denotes the total number of targets.

3 Structured variational inference

Our goal is to estimate $p(y^h \mid y^v, x, \beta)$, the posterior over locations of unlabeled targets y^h, given the frames from the video x, the locations of the labeled markers y^v, and the parameters β. (Here h denotes hidden, for the unlabeled data, and v denotes visible, for the labeled data.) Calculating this posterior distribution exactly is intractable, due to the highly nonlinear potentials ϕ_n. We chose to use structured variational inference [21, 22] to approximate this posterior. We approximate $p(y^h, y^v \mid x, \beta)$ with a Gaussian graphical model (GGM) with the same graphical model as Figure 1, leading to a Gaussian posterior approximation $q(y^h \mid y^v, x, \beta)$ for $p(y^h \mid y^v, x, \beta)$ in which the inverse covariance (precision) matrix is block tridiagonal, with one block per frame t. (Since the potentials ϕ_t and ϕ_s are quadratic, yielding Gaussian distributions, the neural-network image potential ϕ_n is the only term that needs to be replaced with a new quadratic potential to form a Gaussian q.)

Updating the parameters of this GGM scales as $O(TJ^3)$ in the worst case, due to the chain structure of the graphical model (and the corresponding block tridiagonal structure of the precision matrix). If the edge graph $\mathcal{E}$ defined by the user-specified spatial potential function set is disconnected, this J^3 factor can be replaced by K^3, where K is the size of the largest connected component in $\mathcal{E}$.

We use a structured inference network approach [22] to estimate the model and variational parameters. We compute gradients of the evidence lower bound (ELBO) for this model using standard automatic differentiation tools, and perform standard stochastic gradient updates or quasi-Newton methods to estimate the parameters. Full details regarding the ELBO derivation and optimization can be found in the appendix.

3.1 Conceptual comparison against fully-supervised approaches

Standard fully-supervised approaches like DeepLabCut [4] learn a neural network (or more precisely, use transfer learning to adjust the parameters of an existing neural network) to essentially perform a classification task: the network is trained to output large values at the known location of the markers (i.e., the "positive" training examples), and small values everywhere else (the "negative" training examples). Given a small number of training examples, these methods are prone to overfitting.

In contrast, the approach we propose here is semi-supervised: it takes advantage of both the labeled and unlabeled frames to learn better model parameters θ. On labeled frames, the posterior distribution $p(y^v \mid y^v, x, \beta)$ is deterministic, and the objective function reduces to the fully supervised case. On the other hand, on unlabeled frames we have new terms in the objective function (see section S2.2.1 for more details). Clearly, the spatial and temporal potentials ϕ_s and ϕ_t encourage the outputs to be temporally smooth and to obey the user-specified spatial constraints (at least on average). But in addition the objective function encourages ϕ_n to output large values where $p(y^h \mid y^v, x, \beta)$ is large, and small values where $p(y^h \mid y^v, x, \beta)$ is small. Since we approximate $p(y^h \mid y^v, x, \beta)$ as Gaussian, the resulting ELBO encourages ϕ_n to be (on average) unimodal on unlabeled frames — a constraint that is not enforced in standard approaches. This turns out to be a powerful regularizer and can lead to significant improvements even in cases where the spatial and temporal constraints ϕ_s and ϕ_t are weak, as we will see in the next section.

4 Results

We applied the DGP method summarized above to a variety of datasets, including behavioral videos from three different species, in a variety of poses and environments (see Table 1 in the appendix for a summary). The new model (DGP) consistently outperformed the baseline (DLC). In each example video analyzed here, DLC outputs occasional "glitch" frames where tracking of at least one target is lost (e.g., around frame index 100 in the lower right panel); these glitches are much less prevalent in the DGP output.

We experimented with running Kalman smoothers and total variation denoisers to post-process the DLC output, but were unable to find any parameter settings that could reliably remove these glitches without oversmoothing the data (results not shown). (The frequency of these "glitches" can be reduced by increasing the training set through labeling more data — but this is precisely the user effort we aim to minimize here.) See the full videos summarizing the performance of the two methods here. An example screenshot for the mouse-wheel dataset [23] is shown in Figure 2. More example screenshots for all other datasets can be found in Figures S2-S5 in the appendix.

We also examined the "confidence maps" generated by visualizing the output of the neural network ϕ_n as an image; large values of the confidence map indicate the regions where the network "believes" the target is located with high confidence. Comparing the confidence maps output by DLC versus DGP, we see that the latter tend to be more unimodal (see Figure 2, small panels in the middle column). Nonetheless, DGP does occasionally output multi-modal confidence maps (e.g., in frames where the target is occluded), since the ELBO objective function used to train DGP encourages unimodality but does not impose unimodality as a hard constraint.

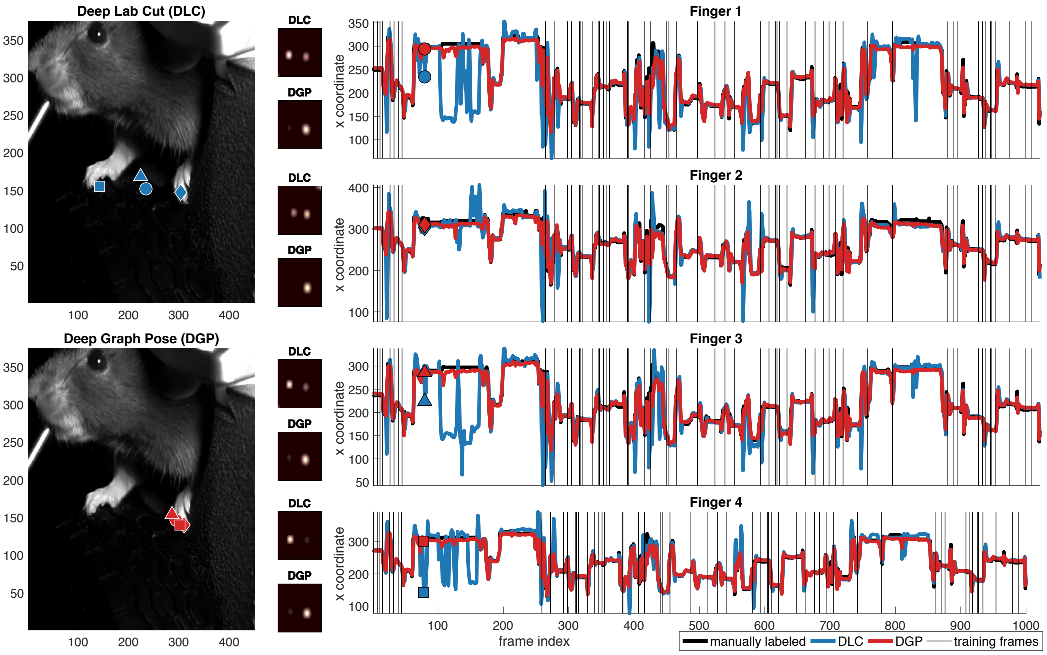

Figure 2: **Comparison of Deep Graph Pose (DGP) versus DeepLabCut (DLC) and manually-labeled data on the mouse-wheel dataset from [23]**; see also [24]. Left panels show an example frame, with the DLC output markers superimposed in blue (top) and the DGP markers in red (bottom). The right panels show the horizontal marker positions as a function of time (with DLC in blue, DGP in red and the full manually-labeled trace in black). Vertical lines indicate labeled (training) frames. The small inset images show confidence maps for each marker output by DLC (top) and DGP (bottom); the DGP confidence maps tend to be more unimodal than the DLC confidence maps. Note that the DLC and DGP marker locations tend to agree on labeled frames, but we see significant discrepancies on unlabeled test frames. Visual inspection of the videos (and comparison again the manual labels) indicates that when the DLC and DGP markers disagree, typically the DLC marker is in the wrong location.

To develop more quantitative comparisons, we manually labeled 1000 training frames in the mouse-wheel dataset[2] and randomly assigned 55 labels to the training set. Next we randomly subsampled $10-90\%$ of this training set and retrained the models to quantify the dependence of the test error on the number of labeled frames. Figure 3 shows the test error averaged over five random subsamples. Focusing on DGP (red) and DLC (blue) bars, we see that DGP outperforms DLC uniformly over the training set fraction (i.e., the number of labeled frames used to train the model). Similar results were obtained using an ϵ-insensitive loss that ignored errors below a threshold ϵ (on the order of 5-10 pixels here) below which the "true" marker location becomes somewhat subjective.

Next we performed an ablation study, to better understand the source of the performance gains exhibited by DGP. We examined two approaches that can be considered as conceptual waypoints between DLC and DGP. First, we experimented with a model in which the spatial and temporal potentials are turned off (i.e., $w_s = w_t = 0$) during both training and testing. The resulting graphical model is now independent over targets j and frames t, and variational inference in this model reduces to standard "mean-field" variational inference, in which the variational approximation is a product of Gaussian distributions, one for each $y_{t,j}$. We call the resulting model DLC-semi, since the resulting ELBO objective function combines a usual supervised loss (as in DLC) with an unsupervised term that encourages the output of the image potential ϕ_n to match its Gaussian approximation for each (t, j) pair (i.e., the resulting loss can be considered a semi-supervised DLC hybrid). Surprisingly, although this model does not enforce any spatial constraints or temporal smoothness, the extra

[2] This exhaustive labeling was labor-intensive and we have not yet performed the same analysis for the other datasets in Table 1. As is visible in the appendix figures, our qualitative results are similar across all the datasets analyzed here; we plan to perform more exhaustive comparisons on other datasets in the future.

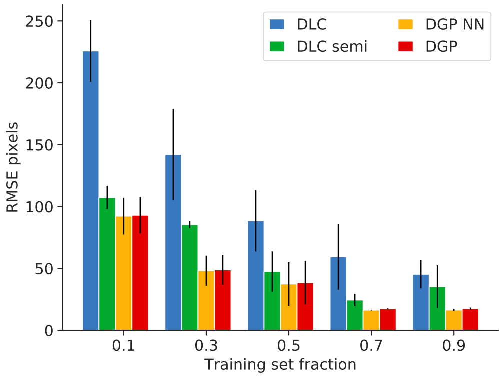

Figure 3: **Quantification of the results from Figure 2 over multiple training set sizes and ablation experiments**. DGP outperforms DLC and the intermediate variant DLC-semi, while the faster version DGP-NN achieves similar accuracy as full DGP over the range of training dataset sizes. We evaluated the different methods (see main text for definition of DLC-semi and DGP-NN) using multiple random subsets of the training set (55 labels) and compared the differences in test error. Error bars represent one standard error across five random trials.

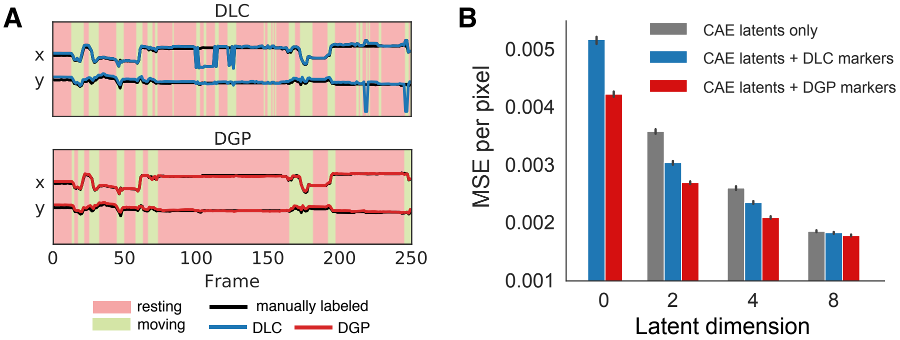

Figure 4: (A) **Simple unsupervised methods segment DGP traces into interpretable "resting" versus "moving" states, while DLC trace segmentation is hampered by glitches**. We ran a two-state autoregressive hidden Markov model (ARHMM) on the DGP and DLC outputs (in this case, on the x- and y-coordinates of a single paw). Background colors indicate the inferred states from the ARHMM fit to the DGP or DLC traces. The model fit with the DGP output clearly learns interpretable states, a "resting" state (red) and a "moving" state (green) (bottom). The model fit with the DLC output learns two states that are partially corrupted by "glitches" where DLC jumps away from the manually-labeled paw position (bottom); see video for full details. (B) **Conditioning CAEs on DGP markers improves reconstruction performance.** We computed mean square error (MSE) per pixel on reconstructed test frames from the mouse-wheel dataset when using a CAE (gray bars), or conditional CAEs, where the markers output by DLC (blue) or DGP (red) are used as input to both the encoder and decoder networks. A latent dimension of 0 corresponds to directly decoding the frames from markers. We see that test MSE decreases with latent dimensionality (as expected), and that the model conditioned on DGP markers consistently outperforms the model conditioned on DLC markers. Error bars represent 95% bootstrapped confidence interval over test frames. Reconstruction videos are available here.

regularization from the unsupervised term in the ELBO encourages the model output to be more unimodal, leading to significantly improved predictions compared to DLC, as shown in Figure 3. Indeed, DLC-semi performs almost as well as the full DGP model in some cases, although some significant gaps between DLC-semi versus DGP still remain. Similar results are seen in all of the other datasets examined here; see Figures S6-S10 in the appendix.

As a final comparison we considered training the full DGP model, but then turning off the spatial and temporal potentials (setting $w_s = w_t = 0$) at test time. This means that the model can exploit temporal and spatial regularities in the data to improve its estimate of the neural network parameters θ during training, but then at test time the output of the model depends solely on the estimated image potential ϕ_n. In practice, this results in a speedup at test time, since we simply have to perform one ϕ_n evaluation per marker $y_{t,j}$ (thus this approach has the same computational complexity as DLC), with no final E step required to enforce the spatial and temporal potentials. We call the resulting model DGP-NN. Figure 3 and the appendix Figures S6-S10 illustrate that DGP-NN performs similarly to DGP across datasets and a wide range of training dataset size. Thus, given the speed benefits at test time, DGP-NN seems to be the method of choice in practice.

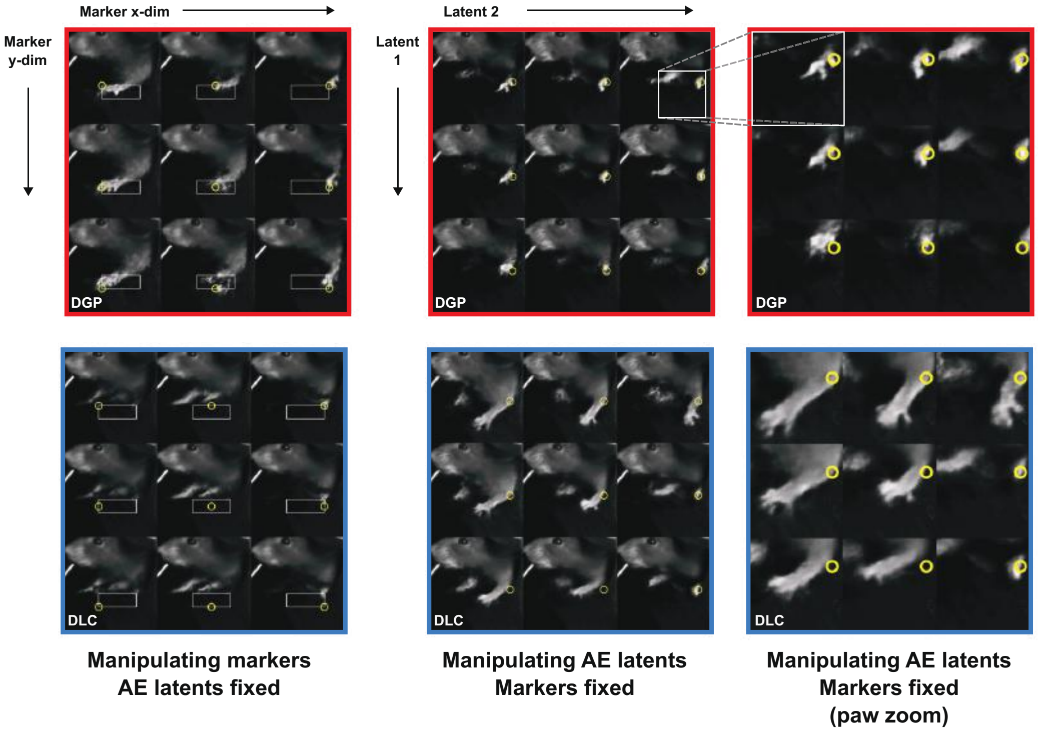

Figure 5: **Conditioning CAEs on DGP markers, but not DLC markers, leads to disentangled latents.** We incorporated the DLC and DGP markers into conditional CAEs trained on the mouse-wheel dataset. All frames are generated from 2-latent networks. **Left**: frames generated from the CAEs when changing the x and y coordinates of the left paw marker (yellow circle) for a given frame, with all other latents/markers fixed (white bounding box denotes the range of x/y coordinates). This manipulation should lead to noticeable changes in left paw position if markers are disentangled from latents. The network trained with DGP markers affords a much higher degree of control and produces more realistic looking images than that trained with DLC. **Center**: frames generated from the CAEs when changing the latents, with all markers fixed (white bounding box denotes the crop used for the right panels). This manipulation should not change the left paw position, but rather vary other (untracked) features of the image. Changes in the DGP reconstructions are limited to a small region around the tracked paws (yellow circle denotes left paw marker; see right panels for crop), demonstrating that the latents are encoding more local information such as paw configuration. DLC reconstructions show undesirable large movements of the left paw, demonstrating that the latents are encoding information about this tracked body part that should be present in the markers. **Right**: zoom of cropped region around the original paw location for frames in the center panel. See appendix Figure S1 for a more detailed quantitative analysis of latent/marker disentanglement.

4.1 Downstream analyses

The above results demonstrate that DGP provides improved tracking performance compared to DLC. Next we show that these accuracy improvements can in turn lead to more robust and interpretable results from downstream analyses based on the tracked output.

Unsupervised temporal segmentation. We begin with a simple segmentation task: given the estimated trace for the paw, can we use unsupervised methods to determine, e.g., when the paw is moving versus still? Figure 4A shows that the answer is yes if we use the DGP output: a simple two-state auto-regressive hidden Markov model (ARHMM; fit via Gibbs sampling on 1000 frames' output from either DGP or DLC; [25]) performs well with no further pre- or post-processing. In contrast, the multiple DLC "glitches" visible in Figure 2 contaminate the segmentation based on the DLC traces, resulting in unreliable segmentation. See the video for further details. Similar results were obtained when fitting models with more than two states (data not shown).

Conditional convolutional autoencoder (CAE) for more interpretable low-dimensional representation learning. As a second downstream application, we consider unsupervised dimensionality reduction of behavioral videos [3, 1, 26, 27]. This approach, which typically uses linear methods like singular value decomposition (SVD), or nonlinear methods like convolutional autoencoders (CAEs), does not require user effort to label video frames. However, interpreting the latent features of these models can be difficult [28, 29], limiting the scientific insight gained by using these models. A hybrid approach that combines supervised (or semi-supervised) object tracking with unsupervised CAE training has the potential to ameliorate this problem [30, 31, 32, 33] – the tracked targets encode information about the location of specific body parts, while the estimated CAE latent vectors encode the remaining sources of variability in the frames. We refer to this ideal partitioning of variability into more interpretable subspaces as "disentangling." Below we show that these hybrid models produce features that are more disentangled when trained with the output from DGP compared to DLC.

We fit conditional CAEs that take the markers output by DLC or DGP (hereafter referred to as CAE-DLC and CAE-DGP, respectively) as conditional inputs into both the encoding and decoding networks of the CAE, using the mouse-wheel dataset with 13 randomly chosen labeled frames (see Section S3 for implementation details). For this analysis, to obtain useful information across the full image, we labeled the left paw, right paw, tongue, and nose, rather than the four fingers on the left paw as in the previous section. Incorporating the tracking output from either method decreases the mean square error (MSE) of reconstructed test frames, for a given number of latents (Figure 4B). Furthermore, the networks trained with DGP outputs show improved performance over those trained with DLC outputs. Subsequent analyses are performed on the 2-latent networks, for easier visualization.

To test the degree of disentanglement between the CAE latents and the DGP or DLC output markers, we performed two different manipulations. First, we asked how changing individual markers affects the CAE reconstructions. We manipulate the x/y coordinates of a single marker while holding all other markers and all latents fixed. If the markers are disentangled from the latents we would expect to see the body part corresponding to the chosen marker move around the image, while all other features remain constant. We randomly chose a test frame and simultaneously varied the x/y marker values of the left paw (Figure 5, left). This manipulation results in realistic looking frames with clear paw movements in the CAE-DGP reconstructions, demonstrating that this marker information has been incorporated into the decoder. For the CAE-DLC reconstructions, however, this manipulation does not lead to clear movements of the left paw, indicating that the decoder has not learned to use these markers as effectively (a claim which is also supported by the higher MSE in the CAE-DLC networks, Figure 4B).

Second, we asked how changing the latents (rather than markers) affects the reconstructed frames. In this manipulation we simultaneously change the values of the two latents while holding all markers fixed. If the latents are disentangled from the markers we expect to see the tracked features remain constant while other, untracked features change. For the CAE-DGP network this latent manipulation has very little effect on the tracked body parts, as desired (Figure 5, top center); instead, the manipulation leads to small changes in the configuration of the left paw (rather than its absolute location; Figure 5, top right). On the other hand, for the CAE-DLC network this latent manipulation has a large effect on the left paw location (Figure 5, bottom center), which should instead be encoded by the markers. These results qualitatively demonstrate that the CAE-DGP networks have better learned to disentangle the markers and the latents, a desirable property for more in-depth behavioral analysis. Furthermore, we find through an unbiased, quantitative assessment of disentangling, that using DGP markers in these models leads to higher levels of disentangling between latents and markers than DLC across many different animal poses present in this dataset (see Figure S1).

5 Discussion

In this work, we proposed a probabilistic graphical model built on top of deep neural networks, Deep Graph Pose (DGP), which leverages the rich spatial and temporal structures pervasive in behavioral videos. We also developed an efficient structured variational approach to perform inference in this model. The resulting semi-supervised model exploits information from both labeled and unlabeled frames to achieve significantly more accurate and robust tracking, using fewer labels. Our results illustrate how the smooth behavioral trajectories from DGP lead to improved downstream applications, including the discovery of behavioral “syllables,” and interpretable or “disentangled” low-dimensional features from the behavioral videos.

An important direction for future work is to optimize the code to perform online inference for real-time experiments, as in [34]. As emphasized in the context of Figure 3, the version of DGP without the E-step at test time, DGP-NN, has the same computational complexity as DLC or DPK and should therefore be amenable to online tracking, while usually achieving the same accuracy as the full DGP approach. Extending our method to operate in 3D, fusing information from multiple cameras, would be another important direction for future work. Our variational inference approach should be extensible to this case, using similar epipolar constraints as in [35, 36] (using different inference approaches) to perform semi-supervised inference across views.

In addition, [4, 5, 6] all use slightly different architectures and achieve similar accuracies. We plan to perform more experiments with the architectures from [5, 6] in the future. Finally, we would like to incorporate our model into existing toolboxes and GUIs to facilitate user access.

Code

Open source code is available here.

6 Broader impacts

We propose a new method for animal behavioral tracking. As highlighted in the introduction and in [10], recent years have seen a rapid increase in the development of methods for animal pose estimation, which need to operate in a different regime than methods developed for human pose estimation. Our work significantly improves the state of the art for animal pose estimation, and thus advances behavioral analysis for animal research, an essential task for scientific discovery in fields ranging from neuroscience to ecology. Finally, our work represents a compelling fusion of deep learning methods with probabilistic graphical model approaches to statistical inference, and we hope to see more fruitful interactions between these rich topic areas in the future.

References

[1] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437):eaav7893, 2019.

[2] Nicholas A Steinmetz, Peter Zatka-Haas, Matteo Carandini, and Kenneth D Harris. Distributed coding of choice, action and engagement across the mouse brain. *Nature*, 576(7786):266–273, 2019.

[3] Simon Musall, Matthew T Kaufman, Ashley L Juavinett, Steven Gluf, and Anne K Churchland. Single-trial neural dynamics are dominated by richly varied movements. *Nature neuroscience*, 22(10):1677–1686, 2019.

[4] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. Technical report, Nature Publishing Group, 2018.

[5] Jacob M Graving, Daniel Chae, Hemal Naik, Liang Li, Benjamin Koger, Blair R Costelloe, and Iain D Couzin. Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife*, 8:e47994, 2019.

[6] Talmo D Pereira, Diego E Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S-H Wang, Mala Murthy, and Joshua W Shaevitz. Fast animal pose estimation using deep neural networks. *Nature methods*, 16(1):117, 2019.

[7] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[8] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[9] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.

[10] Mackenzie Weygandt Mathis and Alexander Mathis. Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, 60:1–11, 2020.

[11] Xianjie Chen and Alan L Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in neural information processing systems*, pages 1736–1744, 2014.

[12] Guoqiang Liang, Xuguang Lan, Jiang Wang, Jianji Wang, and Nanning Zheng. A limb-based graphical model for human pose estimation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(7):1080–1092, 2017.

[13] Steven Schwarcz and Thomas Pollard. 3d human pose estimation from deep multi-view 2d pose. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2326–2331. IEEE, 2018.

[14] Deying Kong, Yifei Chen, Haoyu Ma, Xiangyi Yan, and Xiaohui Xie. Adaptive graphical model network for 2d handpose estimation. *arXiv preprint arXiv:1909.08205*, 2019.

[15] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–84, 2018.

[16] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[17] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

[18] Xiao Chu, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Structured feature learning for pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4715–4723, 2016.

[19] Lisha Chen, Hui Su, and Qiang Ji. Deep structured prediction for facial landmark detection. In *Advances in Neural Information Processing Systems*, pages 2447–2457, 2019.

[20] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[21] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[22] Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. *arXiv preprint arXiv:1803.05589*, 2018.

[23] G. T. Meijer, M. M. Schartner, V. Aguillon, N. Bonacchi, M. Carandini, F. Cazettes, G. A. Chapius, A. K. Churchland, Y. Dan, E. E. J. Dewitt, H. Martinez Vergara, M. Faulkner, M. Hausser, F. Hu, I. C. Laranjeira, Z. F. Mainen, N. J. Miska, T. D. Mrsic-flogel, J. P. Noel, A. Pan Vazquez, L. M. Paninski, A. Pouget, K. Z. Socha, K. Svoboda, A. E. Urai, M. R. Whiteway, O. Winter, and IBL Collaboration. Robust and generalizable tracking of body parts of head-fixed mice. In *SFN*, 2019.

[24] The International Brain Laboratory, Valeria Aguillon-Rodriguez, Dora E. Angelaki, Hannah M. Bayer, Niccolò Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle A. Chapuis, Anne K. Churchland, Yang Dan, Eric E. Dewitt, Mayo Faulkner, Hamish Forrest, Laura M. Haetzel, Michael Hausser, Sonja B. Hofer, Fei Hu, Anup Khanal, Christopher S. Krasniak, Inês Laranjeira, Zachary F. Mainen, Guido T. Meijer, Nathaniel J. Miska, Thomas D. Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Josh I. Sanders, Karolina Z. Socha, Rebecca Terry, Anne E. Urai, Hernando M. Vergara, Miles J. Wells, Christian J. Wilson, Ilana B. Witten, Lauren E. Wool, and Anthony Zador. A standardized and reproducible method to measure decision-making in mice. *bioRxiv*, 2020.

[25] Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping sub-second structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

[26] Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pages 2946–2954, 2016.

[27] Eleanor Batty, Matthew Whiteway, Shreya Saxena, Dan Biderman, Taiga Abe, Simon Musall, Winthrop Gillis, Jeffrey Markowitz, Anne Churchland, John P Cunningham, et al. Behavenet: nonlinear embedding and bayesian neural decoding of behavioral videos. In *Advances in Neural Information Processing Systems*, pages 15680–15691, 2019.

[28] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

[29] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

[30] Antonia Creswell, Anil A Bharath, and Biswa Sengupta. Conditional autoencoders with adversarial information factorization. *arXiv preprint arXiv:1711.05175*, 2017.

[31] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976, 2017.

[32] Jack Klys, Jake Snell, and Richard Zemel. Learning latent subspaces in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 6444–6454, 2018.

[33] Xiao Li, Chenghua Lin, Chaozheng Wang, and Frank Guerin. Latent space factorisation and manipulation via matrix subspace projection. *arXiv preprint arXiv:1907.12385*, 2019.

[34] Jens F Schweihoff, Matvey Loshakov, Irina Pavlova, Laura Kück, Laura A Ewell, and Martin K Schwarz. Deeplabstream: Closing the loop using deep learning-based markerless, real-time posture detec-tion. *bioRxiv*, 2019.

[35] Yilun Zhang and Hyun Soo Park. Multiview supervision by registration. *arXiv preprint arXiv:1811.11251*, 2018.

[36] Semih Günel, Helge Rhodin, Daniel Morales, João Campagnolo, Pavan Ramdya, and Pascal Fua. Deepfly3d, a deep learning-based approach for 3d limb and appendage tracking in tethered, adult drosophila. *eLife*, 8, 2019.

[37] Virginie Uhlmann, Pavan Ramdya, Ricard Delgado-Gonzalo, Richard Benton, and Michael Unser. Flylimbtracker: An active contour based approach for leg segment tracking in unmarked, freely behaving drosophila. *PLoS One*, 12(4), 2017.

[38] Praneet C Bala, Benjamin R Eisenreich, Seng Bum Michael Yoo, Benjamin Y Hayden, Hyun Soo Park, and Jan Zimmermann. Openmonkeystudio: Automated markerless pose estimation in freely moving macaques. *bioRxiv*, 2020.

[39] Oliver Sturman, Lukas Matthias von Ziegler, Christa Schälppi, Furkan Akyol, Benjamin Friedrich Grewe, and Johannes Bohacek. Deep learning based behavioral analysis enables high precision rodent tracking and is capable of outperforming commercial solutions. *bioRxiv*, 2020.

[40] Adam J Calhoun and Mala Murthy. Quantifying behavior to solve sensorimotor transformations: advances from worms and flies. *Curr. Opin. Neurobiol.*, 46:90–98, August 2017.

[41] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[42] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised Learning*. Adaptive computation and machine learning. MIT Press, 2010.

[43] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, pages 1–22, 2019.

[44] Norimichi Ukita and Yusuke Uematsu. Semi-and weakly-supervised human pose estimation. *Computer Vision and Image Understanding*, 170:67–78, 2018.

[45] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[46] James M Varah. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Mathematics of Computation*, 26(120):859–868, 1972.

[47] Luca Guido Molinari. Determinants of block tridiagonal matrices. *Linear algebra and its applications*, 429(8-9):2221–2226, 2008.

[48] Matthew G Reuter and Judith C Hill. An efficient, block-by-block algorithm for inverting a block tridiagonal, nearly block toeplitz matrix. *Computational Science & Discovery*, 5(1):014009, 2012.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] Mackenzie Weygandt Mathis, Alexander Mathis, and Naoshige Uchida. Somatosensory cortex plays an essential role in forelimb motor adaptation in mice. *Neuron*, 93(6):1493–1503, 2017.

[51] Evan Schaffer, Neeli Mishra, Wenze Li, Matthew Whiteway, Jason Freedman, Kripa Patel, Venkatakaushik Voleti, Liam Paninski, Larry Abbott, Elizabeth Hillman, and Richard Axel. Flygenvectors: large-scale dynamics of internal and behavioral states in a small animal. In *Cosyne*, 2020.

[52] Amy L. Norovich*, Claire P. Everett*, Taiga Abe, and Andrés Bendesky. Probing the neural basis of visually-evoked aggression in siamese fighting fish. In *Cold Spring Harbor Zebrafish Neural Circuits and Behavior, November 20-23, Cold Spring Harbor, NY, USA*, 2019.

S1 Related Work

Animal pose estimation. The proposed approach fills a void between state of the art human pose estimation algorithms, which often rely on large quantities of manually labeled samples (see [9] for a recent review), and their counterparts in animal pose estimation [37, 4, 6, 5, 38, 39]. Among these animal pose estimation algorithms, Deep Lab Cut (DLC) [4], Leap Estimates Animal Pose (LEAP) [6], and Deep Pose Kit (DPK) [5], stand out as they can achieve near human-level accuracy using a modest number of labels.

As emphasized above, the major difference between these approaches and the method we propose here is in the treatment of the large number of unlabeled frames that are typically available in behavioral videos. DLC and LEAP do not incorporate any temporal or spatial structural knowledge into their models, while DLC, LEAP, and DPK treat frames independently, only learning from labeled samples. DPK incorporates some skeleton structures in labeled frames to predict the targets as well as the edges among those targets, similar to the part affinity fields described by [40]. However, this skeleton model must again be learned from labeled data, and it is not clear if the skeletal model can be learned with few data points (and therefore may not be useful in the low-data regime we are most interested in).

In our case, we encode spatial structure in a mild spatial potential that does not need to be learned, simply applying a soft constraint on the distance between pairs of targets that are known to be close (e.g., the fingers on a hand). It would be interesting in future work to explore combinations of the simple approach we use here with the more detailed skeletal model employed by [20, 5], since we expect the benefits of these two terms to be complementary, with the former most helpful in the low-training-data regime and the latter potentially more helpful once large training sets are acquired.

Graphical models. Previous work on human pose estimation has employed graphical models as network regularizers [11, 12, 13, 14]. Among these, [13] and [14], like DGP, build an undirected graphical model (UGM) on top of deep neural networks. However, [13] and [14] assign tracked locations discrete values, which allows for (discrete) message passing algorithms during the inference step. DGP models the targets as continuous random variables, and estimates the unknown targets using variational inference. Another critical difference between the proposed method and similar previous approaches is that the latter focused on labeled frames. Therefore, their UGMs were conditional random fields without additional hidden variables. Again, our proposed model includes a mixture of hidden and visible variables, leading to a semi-supervised learning framework.

Semi-supervised learning. Semi-supervised learning aims to fully utilize unlabeled or weakly-labeled data to gain additional insights into the structure of the data [41, 42, 43]. Many pose estimation algorithms have adopted such learning schemes to enhance the performance given limited training data [44, 35]. One conceptually similar “weakly-supervised” approach is described in [45], who trained a network to extract flying objects (obeying Newtonian acceleration) simply by constraining the output to resemble a parabola. As discussed above, the ELBO objective function in DGP encourages the output to follow a Gaussian distribution on each frame; this can be seen as a form of weak supervision that leads to improved accuracy even when the temporal and spatial soft constraints are removed (Figure 3).

S2 Expanded methods

In this section we present our model and inference approach in fuller detail than was possible given space limitations in the main text. (To maintain the logical flow, in some cases we repeat points that were made in the main text methods.)

S2.1 Deep Graph Pose model

The graphical model of DGP is summarized in Figure 1. We observe frames x_t indexed by t, along with a small subset of labeled markers $y_{t,j}$ (where j indexes the different targets we would like to track). The target locations $y_{t,j}$ on most frames are unlabeled, but we have several sources of information to constrain these latent variables: temporal smoothness constraints between the targets $y_{t,j}$ and $y_{t+1,j}$, which we capture with quadratic potentials ϕ_t; spatial constraints between the targets

$y_{t,i}$ and $y_{t,j}$, modeled with quadratic potentials ϕ_s; and information from the image x_t, modeled by potentials ϕ_n parametrized by neural networks.

First let's define the potential function ϕ_n between the input image x and the target's 2D location y. We define $f_\theta(\cdot)$ as a stack of a fixed pretrained ResNet-50 network and a trainable ConvNet parametrized by θ. $f_\theta(\cdot)$ takes a frame x as the input and outputs a 2D affinity map image, which ideally has a sharp peak at the most likely coordinates of the target. We then denote the sigmoid function as $\sigma(\cdot)$, and refer to $\sigma(f_\theta(x))$ as a "confidence map."

With the potential ϕ_n, our target is to match this 2D confidence map to a 2D Gaussian bump centered at y by minimizing the sigmoid cross entropy. Now let's define the Gaussian bump. We construct a bivariate Gaussian function with mean $y = [y_m, y_n]$ and variance l^2. The Gaussian function at the mth row and the nth column is

$$G(y, l^2)_{mn} = \frac{1}{2\pi l^2} \exp(-\frac{1}{2l^2}(m - y_m)^2 - \frac{1}{2l^2}(n - y_n)^2). \tag{S4}$$

The variance parameter was set as $l^2 = 1$ in practice.

The potential function ϕ_n for the mth row and the nth column entry in $\sigma(f_\theta(x))$ is defined as

$$\begin{aligned} \phi_n(y, x)_{mn} &= \frac{1}{2} w_n \left(-G(y, l^2)_{mn} \cdot \log(\sigma(f_\theta(x))_{mn}) - (1 - G(y, l^2)_{mn}) \cdot \log(1 - \sigma(f_\theta(x))_{mn})\right) \\ &= \frac{1}{2} w_n \left(-f_\theta(x)_{mn} \cdot G(y, l^2)_{mn} + f_\theta(x)_{mn} + \log(1 + \exp(-f_\theta(x)_{mn}))\right). \end{aligned}$$

Summing over all entries in the confidence map, we get the neural network potential ϕ_n as

$$\phi_n(y, x) = \sum_{m,n} \frac{1}{2} w_n \left(-f_\theta(x)_{mn} \cdot G(y, l^2)_{mn} + f_\theta(x)_{mn} + \log(1 + \exp(-f_\theta(x)_{mn}))\right).$$

We will write everything in vector form hereafter. We define $\mathbf{f}$ as the vectorized $f_\theta(x)$, define $\mathbf{h}$ as the vectorized $f_\theta(x) + \log(1 + \exp(-f_\theta(x)))$, and define $\mathbf{G}(y, l^2)$ as the vectorized $G(y, l^2)$, which is a function of mean y and variance l^2. Thus, for each target j we can rewrite the j-th image-based potential ϕ_n^j as

$$\phi_n^j(y_{t,j}, x_t) = \frac{1}{2} w_n(-\mathbf{f}_{t,j}^\top \mathbf{G}(y_{t,j}, l^2) + \mathbf{h}_{t,j}^\top \mathbf{1}), \tag{S5}$$

where j is the index for target j and t is the index for frame t.

We use a simple quadratic potential ϕ_t to impose temporal smoothness:

$$\phi_t^j(y_{t,j}, y_{t+1,j}) = \frac{1}{2} w_t^j ||y_{t,j} - y_{t+1,j}||^2, \tag{S6}$$

which penalizes the distance between targets in consecutive frames. The weights w_t^j in general may depend on the target index j, but are constant in time.

The spatial potential ϕ_s is dataset-dependent and can be selected according to the spatial constraints the markers should satisfy. Typical examples include a soft constraint that the paw should never be greater than some distance from the elbow.Again, we use a simple quadratic potential to encode these soft constraints:

$$\phi_s(y_{t,i}, y_{t,j}) = \frac{1}{2} w_s^{ij} ||y_{t,i} - y_{t,j}||^2, \tag{S7}$$

which penalizes the distance between "connected" targets $y_{t,i}$ and $y_{t,j}$. The user pre-specifies pairs of connected targets that should remain close across time, e.g. paw and elbow.

We want to "let the data speak" and avoid oversmoothing, so the penalty weights w_s and w_t should be small. In practice we found that w_t^j could be fixed to a small constant value (independent of dataset and target index j) with good results; similarly, setting $w_s^{ij} = c/d_{ij}$, where d_{ij} is a crude estimate of the average distance (in pixels) between targets i and j and $c > 0$ is a small scalar (again independent of dataset and target indices i, j), led to robust results without any need to fit extra parameters.

We summarize the parameter vector as $\beta = \{\theta, w_n, w_t, w_s\}$, where θ denotes the neural net parameters. Given β and the full collection of images x, the joint probability distribution over targets y

is

$$p(y|x,\beta) = \frac{1}{Z(x,\beta)} \exp\Bigg(\underbrace{-\sum_{t=1}^{T}\sum_{j=1}^{J}\phi_n^j(y_{t,j},x_t)}_{\text{neural network}}\underbrace{-\sum_{t=1}^{T-1}\sum_{j=1}^{J}\phi_t^j(y_{t,j},y_{t+1,j}) - \sum_{t=1}^{T}\sum_{i,j\in\mathcal{E}}\phi_s^{ij}(y_{t,i},y_{t,j})}_{\text{Gaussian graphical model}}\Bigg), \tag{S8}$$

where $\mathcal{E}$ denotes the edge set of constrained targets (i.e., the pairs i, j with a nonzero potential function), $Z(x,\beta) = \int p(y|x,\beta)dy$ is the normalizing constant marginalizing out y, T denotes the total number of frames, and J denotes the total number of targets. The joint distribution can be described as a combination of a neural network component and a probabilistic graphical model over the latent variables (the unobserved targets y).

S2.2 Structured variational inference

Our goal is to estimate $p(y^h \mid y^v, x, \beta)$, the posterior over locations of unlabeled targets y^h, given the frames from the video x, the locations of the labeled markers y^v, and the parameters β. (Here h denotes hidden, for the unlabeled data, and v denotes visible, for the labeled data.) Calculating this posterior distribution exactly is intractable, due to the highly nonlinear potentials ϕ_n. We chose to use structured variational inference, similar to [22], to approximate this posterior. We approximate $p(y^h, y^v \mid x, \beta)$ with a Gaussian graphical model (GGM) with the same graphical model as Figure 1. We denote the approximate posterior as $q(y^h, y^v|x, \beta_q)$ (β_q encodes variational parameters). To obtain a fully Gaussian variational approximation, we replace the neural network potentials ϕ_n^j with quadratic terms

$$\hat{\phi}_n^j(y_{t,j}, x_t) = \frac{1}{2} w_{n,q}^{t,j} ||y_{t,j} - \mu_n^{t,j}(x_t)||^2. \tag{S9}$$

Here the precision variables $w_{n,q}^{t,j}$ and means $\mu_n^{t,j}$ are variational parameters that we could optimize over independently. However, we found it more efficient to model the means μ_n as $\mu_n^{t,j}(x_t) = \sum_{m,n} \alpha_{mn}\text{Softmax}(f_\gamma^j(x_t))_{mn}$. Here $f_\gamma(\cdot)$ is an inference neural network with parameters γ whose output is a 2D affinity map, similar to $f_\theta(\cdot)$. Putting the pieces together, we have the fully Gaussian approximate posterior

$$q(y^h, y^v|x,\beta_q) = \frac{1}{\hat{Z}(x,\beta_q)} \exp\Bigg(\underbrace{-\sum_{t=1}^{T}\sum_{j=1}^{J}\hat{\phi}_n^j(y_{t,j},x_t)}_{\text{inference network}}\underbrace{-\sum_{t=1}^{T-1}\sum_{j=1}^{J}\phi_t^j(y_{t,j},y_{t+1,j}) - \sum_{t=1}^{T}\sum_{i,j\in\mathcal{E}}\phi_s^{ij}(y_{t,i},y_{t,j})}_{\text{identical to equation S8}}\Bigg), \tag{S10}$$

where $\hat{Z}(x, \beta_q)$ is the normalizing constant (which can be computed explicitly, due to the fully-Gaussian form of q), and $\beta_q = \{\gamma, w_{n,q}, w_t, w_s\}$.

Since $q(y^h, y^v|x, \beta_q)$ is a GGM, we can rewrite eq. S10 in the standard Gaussian form

$$\begin{aligned} q(y^h, y^v|x,\beta_q) &= \mathcal{N}(\mu_a, \Sigma_a) \\ \Sigma_a &= (\Sigma_s^{-1} + \Sigma_t^{-1} + \Sigma_n^{-1})^{-1} && \text{(S11)} \\ \mu_a &= \Sigma_a \Sigma_n^{-1}\mu_n && \text{(S12)} \end{aligned}$$

where Σ_n^{-1}, Σ_t^{-1} and Σ_s^{-1} are the precision matrices corresponding to the potentials in $q(y^h, y^v|x, \beta_q)$; these have the form

$$\Sigma_n^{-1} = \begin{bmatrix} w_{n,q}^{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{n,q}^{1,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{n,q}^{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{n,q}^{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{n,q}^{3,1} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{n,q}^{3,2} \end{bmatrix} \tag{S13}$$

$$\Sigma_t^{-1} = \begin{bmatrix} w_t^1 & 0 & -w_t^1 & 0 & 0 & 0 \\ 0 & w_t^2 & 0 & -w_t^2 & 0 & 0 \\ -w_t^1 & 0 & 2w_t^1 & 0 & -w_t^1 & 0 \\ 0 & -w_t^2 & 0 & 2w_t^2 & 0 & -w_t^2 \\ 0 & 0 & -w_t^1 & 0 & w_t^1 & 0 \\ 0 & 0 & 0 & -w_t^2 & 0 & w_t^2 \end{bmatrix} \tag{S14}$$

$$\Sigma_s^{-1} = \begin{bmatrix} w_s & -w_s & 0 & 0 & 0 & 0 \\ -w_s & w_s & 0 & 0 & 0 & 0 \\ 0 & 0 & w_s & -w_s & 0 & 0 \\ 0 & 0 & -w_s & w_s & 0 & 0 \\ 0 & 0 & 0 & 0 & w_s & -w_s \\ 0 & 0 & 0 & 0 & -w_s & w_s \end{bmatrix}. \tag{S15}$$

Thus the mean and covariance of the variational distribution $q(y^h, y^v|x, \beta_q)$ are μ_a and Σ_a, where μ_a is a function of γ, $w_{n,q}$, w_t, and w_s, and Σ_a is a function of $w_{n,q}$, w_t, and w_s.

Let P_h and P_v denote the permutation matrices that map the vector y to y^h and y^v respectively, i.e.

$$y^h = P_h y, \quad y^v = P_v y. \tag{S16}$$

Due to the Gaussianity of the joint distribution, we can write down the closed-form expression for $q(y^h \mid y^v, x, \beta_q)$ as

$$q(y^h \mid y^v, x, \beta_q) \quad = \quad \mathcal{N}(\mu_h, \Sigma_h), \tag{S17}$$

where

$$\Sigma_h \quad = \quad (P_h \Sigma_a^{-1} P_h^\top)^{-1}, \tag{S18}$$

$$\mu_h \quad = \quad P_h \mu_a - (P_h \Sigma_a^{-1} P_h^\top)^{-1} P_h \Sigma_a^{-1} P_v^\top (y^v - P_v \mu_a). \tag{S19}$$

S2.2.1 Evidence Lower Bound (ELBO)

Given the approximate posterior (eq. S17), and abbreviating $q(y^h) = q(y^h \mid y^v, x, \beta_q)$, we can now write down the evidence lower bound (ELBO) as

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{q(y^h)}[-\log q(y^h) + \log p(y^h, y^v|x, \beta)] \\
&= \mathbb{E}_{q(y^h)}[\log p(y^h, y^v|x, \beta)] + H(q) \\
&= -\mathbb{E}_{q(y^h)}\left[\sum_{t=1}^{T}\sum_{j=1}^{J} \phi_n^j(y_{t,j}, x_t)\right] - \mathbb{E}_{q(y^h)}\left[\sum_{t=1}^{T-1}\sum_{j=1}^{J} \phi_t^j(y_{t,j}, y_{t+1,j})\right] - \mathbb{E}_{q(y^h)}\left[\sum_{t=1}^{T}\sum_{i,j\in\mathcal{E}} \phi_s^{ij}(y_{t,i}, y_{t,j})\right] \\
&\quad -\log Z(x, \beta) + H(q) \\
&= \sum_{\substack{t=1,j=1 \\ t,j\in\mathcal{V}}}^{T,J} \frac{w_n}{2}(-\mathbf{f}_{t,j}^\top \mathbf{G}(y_t, l^2) + \mathbf{h}_{t,j}^\top \mathbf{1}) + \sum_{\substack{t=1,j=1 \\ t,j\in\mathcal{H}}}^{T,J} \frac{w_n}{2}(-\mathbf{f}_{t,j}^\top \mathbf{G}(\mu_{ht}, l^2 + \Sigma_{htt}) + \mathbf{h}_{t,j}^\top \mathbf{1}) \\
&\quad -\mathrm{Tr}((\Sigma_s^{-1} + \Sigma_t^{-1}) P_h^\top \Sigma_h P_h) - \frac{1}{2}\mathrm{Tr}((P_h^\top \mu_h + P_v^\top y^v)^\top (\Sigma_s^{-1} + \Sigma_t^{-1})(P_h^\top \mu_h + P_v^\top y^v)) \\
&\quad -\log Z(x, \beta) + \log|\Sigma_a| - \log|P_v \Sigma_a P_v^\top|,
\end{aligned} \tag{S20}$$

where $\mathcal{V}$ and $\mathcal{H}$ denote the sets of visible targets in visible frames and hidden targets in all frames respectively.

S2.2.2 Semi-supervised DLC

To understand the various terms in the ELBO above it is helpful to start with a simpler special case. If we turn off the temporal and spatial potentials in eq. S20 (i.e., set $w_t = w_s = 0$) we arrive at the DLC-semi model discussed in the Results section. The corresponding ELBO is

$$\begin{aligned}
\mathcal{L} &= \sum_{\substack{t=1,j=1 \\ t,j\in\mathcal{V}}}^{T,J} \frac{w_n}{2}(-\mathbf{f}_t^\top \mathbf{G}(y_t, l^2) + \mathbf{h}_t^\top \mathbf{1}) + \sum_{\substack{t=1,j=1 \\ t,j\in\mathcal{H}}}^{T,J} \frac{w_n}{2}(-\mathbf{f}_t^\top \mathbf{G}(\mu_{ht}, l^2 + \Sigma_{htt}) + \mathbf{h}_t^\top \mathbf{1}) \\
&- \quad \log Z(x, \beta) + \log|\Sigma_n| - \log|P_v \Sigma_n P_v^\top|,
\end{aligned} \tag{S21}$$

where $\Sigma_h = (P_h \Sigma_n P_h^\top)^{-1}$ and $\mu_h = P_h \mu_n$. The first term is a conventional DLC-type cross entropy for labeled frames. The second term is a semi-supervised cross entropy for unlabeled frames. Instead of having the true marker locations for unobserved frames, we construct the Gaussian function using the 2D location output from the neural net. The second term encourages the confidence map f_θ to be unimodal to match the Gaussian approximate posterior. This semi-supervised term leads to better performance of DLC-semi compared to the original fully-supervised DLC (Figure 3).

The log normalization term $\log Z(x, \beta)$ deserves a bit of additional explanation. In this DLC-semi model the graphical model factorizes over markers j and frames t, which means that we can calculate $\log Z(x, \beta)$ directly by summing over pixels for each t and j; in the full non-factorized DGP model, this term is not directly tractable. However, in practice, dropping the $\log Z$ term from the optimization in the DLC-semi model did not affect the results significantly. Therefore we also dropped $\log Z$ from the optimization in the full DGP model.

S2.3 Implementation details

The bottlenecks of the ELBO computation in the full DGP model are $\Sigma_a \Sigma_n^{-1} \mu_n$, $\log |\Sigma_a|$, and $\text{diag}(\Sigma_a)$, where $\Sigma_a \in \mathbb{R}^{TJ \times TJ}$ and Σ_a^{-1} is a block tridiagonal matrix. All of these terms can be computed via message passing with $O(TJ^3)$ time complexity, due to the chain structure of the graphical model (and the corresponding block tridiagonal structure of the precision matrix). We used standard message passing algorithms to handle the required block tridiagonal matrix computations [46, 47, 48].

The unknown parameters in the ELBO are $\{\theta, \gamma, w_n, w_{n,q}, w_t, w_s\}$. As stated above, in practice, we fixed w_t^j to a small constant value 0.1, and used $w_s^{ij} = c/d_{ij}$, where d_{ij} is the average distance (in pixels) between targets i and j and $c > 0$ is a small scalar (independent of dataset and target indices i, j); this led to robust results without any need to fit extra parameters. We also differentiated w_n to be w_n^v and w_n^h for visible and hidden frames. Empirically, $w_n^h = 3$ and $w_n^v = 2w_n^h T/T_v$ (T_v is the number of visible frames) led to good results; this upweights the strength of labeled frames relative to unobserved frames. Moreover, θ and γ govern the neural network potential and the variational approximation, respectively. We found that setting $\gamma = \theta$ led to good results and reduced the number of parameters. Therefore, we only optimized the ELBO w.r.t. $\{\theta, w_{n,q}\}$, using coordinate ascent. We calculated the gradients for $w_{n,q}$ and θ using standard automatic differentiation tools, and performed standard stochastic gradient updates or quasi-Newton methods to estimate these parameters.

In the experiments, we compared DLC with our DGP model. We used the public DLC implementation available at this link. We ran DLC with learning rate = 0.002 for 50k iterations using 1 batch stochastic gradient descent (sgd). Notice that DLC only used the small number of training frames. We initialized our DGP model with the DLC network parameters estimated after 5k iterations, and ran the model with all frames using Limited-memory BFGS (LBFGS). The public DLC implementation trains with a 2048-channel ResNet-50 and a one-layer ConvNet. We reduced the number of channels to 200 and fixed ResNet without training the large number of network parameters, and still obtained good performance.

S3 Conditional convolutional autoencoder

S3.1 Implementation details

We fit conditional convolutional autoencoders (conditional CAEs) on 192x192 grayscale images from [23]. In addition, we used 4 markers output by DLC/DGP: left paw, right paw, tongue, and nose. To condition the encoder network on these values we turned each marker into a one-hot 2D array and concatenated these with the corresponding frame, so that the input to the encoder was of size $(192, 192, 5)$. To condition the decoder network on these values we first centered the marker values by subtracting their median (computed over the entire dataset) and then concatenated these values to the latents before feeding them into the decoder. See Table S1 for network architecture details. We trained the autoencoders by minimizing the MSE between original and reconstructed frames using the Adam optimizer [49] with a learning rate of 10^{-4}, a batch size of 100, and no regularization. Models were trained for 300 epochs.

Layer	Type	Channels	Kernel Size	Stride Size	Zero Padding	Output Size
0	conv	32	(5, 5)	(2, 2)	(1, 2, 1, 2)	(96, 96, 32)
1	conv	64	(5, 5)	(2, 2)	(1, 2, 1, 2)	(48, 48, 64)
2	conv	128	(5, 5)	(2, 2)	(1, 2, 1, 2)	(24, 24, 128)
3	conv	256	(5, 5)	(2, 2)	(1, 2, 1, 2)	(12, 12, 256)
4	conv	512	(5, 5)	(2, 2)	(1, 2, 1, 2)	(6, 6, 512)
5	dense	*N*	NA	NA	NA	(1, 1, *N*)
5	concatenate	NA	NA	NA	NA	(1, 1, *N*+2*M*)
6	dense	36	NA	NA	NA	(1, 1, 36)
7	reshape	NA	NA	NA	NA	(6, 6, 1)
8	conv transpose	256	(5, 5)	(2, 2)	(1, 2, 1, 2)	(12, 12, 256)
9	conv transpose	128	(5, 5)	(2, 2)	(1, 2, 1, 2)	(24, 24, 128)
10	conv transpose	64	(5, 5)	(2, 2)	(1, 2, 1, 2)	(48, 48, 64)
11	conv transpose	32	(5, 5)	(2, 2)	(1, 2, 1, 2)	(96, 96, 32)
12	conv transpose	1	(5, 5)	(2, 2)	(1, 2, 1, 2)	(192, 192, 1)

Table S1: Conditional CAE architecture for the mouse-wheel dataset using *N* latents and *M* markers (each with an x and y value, for a total of *2M* marker dimensions). Kernel size and stride size are defined as (x pixels, y pixels); padding size is defined as (left, right, top, bottom); output size is defined as (x pixels, y pixels, channels).

S3.2 Disentangling analysis

The disentangling analyses presented in Figure 5 require fixing some inputs to the network while varying others. Below we describe this manipulation in more detail. We performed these manipulations on 2-latent networks to make visualization in the latent space easier.

Manipulating markers. We chose a random test frame and varied the x/y coordinates for a specific marker (left paw). The limits of the x/y values were the 10^{th} (minimum) and 90^{th} (maximum) percentiles of the DGP outputs on the test set for the specified marker. We did not allow different limits for the DLC/DGP networks, in order to make the comparison more direct. After choosing x/y values for the specified marker we converted these into a one-hot 2D array, as with the other (unchanged) markers from the chosen frame. These one-hot 2D arrays were concatenated with the original frame and then fed into the CAE encoder to produce the latents. The latents were then concatenated with the median-subtracted marker values (one of which is being changed, the rest of which stay the same). This vector was then pushed through the decoder network to produce the reconstructions.

Note that in this conditional architecture the latents themselves are marker-dependent, so are not truly held fixed. We also fit conditional CAE architectures where just the decoder was conditioned on the markers, and the encoder only used the frame as input. We found the results from the disentangling analysis to be qualitatively similar, though reconstructions generally looked cleaner with the architectures that incorporated conditioning in both the encoder and decoder networks (data not shown).

Manipulating latents. We chose a random test frame and this time varied the latents while keeping the marker values fixed. Similar to above, we used the 10^{th} (minimum) and 90^{th} (maximum) percentiles of the latents on the test set as limits (this time allowing different limits for each DLC/DGP architecture). We then concatenated the new latent values with the marker values from the original frame, and pushed this vector through the decoder network to produce the reconstructions.

Quantifying disentanglement. To quantify the disentanglement results from Figure 5 (center, right panels), we chose the left paw as a (tracked) target of interest that should ideally not undergo large changes when manipulating the latents. If disentanglement is high (which we desire), the differences between the generated paw and the original paw should be small. For each image generated from the

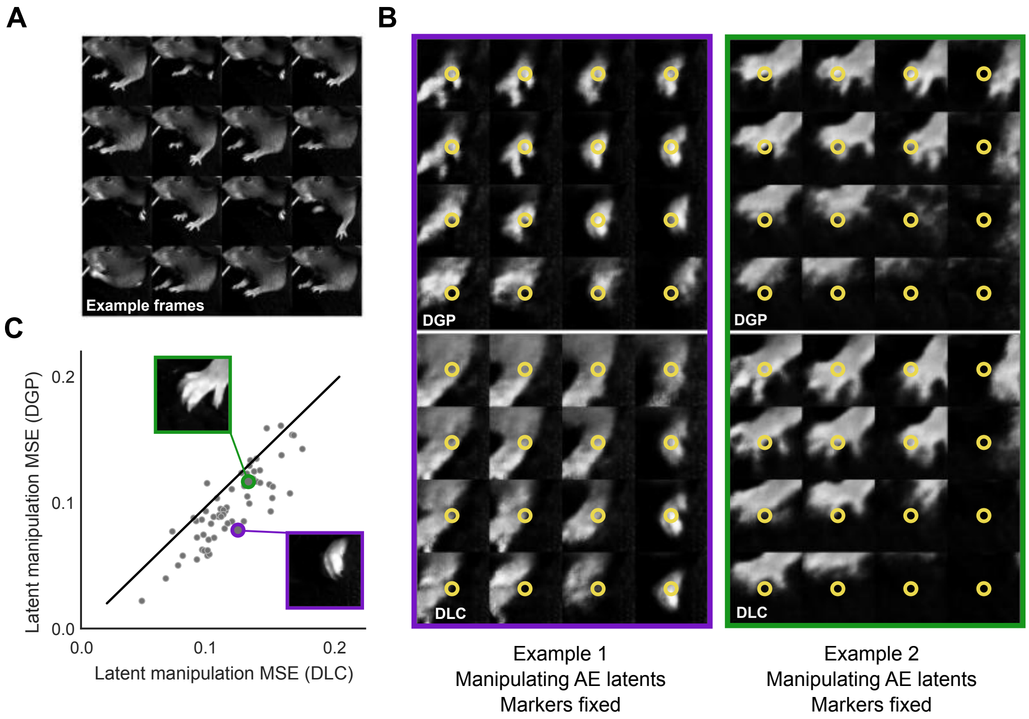

Figure S1: **Conditional CAEs display better disentangling properties when using DGP rather than DLC markers.** **A**: A total of 64 representative frames were chosen from the test set by performing k-means on the 2-latent unconditional AE latents; pictured are 16 of those frames, which include the left paw (the target feature of this analysis) in multiple positions. **B**: For each frame, the markers are held fixed while the 2 latents are varied, producing a matrix of generated images. For each generated image we crop around the left paw. The paw position should remain stable throughout the latent manipulation if there is good disentanglement between latents and markers. Latent manipulations of two example frames are shown (in columns) for networks trained using DGP or DLC markers (in rows). Yellow circles indicate the left paw marker location. **C**: We compute MSE between each generated frame and the original frame around the paw, and average over all latent values. Small MSE indicates desired stability in the tracked paw. For almost all frames the DGP-trained network exhibits lower MSE, thus demonstrating a higher degree of disentangling.

latent manipulation, we take a small crop around the original location of the left paw and compute the MSE between this generated paw and the original paw.

We repeat this process for an unbiased sample of test frames. To obtain these frames we performed k-means clustering on the unconditional CAE latents. From each of 64 clusters we take the frame that is closest to the cluster centroid (Figure S1A), and perform the process of generating frames by evenly sampling the latent space - 4 grid points along each of 2 dimensions, for a total of 16 generated frames. We compute the MSE in crops around the labeled paw position as described above (Figure S1B, C). We find that on average the CAE-DGP networks have lower MSE, indicating that these disentanglement results generalize to many other paw positions (and therefore marker values) found in this dataset.

S4 Results on additional datasets

We first show the comparison between DLC and DGP on the mouse-reach, fly-run, twomice-top-down, and fish-swim datasets (Table S2) in Figures S2-S5. In each case, results were similar to those seen in Figure 2.

Table S2: Dataset summary.

Dataset	Brief Description	Dimensions (x, y, t)	Number of labeled frames
mouse-wheel [23]	moving a wheel	(374, 450, 1000)	55
mouse-reach [50]	grabbing a stick	(747, 832, 256)	15
fly-run [51]	running on a ball	(600, 600, 1210)	13
twomice-top-down*	freely moving	(480, 640, 1364)	20
fish-swim [52]	freely swimming	(471, 475, 2000)	20

(*) unpublished

The remaining figures (S6-S10) show the same datasets, but with additional panels and traces illustrating the output of the intermediate models DLC-semi and DGP-NN, along with a reimplementation of DLC (labeled as DLC-ours) using the same optimizer, 200 ResNet channels, etc., as in our implementation of DGP. (We include these comparisons to rule out minor differences caused by these distinct implementation details.) Results are consistent with Figure 3: DGP-NN and DGP achieve similar performance across all five datasets; each tends to outperform DLC-semi, which in turn outperforms either implementation of DLC. Full videos can also be found here.

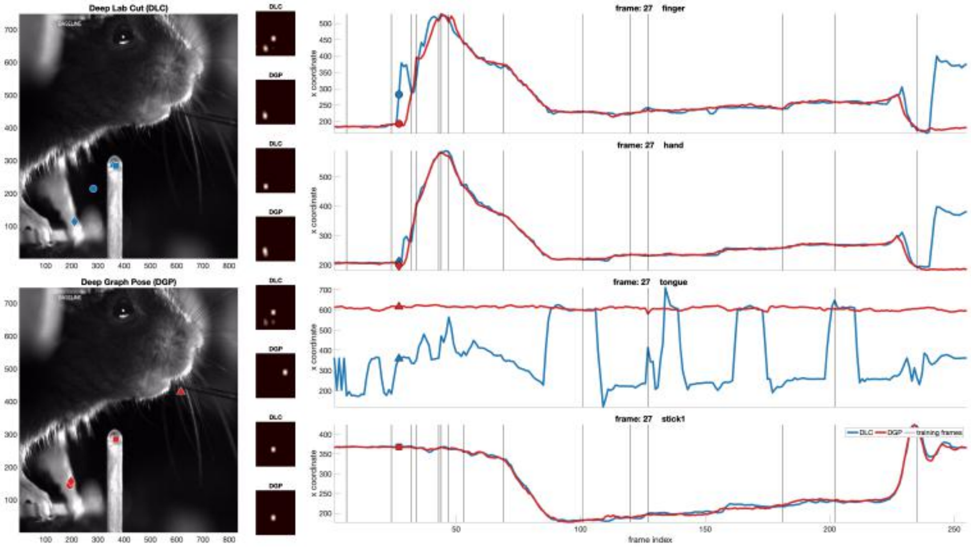

Figure S2: Comparing DeepLabCut (DLC) versus Deep Graph Pose (DGP) on the mouse reaching dataset from [4]. Conventions and conclusions as in Figure 2.

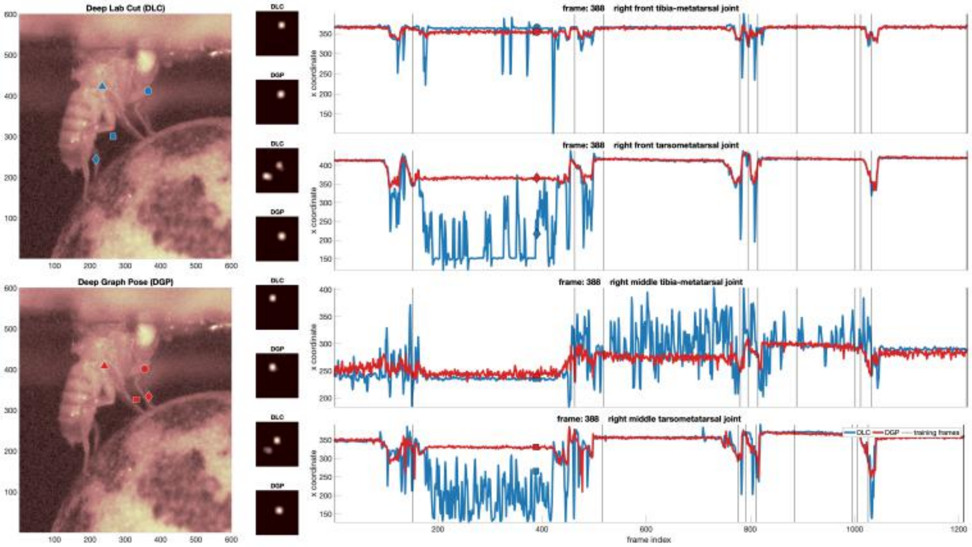

Figure S3: Comparing DeepLabCut (DLC) versus Deep Graph Pose (DGP) on the fly ball-turning dataset from [51]. Conventions and conclusions as in Figure 2.

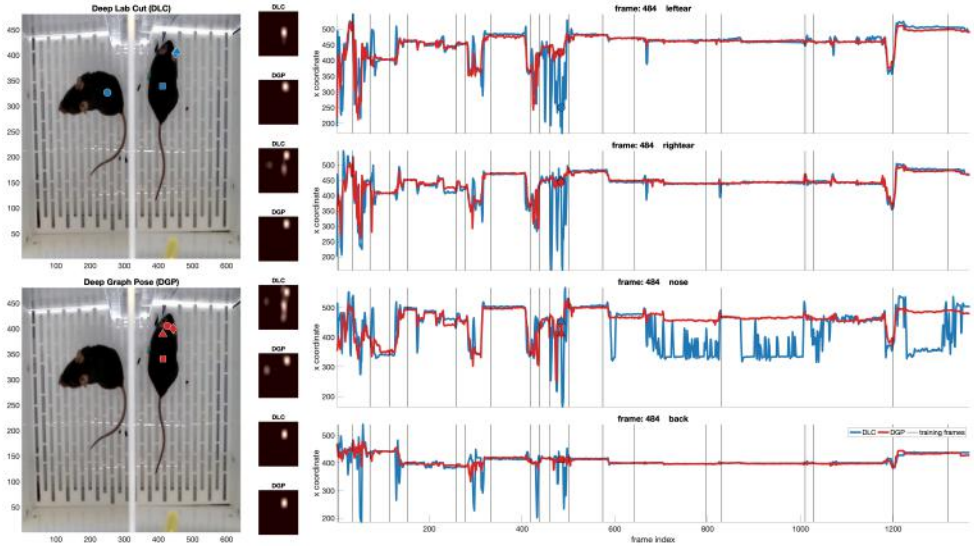

Figure S4: Comparing DeepLabCut (DLC) versus Deep Graph Pose (DGP) on the twomice-top-view dataset with two mice. Conventions and conclusions as in Figure 2.

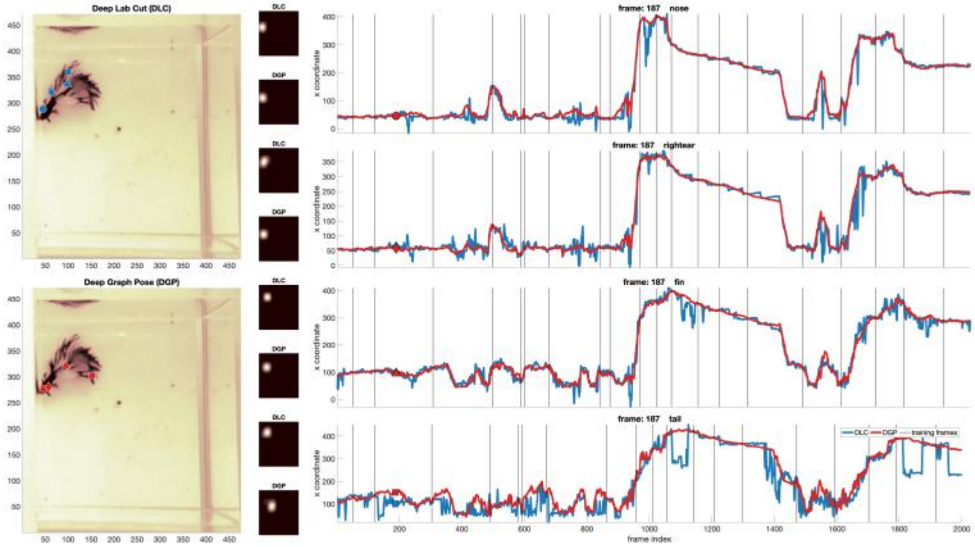

Figure S5: Comparing DeepLabCut (DLC) versus Deep Graph Pose (DGP) on the swimming fish dataset from [52]. Conventions and conclusions as in Figure 2.

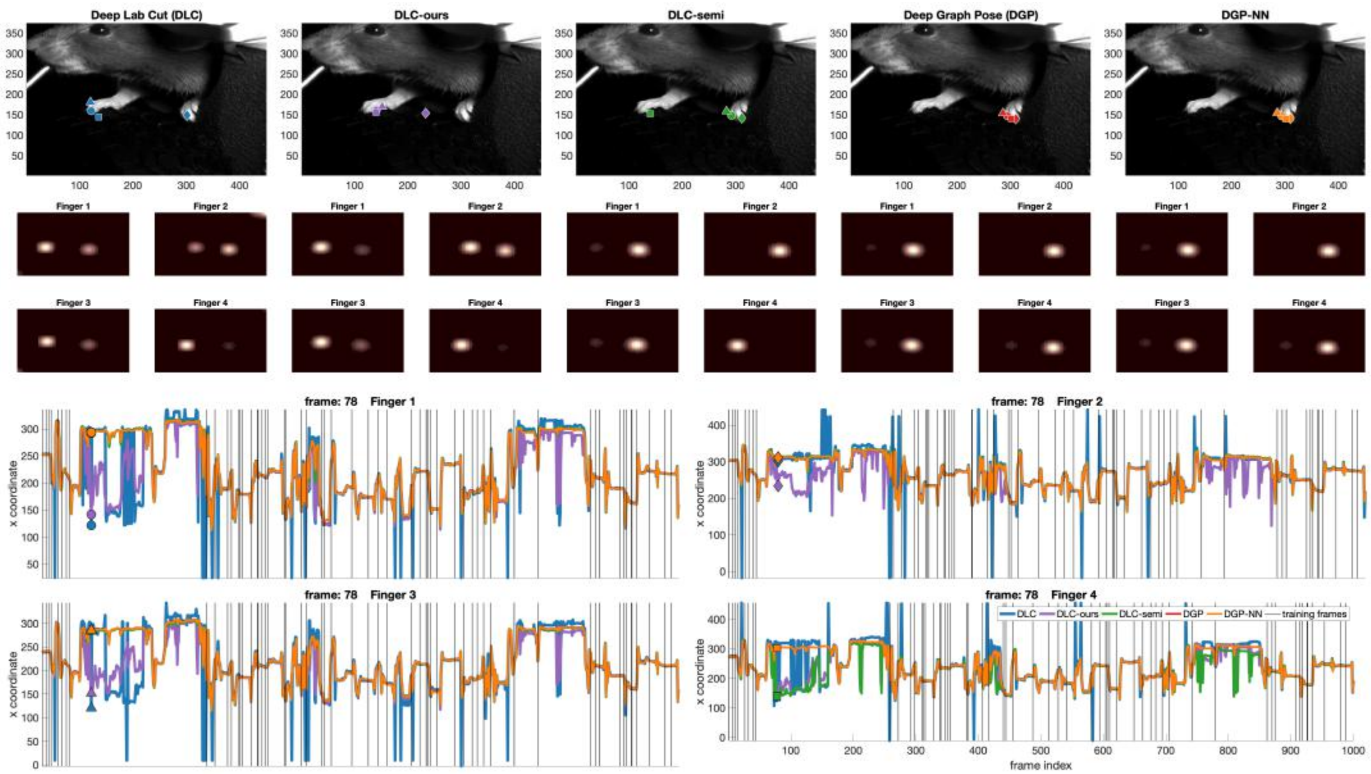

Figure S6: Expanded view of Figure 2: comparing DLC, DLC-semi, DGP, and DGP-NN on the wheel-turning dataset from [23]. DGP-NN and DGP have very similar outputs; DLC-semi improves the accuracy of DLC, but DGP improves this accuracy further (visible here in frame 240 or so in the “finger” trace).

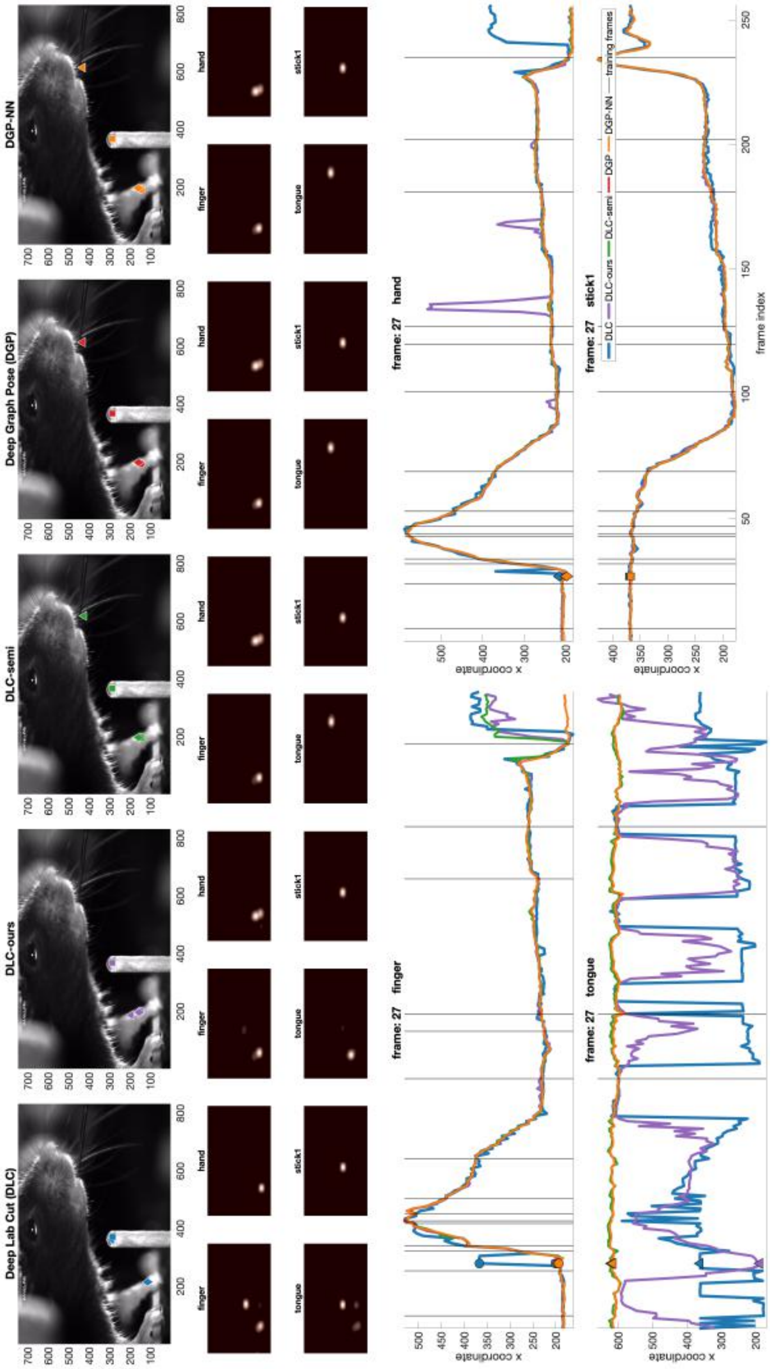

Figure S7: Expanded view of Figure S2; conventions and conclusions as in Figure S7

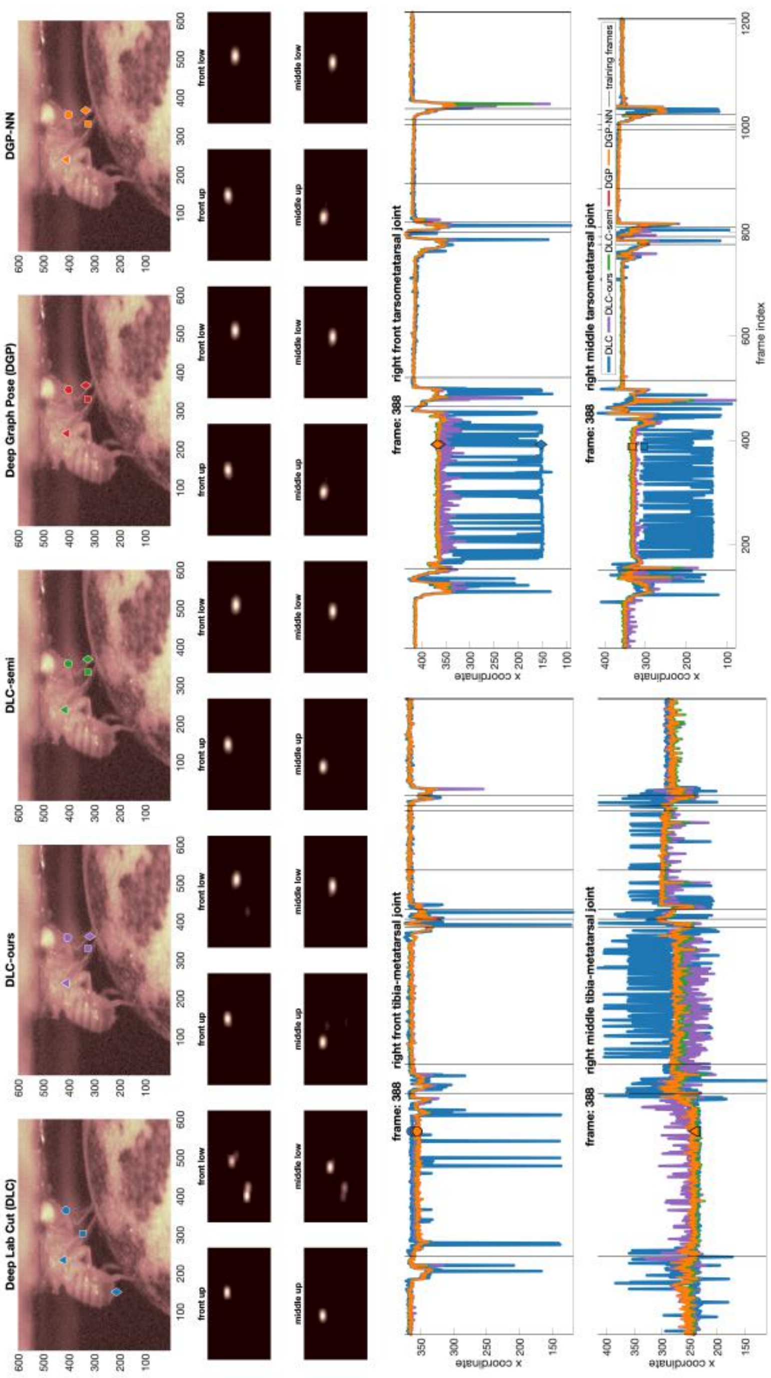

Figure S8: Expanded view of Figure S3; conventions and conclusions as in Figure S6. The abbreviations over the confidence maps are: front up (right front tibia-metatarsal joint), front low (right front tarsometatarsal joint), middle up (right middle tibia-metatarsal joint), middle low (right middle tarsometatarsal joint).

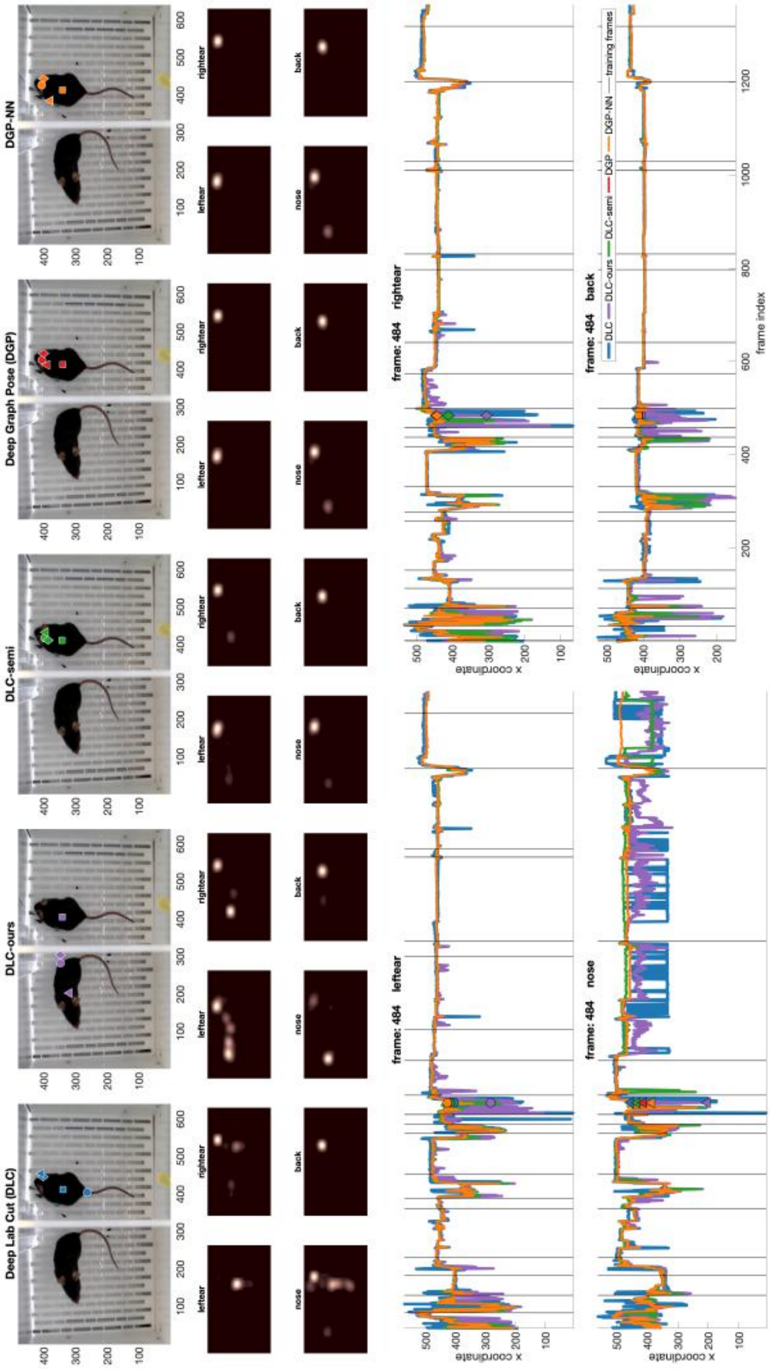

Figure S9: Expanded view of Figure S4; conventions and conclusions as in Figure S6.

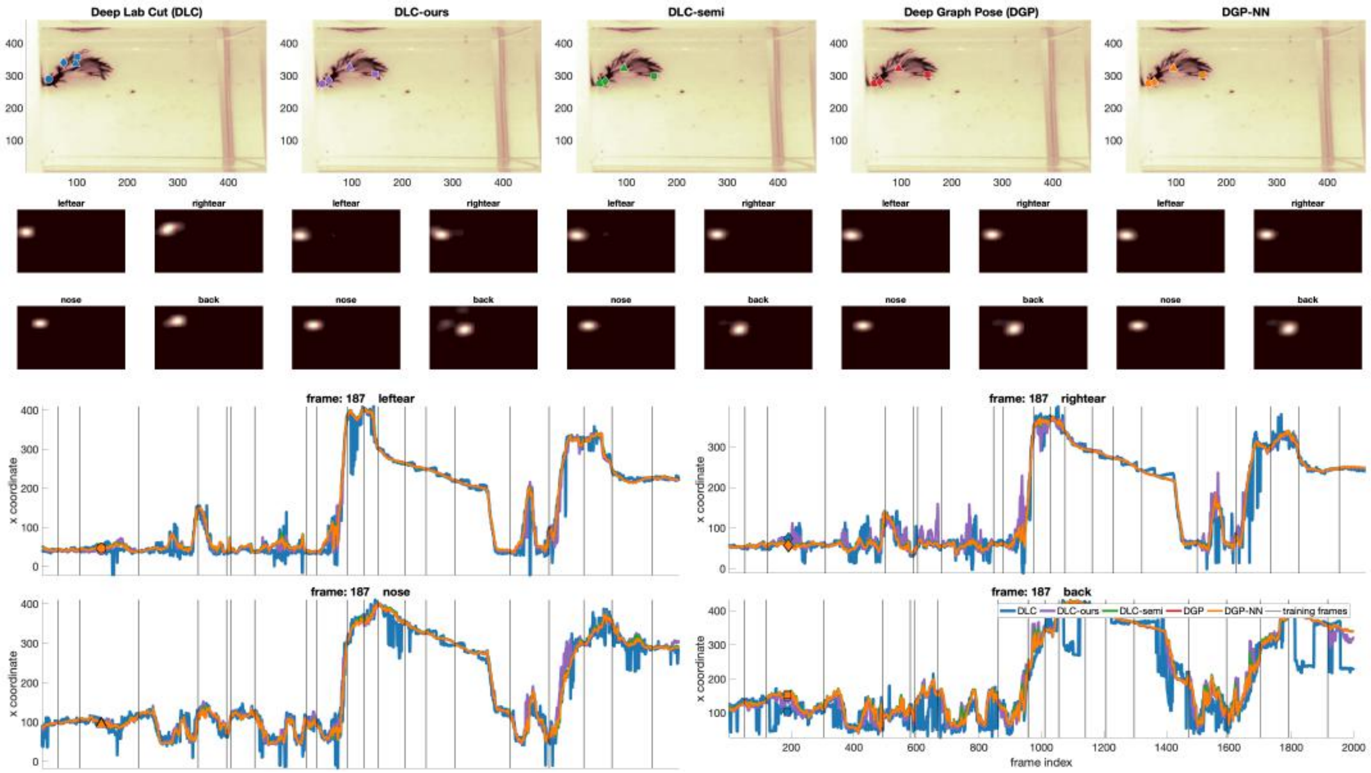

Figure S10: Expanded view of Figure S5; conventions and conclusions as in Figure S6.