Fast and robust animal pose estimation

Jacob M. Graving[1,2,3,*], Daniel Chae[4], Hemal Naik[1,2,3,5], Liang Li[1,2,3], Benjamin Koger[1,2,3], Blair R. Costelloe[1,2,3], Iain D. Couzin[1,2,3,*]

***For correspondence:**
jgraving@gmail.com;
icouzin@orn.mpg.de

[1]Department of Collective Behaviour, Max Planck Institute for Ornithology, 78464 Konstanz, Germany; [2]Chair of Biodiversity and Collective Behaviour, University of Konstanz, 78464 Konstanz, Germany; [3]Centre for the Advanced Study of Collective Behaviour, University of Konstanz, 78464 Konstanz, Germany; [4]Department of Computer Science, Princeton University, 08544 Princeton, NJ, USA; [5]Chair for Computer Aided Medical Procedures, Technische Universität München, 80333 Munich, Germany

Abstract Quantitative behavioral measurements are important for answering questions across scientific disciplines—from neuroscience to ecology. State-of-the-art deep-learning-based methods offer major advances in data quality and detail by allowing researchers to automatically estimate locations of an animal's body parts directly from images or videos. However, currently-available animal pose estimation methods have limitations in speed, robustness, and usability. Here we introduce an open-source software toolkit, *DeepPoseKit*, that addresses these problems. Using modern desktop hardware, our methods perform real-time measurements at ∼30–110-Hz with offline performance >1000-Hz—approximately 2–6× faster than current methods. We achieve these results while only increasing average error <0.5-pixels compared to the most-accurate methods currently available. We demonstrate the versatility of our approach with multiple challenging animal pose estimation tasks in laboratory and field settings—including groups of interacting individuals. Our work reduces barriers to using advanced tools for measuring behavior and has broad applicability across the behavioral sciences.

Introduction

Understanding the relationships between individual behavior, brain activity (reviewed by ***Krakauer et al. 2017***), and collective and social behaviors (***Rosenthal et al., 2015***; ***Strandburg-Peshkin et al., 2013***; ***Jolles et al., 2017***; ***Klibaite et al., 2017***; ***Klibaite and Shaevitz, 2019***) is a central goal of the behavioral sciences—a field that spans disciplines from neuroscience to psychology, ecology, and genetics. Measuring and modelling behavior is key to understanding these multiple scales of complexity, and, with this goal in mind, researchers in the behavioral sciences have begun to integrate theory and methods from physics, computer science, and mathematics (***Anderson and Perona, 2014***; ***Berman, 2018***; ***Brown and De Bivort, 2018***). A cornerstone of this interdisciplinary revolution is the use of state-of-the-art computational tools, such as computer vision algorithms, to automatically measure locomotion and body posture (***Dell et al., 2014***). Such a rich description of animal movement then allows for modeling, from first principles, the full behavioral repertoire of animals (***Berman et al., 2014a***, ***2016***; ***Wiltschko et al., 2015***; ***Johnson et al., 2016***; ***Todd et al., 2017***; ***Klibaite et al., 2017***; ***Markowitz et al., 2018***; ***Klibaite and Shaevitz, 2019***; ***Costa et al., 2019***). Tools for automatically measuring animal movement represent a vital first step toward developing unified theories of behavior across scales (***Berman, 2018***; ***Brown and De Bivort, 2018***). Therefore, technical factors like scalability, robustness, and usability are issues of critical importance, especially as researchers across disciplines begin to increasingly rely on these methods.

Two of the most recent contributions to the growing toolbox for quantitative behavioral analysis are from ***Mathis et al.*** (***2018***) and ***Pereira et al.*** (***2019***), who make use of a popular type of machine learning model known as *convolutional neural networks*, or *CNNs* (***LeCun et al. 2015***; Appendix 1), to automatically measure detailed representations of animal posture—structural *keypoints*, or *joints*, on the animal's body—directly from images and without markers. While these methods offer a major advance over conventional methods with regard to data quality and detail, they have disadvantages in terms of speed, robustness, and ease of use, which may limit their practical applications. To address these problems, we introduce a new software toolkit called *DeepPoseKit* that is fast, robust, and easy-to-use. We run experiments using multiple datasets to compare our methods to those from ***Mathis et al.*** (***2018***) and ***Pereira et al.*** (***2019***) and find that our approach offers considerable performance improvements. These results also demonstrate the flexibility of our methods in both the laboratory and the field, and our work is widely applicable across a range of scientific disciplines.

Animal pose estimation using deep learning

In the past, conventional methods for measuring posture with computer vision relied on species-specific algorithms (***Uhlmann et al., 2017***), highly-specialized or restrictive experimental setups (***Mendes et al., 2013***; ***Kain et al., 2013***), attaching intrusive physical markers to the study animal (***Kain et al., 2013***), or some combination thereof. These methods also typically required expert computer-vision knowledge to use, were limited in the number or type of body parts that could be tracked (***Mendes et al., 2013***), involved capturing and handling the study animals to attach markers(***Kain et al., 2013***)—which is not possible for many species—and despite best efforts to minimize human involvement, often required manual intervention to correct errors (***Uhlmann et al., 2017***). All of these methods were built to work for a small range of conditions and typically required considerable effort to adapt to novel contexts.

In contrast to conventional computer-vision methods, modern deep-learning–based methods can be used to achieve human-level accuracy in nearly any context by manually annotating data (Figure 1)—known as a *training set*—and training a general-purpose image-processing algorithm—a convolutional neural network or CNN—to automatically estimate the locations of an animal's body parts directly from images (Figure 2). State-of-the-art machine learning methods, like CNNs, use these training data to parameterize a model of the relationship between a set of input data—i.e. images—and the desired output distribution—i.e. posture keypoints. After adequate training, a model can be used to make predictions on previously-unseen data from the same dataset—inputs that were not part of the training set—which is known as *inference*. In other words, these models are able to generalize human-level expertise at scale after having been trained on only a relatively small number of examples. We provide more detailed background information on using CNNs for pose estimation in Appendices 1–6.

Similar to conventional pose estimation methods, the task of implementing deep-learning models in software and training them on new data is complex and requires expert knowledge. However, in most cases, once the underlying model and training routine are implemented, a high-accuracy pose estimation model for a novel context can be built with minimal modification—often just by changing the training data. With a simplified toolkit and high-level software interface designed by an expert, even scientists with limited computer-vision knowledge can begin to apply these methods to their research. Once the barriers for implementing and training a model are sufficiently reduced, the main bottleneck for using these methods becomes collecting an adequate training set—a non-expert but labor-intensive task made less time-consuming by techniques described in Appendix 2.

Mathis et al. (***2018***) and ***Pereira et al.*** (***2019***) were the first to popularize the use of CNNs for animal pose estimation. These researchers built on work from the human pose estimation literature (e.g. ***Andriluka et al. 2014***; ***Insafutdinov et al. 2016***; ***Newell et al. 2016***) using a type of

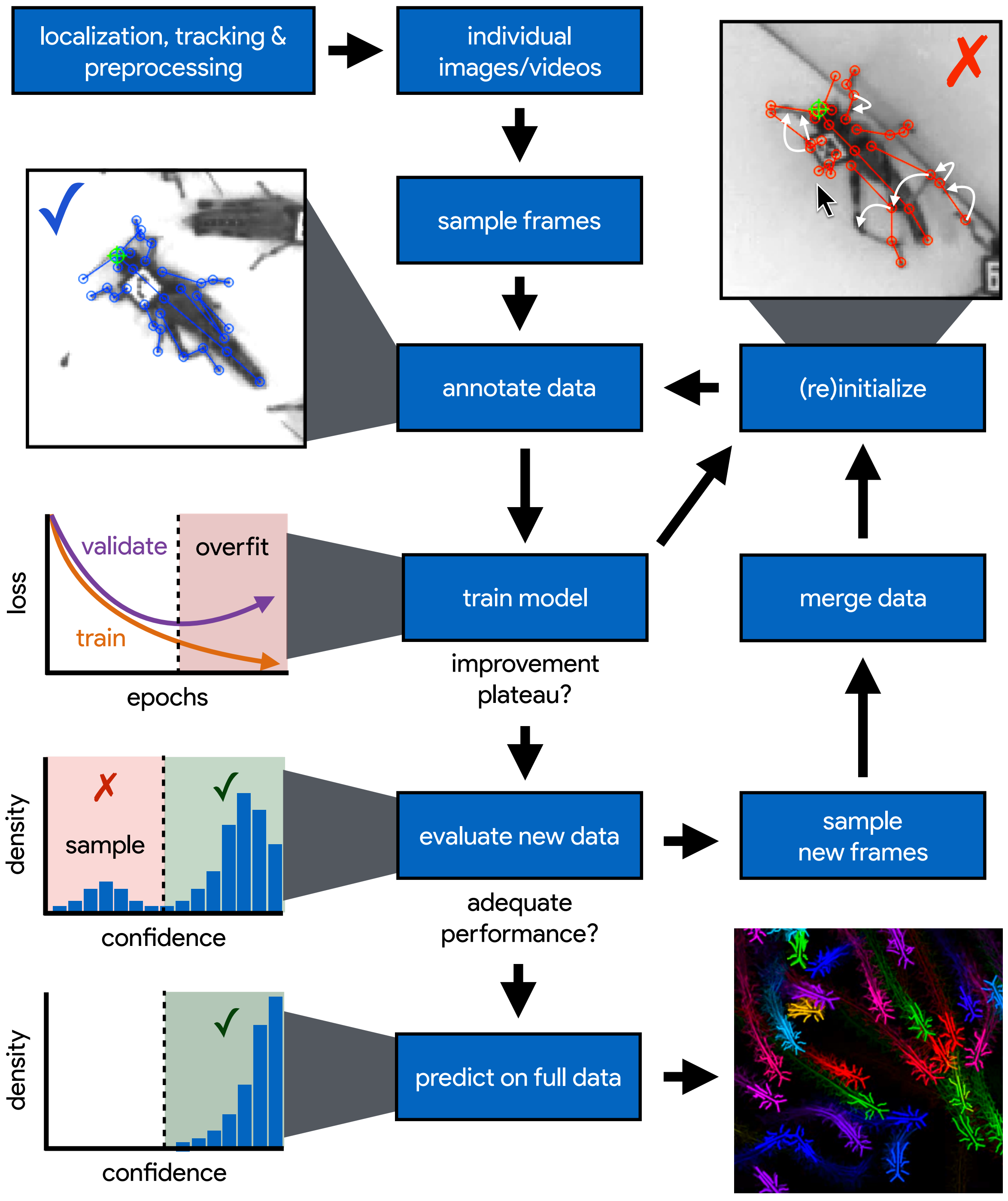


Figure 1. An illustration of the workflow for DeepPoseKit. An initial image set is annotated and then iteratively updated using the active learning approach developed by *Pereira et al.* (*2019*). The training set is updated with new training examples selected based on the current model performance. This process is repeated as necessary until performance is adequate. The pose estimation model can then be used to make predictions for the full data set, which can then be used for further analysis.

Figure 1–video 1. A visualization of the posture data output for a group of locusts (5× speed).

fully-convolutional neural network or *F-CNN* (***Long et al. 2015***; Appendix 3) known as an *encoder-decoder* model (Appendix 3 Box 1). These models are used to measure animal posture by training the network to transform images into probabilistic estimates of keypoint locations, known as *confidence maps* (shown in Figure 2), that describe the body posture for one or more individuals. These confidence maps are processed to produce the 2-D spatial coordinates of each keypoint, which can then be used for further analysis. The methods from ***Mathis et al.*** (***2018***) can be used to estimate posture for single individuals—known as *individual pose estimation*—or multiple individuals simultaneously—known as *multiple pose estimation*. In contrast, the methods from ***Pereira et al.*** (***2019***) are limited to individual pose estimation. The methods we present in this paper are technically limited to individual pose estimation; however, we successfully remove this limitation and extend our methods to groups of interacting individuals by first localizing and tracking individuals using additional software (see Appendix 4 for discussion).

Mathis et al. (***2018***) use a previously-published pose estimation model known as *DeeperCut* (***Insafutdinov et al., 2016***), which is built on the popular *ResNet* architecture (***He et al., 2016***)—a state-of-the-art model for image classification. This choice is advantageous because the use of a popular architecture allows for using a pre-trained model to improve performance, known as *transfer learning* (***Pratt 1993***; Appendix 2), but it is also disadvantageous as the model is *overparameterized* with >25 million parameters. Overparameterization allows the model to make accurate predictions at the cost of unnecessarily slow inference. More recent work from ***Mathis and Warren*** (***2018***) has shown that the inference speed for DeeperCut (***Insafutdinov et al., 2016***) can be improved at the expense of increased prediction error. The original methods from ***Mathis et al.*** (***2018***) may also be difficult to use for beginners, but recent updates to the software have attempted to address this problem (see ***Nath et al. 2018***).

With regard to model design, ***Pereira et al.*** (***2019***) take the opposite approach of implementing a custom network architecture that attempts to limit model complexity and overparameterization (Appendix 6), which they call *LEAP* (LEAP Estimates Animal Pose). LEAP is advantageous because it is explicitly designed for fast inference but has disadvantages such as a lack of robustness to data variance, like rotations or shifts in lighting, and an inability to generalize to new experimental setups. Additionally, to achieve maximum performance, the LEAP framework requires computationally expensive preprocessing that is not practical for many datasets, which makes it unsuitable for a wide range of experiments (see Appendix 6 for more details). The software from ***Pereira et al.*** (***2019***) is generally easy to install and use, but much of the interface is written in MATLAB (The Mathworks Inc.), which requires an expensive and restrictive software license.

Together the methods from ***Mathis et al.*** (***2018***) and ***Pereira et al.*** (***2019***) represent the two extremes of a phenomenon known as the *speed-accuracy trade-off* (***Huang et al., 2017b***)—an active area of research in the machine learning literature. ***Mathis et al.*** (***2018***) prioritize accuracy over speed by using a large overparameterized model (***Insafutdinov et al., 2016***), and ***Pereira et al.*** (***2019***) prioritize speed over accuracy by using a smaller less-robust model. While this speed-accuracy trade-off can limit the capabilities of CNNs, there has been extensive work to make these models more efficient without impacting performance (e.g. ***Chollet 2017***; ***Huang et al. 2017a***; ***Sandler et al. 2018***). To address the limitations of this trade-off, we apply recent developments from the machine learning literature and provide an effective solution to the problem. In the case of F-CNN models used for pose estimation, improvements in efficiency and robustness have been made through the use of *multi-scale inference* (Appendix 3 Box 1) and by increasing the number of connections between layers in the model (Appendix 3 Figure 1)—both of which we incorporate into our methods.

Methods and Results

Here we introduce fast, flexible, and robust pose estimation methods with a software interface that emphasizes usability. Our methods build on the state-of-the-art for individual pose estimation (***Newell et al. 2016***; Appendix 5), convolutional regression models (***Jégou et al. 2017***; Appendix 3 Box 1), and conventional computer vision algorithms (***Guizar-Sicairos et al., 2008***) to improve

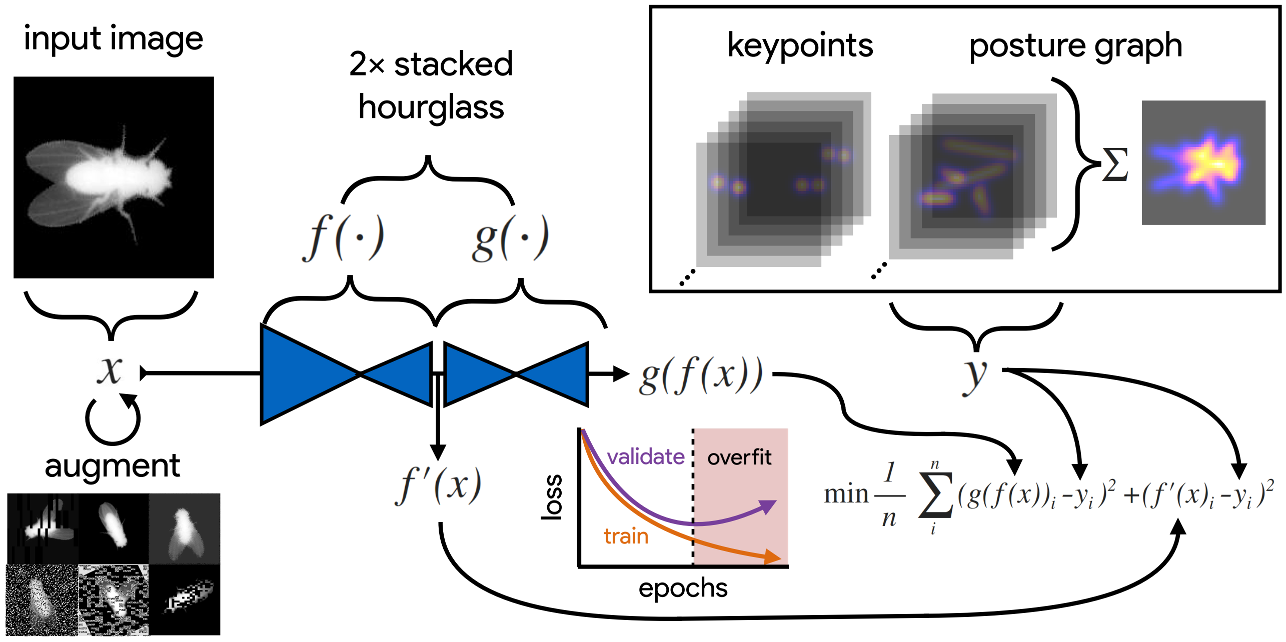


Figure 2. An illustration of the model training process for DeepPoseKit. Input images x (**top-left**) are augmented (**bottom-left**) with various spatial transformations (rotation, translation, scale, etc.) followed by noise transformations (dropout, additive noise, blurring, contrast, etc.) to improve the robustness and generalization of the model. The ground truth annotations are then transformed with matching spatial augmentations (not shown for the sake of clarity) and used to draw the confidence maps y for the keypoints and hierarchical posture graph (**top-right**). The images x are then passed through the network to produce a multidimensional array $g(f(x))$—a stack of images corresponding to the keypoint and posture graph confidence maps for the ground truth y. Mean squared error between the outputs for both hourglasses $g(f(x))$ and $f'(x)$ and the ground truth data y is then minimized (**bottom-right**), where $f'(x)$ indicates a subset of the output from $f(x)$—only those feature maps being optimized to reproduce the confidence maps for the purpose of intermediate supervision (Appendix 5). The loss function is minimized until the validation loss stops improving—indicating that the model has converged or is starting to overfit to the training data.

model efficiency and achieve faster, more accurate results on multiple challenging pose estimation tasks. We developed two model implementations—including a new model architecture that we call *Stacked DenseNet*—and a new method for processing confidence maps called *subpixel maxima* that provides fast and accurate results with subpixel precision—even at low resolutions. We also discuss a modification to incorporate the global geometry between keypoints when training pose estimation models that increases accuracy without decreasing speed. We ran experiments to optimize our approach and compared our models to those from ***Insafutdinov et al.*** (***2016***) (DeeperCut)—the model used by ***Mathis et al.*** (***2018***)—and ***Pereira et al.*** (***2019***) (LEAP) using three image datasets filmed in the laboratory and the field—including multiple interacting individuals that were first localized and cropped from larger, multi-individual images.

An end-to-end pose estimation framework

We provide a full-featured, extensible, and easy-to-use software package that is written entirely in the Python programming language (Python Software Foundation) and is built on the popular Keras deep-learning package (***Chollet et al., 2015***)—using Tensorflow as a backend (***Abadi et al., 2015***). Our software is a complete, end-to-end pipeline (Figure 1) with a custom GUI (graphical user interface) for creating annotated training data with *active learning* similar to ***Pereira et al.*** (2019; Appendix 2), as well as an interface for *data augmentation* (***Jung 2018***; Appendix 2; shown in Figure 2), model training and evaluation (Figure 2; Appendix 1), and running inference on new data. We designed our high-level programming interface to be a testbed for experimentation, allowing the user to go from idea to execution as quickly as possible, and we organized our software into a Python module called *DeepPoseKit*. The code, documentation, and examples for our entire software package are freely available at https://github.com/jgraving/deepposekit under a permissive open-source license.

Our pose estimation models

To achieve the goal of "fast animal pose estimation" introduced by ***Pereira et al.*** (***2019***), while also wanting to achieve the robust predictive power of models like DeeperCut (***Insafutdinov et al., 2016***), we implemented two fast pose estimation models that extend the current state-of-the-art for individual pose estimation introduced by ***Newell et al.*** (***2016***) and the current state-of-the art for convolutional regression from ***Jégou et al.*** (***2017***). Our model implementations use fewer parameters than both DeeperCut (***Insafutdinov et al., 2016***) and LEAP (***Pereira et al., 2019***) while simultaneously removing many of the limitations of these architectures.

In order to limit overparameterization while minimizing performance loss, we designed our models to allow for multi-scale inference (Appendix 3 Box 1) while optimizing our model hyperparameters for efficiency. Our first model is a novel implementation of *FC-DenseNet* from ***Jégou et al.*** (2017; Appendix 3 Box 1) arranged in a stacked configuration similar to ***Newell et al.*** (2016; Appendix 5). We call this new model Stacked DenseNet, and to the best of our knowledge, this is the first implementation of this architecture in the literature—for pose estimation or otherwise. Further details for this model are available in Appendix 8. Our second model is a modified version of the *Stacked Hourglass* model from ***Newell et al.*** (2016; Appendix 5) with hyperparameters that allow for changing the number of filters in each convolutional block to constrain the number of parameters—rather than using 256 filters for all layers as described in ***Newell et al.*** (***2016***).

Subpixel keypoint prediction on the GPU

In addition to implementing our efficient pose estimation models, we developed a new method to process the model outputs to allow for faster, more accurate predictions. When using a fully-convolutional posture estimation model, the confidence maps produced by the model must be converted into coordinate values for the predictions to be useful, and there are typically two choices for making this conversion. The first is to move the confidence maps out of GPU memory and post-process them on the CPU. This solution allows for easy, flexible, and accurate calculation of the coordinates with subpixel precision. However, CPU processing is not ideal because moving large arrays of data between the GPU and CPU is costly, and computation on the CPU is generally slower. The other option is to directly process the confidence maps on the GPU and then move the coordinate values from the GPU to the CPU. This approach usually means converting confidence maps to integer coordinates based on the row and column index of the global maximum for each confidence map. However, this means that, to achieve a precise estimation, the confidence maps should be predicted at the full resolution of the input image, or larger, which slows down inference speed.

As an alternative to these two strategies, we introduce a new GPU-based convolutional layer that we call *subpixel maxima*. This layer uses the fast, efficient, image registration algorithm introduced by ***Guizar-Sicairos et al.*** (***2008***) to translationally align a centered two-dimensional Gaussian filter to each confidence map via Fourier-based convolution. The translational shift between the filter and each confidence map allows us to calculate the coordinates of the global maxima with high speed and subpixel precision. This technique allows for accurate predictions even if the model's confidence maps are dramatically smaller than the resolution of the input image.

Learning global relationships between keypoints

Minimizing extreme prediction errors is important to prevent downstream effects on any further behavioral analysis —especially in the case of analyses based on time-frequency transforms like those from ***Berman et al.*** (***2014a***, 2016); ***Klibaite et al.*** (***2017***); ***Todd et al.*** (***2017***); ***Klibaite and Shaevitz*** (***2019***) and ***Pereira et al.*** (***2019***) where high magnitude errors can cause inaccurate behavioral classifications. One way to minimize extreme errors when estimating posture is to incorporate multiple spatial scales when making predictions. Our pose estimation models are *implicitly* capable of using information from multiple spatial scales (see Appendix 3 Box 1), but there is no *explicit* signal that optimizes the model to take advantage of this information when making predictions.

To remedy this, we modified the model's output to predict, in addition to the keypoint locations, a hierarchical graph of edges describing the global geometry between keypoints—similar to the part affinity fields described by ***Cao et al.*** (***2017***). This was achieved by adding an extra set of confidence maps to the output where edges in the postural graph are represented by Gaussian-blurred lines the same width as the Gaussian peaks in the keypoint confidence maps. Our posture graph output then consists of four levels: (1) a set of confidence maps for the smallest limb segments in the graph (e.g. foot to ankle, knee to hip, etc.; Figure 2), (2) a set of confidence maps for individual limbs (e.g. left leg, right arm, etc.; Figure 3), (3) a map with the entire postural graph, and (4) a fully-integrated map that incorporates the entire posture graph and confidence peaks for all of the joint locations (Figure 2). Each level of the hierarchical graph is built from lower levels in the output, which forces the model to learn correlated features across multiple scales when making predictions.

Experiments and model comparisons

We ran three experiments to test and optimize our approach. First, we compared our new subpixel maxima layer to an integer-based global maxima with downsampled outputs ranging from 1× to $\frac{1}{16}\times$ the input resolution using our Stacked DenseNet model. Next, we tested if training a Stacked DenseNet model to predict the global geometry of the posture graph improves accuracy. Finally, we compared our model implementations of Stacked Hourglass and Stacked DenseNet to the models from ***Pereira et al.*** (***2019***) and ***Insafutdinov et al.*** (***2016***), which we also implemented in our framework (see Appendix 8 for details on our implementation of ***Insafutdinov et al. 2016***). When benchmarking these models we incorporated the relevant improvements from our experiments—including subpixel maxima and predicting global geometry between keypoints—unless otherwise noted.

Datasets

We performed experiments using the vinegar or "fruit" fly (*Drosophila melanogaster*) dataset (Figure 3-video 1) provided by ***Pereira et al.*** (***2019***), and to demonstrate the versatility of our methods we also compared model performance across two previously unpublished posture data sets from groups of desert locusts (*Schistocerca gregaria*) filmed in a laboratory setting (Figure 3-video 2), and herds of Grévy's zebras (*Equus grevyi*) filmed in the wild (Figure 3-video 3). Our locust dataset was filmed from above using a high-resolution camera (Basler ace acA2040-90umNIR) and video recording system (Motif, loopbio GmbH), and our zebra dataset was filmed from above using a commercially-available quadcopter drone (DJI Phantom 4 Pro). Individuals in the videos were positionally tracked and the videos were then cropped using the egocentric coordinates of each individual and saved as separate videos—one for each individual. Further details of how these image datasets were acquired, preprocessed, and tracked before applying our pose estimation methods will be described elsewhere. The locust and zebra datasets are particularly challenging as they feature multiple interacting individuals—with focal individuals centered in the frame—and the latter with highly-variable light conditions. Before training each model we split each data set into randomly selected training and validation sets with 90% training examples and 10% validation examples. The details for each dataset are described in Table 1.

Model training

For each experiment, we set our model hyperparameters to the same configuration and all models were trained with $\frac{1}{4}\times$ resolution outputs and a stack of two hourglasses with two outputs where loss was applied (see Figure 2). Although our model hyperparameters could be infinitely adjusted to trade off between speed and accuracy, we compared only one configuration for each of our model implementations. These results are not meant to be an exhaustive search of model configurations

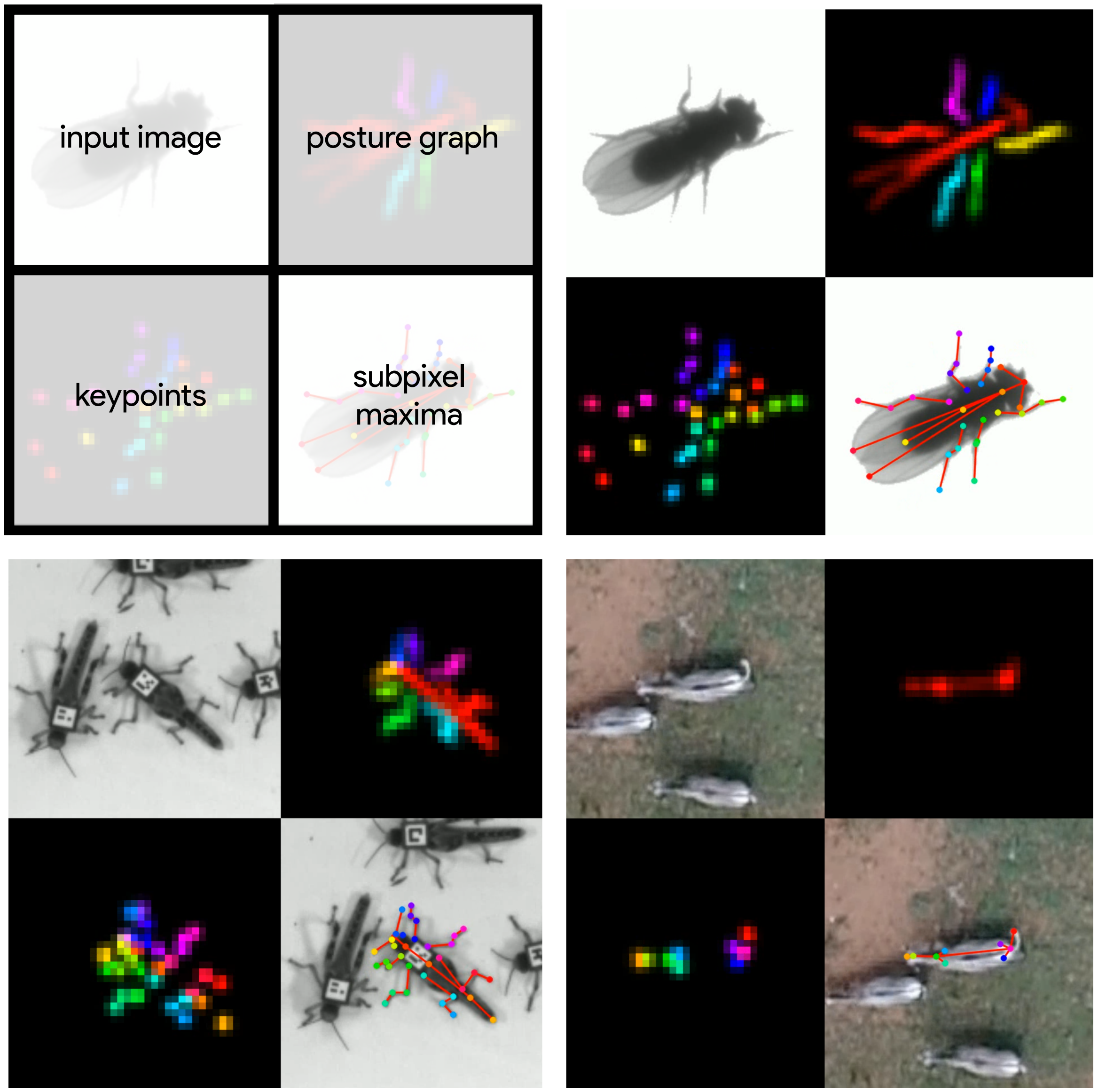


Figure 3. A visualization of the datasets we used to evaluate our methods (Table 1). For each dataset, confidence maps for the keypoints (bottom-left) and posture graph (top-right) are illustrated using different colors for each map. These outputs are from our Stacked DenseNet model at $\frac{1}{4}\times$ resolution.

Figure 3–video 1. A video of a behaving fly from ***Pereira et al.*** (***2019***) with pose estimation outputs visualized.

Figure 3–video 2. A video of a behaving locust with pose estimation outputs visualized.

Figure 3–video 3. A video of a behaving Grévy's zebra with pose estimation outputs visualized.

Table 1. Datasets used for model comparisons.

Name	Species	Resolution	# Images	# Keypoints	Individuals	Source
Vinegar fly	*Drosophila melanogaster*	192×192	1500	32	Single	***Pereira et al.*** (***2019***)
Desert locust	*Schistocerca gregaria*	160×160	800	35	Multiple	This paper
Grévy's zebra	*Equus grevyi*	160×160	1000	9	Multiple	This paper

as the best configuration will depend on the application. The details of the hyperparameters we used for each model are described in Appendix 8.

To make our posture estimation tasks closer to realistic conditions and properly demonstrate the robustness of our methods to rotation, translation, and scale, we applied various augmentations to each data set during training. All models were trained using data augmentations that included random flipping, or mirroring, along both image axes with 0.5 probability, random rotations around the center of the image in the range [-180°, +180°), random scaling between [90%, 110%] for flies and locusts, random scaling between [75%, 125%] for zebras to account for greater size variation in the data set, and random translations in the range [-5%, +5%]. After performing these spatial augmentations we also applied a variety of noise augmentations that included multiple types of additive noise, dropout, blurring, and contrast augmentations to further ensure robustness and generalization.

We trained our models (Figure 2) using mean squared error loss optimized using the ADAM optimizer (***Kingma and Ba, 2014***) with a learning rate of 1×10^{-3} and a batch size of 16. We lowered the learning rate by a factor of 5 each time the validation loss did not improve by more than 1×10^{-3} for 10 epochs. We considered models to be converged when the validation loss stopped improving for 50 epochs, and we calculated validation error as the Euclidean distance between predicted and ground-truth image coordinates for only the best performing version of the model, which we evaluated at the end of each epoch during optimization. We performed this procedure five times for each experiment and randomly selected a new validation set for each replicate.

Model evaluation

Machine learning models are typically evaluated for their ability to generalize to new data, known as *predictive performance*, using a held-out *test set*—a subsample of annotated data that is not used for training or validation. However, when fitting and evaluating a model on a small dataset, using an adequately-sized validation and test set can lead to erroneous conclusions about the predictive performance of the model if the training set is too small (***Kuhn and Johnson, 2013***). Therefore, to maximize the size of the training set, we elected to use only a validation set for model evaluation.

Generally a test set is used to avoid biased performance measures caused by overfitting the model hyperparameters to the validation set. However, we did not adjust our model architecture to achieve better performance on our validation set—only to achieve fast inference speeds. While we did use validation error to decide when to lower the learning rate during training and when to stop training, lowering the learning rate in this way should have no effect on the generalization ability of the model, and because we heavily augment our training set during optimization—forcing the model to learn a much larger image distribution than what is included in the training and validation sets—overfitting to the validation set is unlikely. We also demonstrate the generality of our results for each experiment by randomly selecting a new validation set with each replicate. All of these factors make the Euclidean error for the unaugmented validation set a reasonable measure of the predictive performance for each model.

The inference speed for each model was assessed by running predictions on 100,000 randomly generated images with a batch size of 1 for real-time speeds and a batch size of 100 for offline speeds. Our hardware consisted of a Dell Precision Tower 7910 workstation (Dell, Inc.) running Ubuntu Linux v18.04 with 2× Intel Xeon E5-2623 v3 CPUs (8 cores, 16 threads at 3.00GHz), 64GB of RAM, and a Titan Xp GPU (NVIDIA Corporation). While the hardware we used for development and testing is quite advanced, there is no requirement for this level of performance, and our software can easily be run on lower-end hardware. We evaluated inference speeds on multiple consumer-grade desktop computers and found similar performance (±10%) when using the same GPU.

Assessing prediction accuracy with Bayesian inference

To more rigorously assess performance differences between models, we parameterized the Euclidean error distribution for each experiment by fitting a Bayesian linear model with a Gamma-distributed likelihood function. This model takes the form:

$$\begin{aligned} p(y|X, \theta_\mu, \theta_\phi) &\sim Gamma(\alpha, \beta) \\ \alpha &= \mu^2 \phi^{-1} \\ \beta &= \mu \phi^{-1} \\ \mu &= h(X\theta_\mu) \\ \phi &= h(X\theta_\phi) \end{aligned}$$

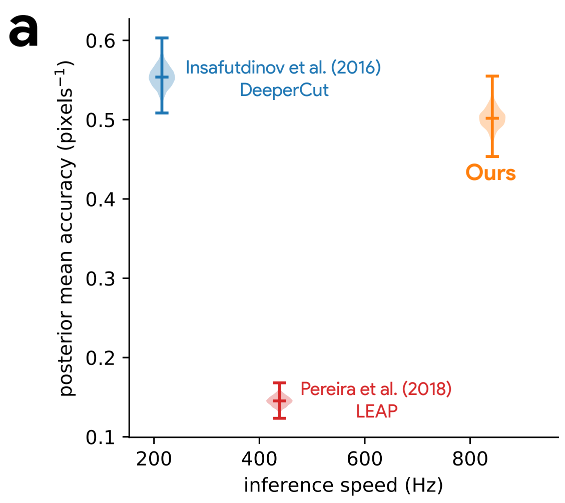
where X is the design matrix composed of binary indicator variables for each pose estimation model, θ_μ and θ_ϕ are vectors of intercepts, $h(\cdot)$ is the softplus function (***Dugas et al., 2001***)—or $h(x) = \log(1 + e^{\mathbf{x}})$—used to enforce positivity of μ and ϕ, and y is the Euclidean error of the pose estimation model. Parameterizing our error distributions in this way allows us to calculate the posterior distributions for the mean $\mathrm{E}[y] = \alpha\beta^{-1} \equiv \mu$ and variance $\mathrm{Var}[y] = \alpha\beta^{-2} \equiv \phi$. This parameterization then provides us with a statistically rigorous way to assess differences in model performance in terms of both central tendency and spread—accounting for both epistemic uncertainty (unknown unknowns, e.g. parameter uncertainty) and aleatoric uncertainty (known unknowns, e.g. data variance). Details of how we fitted these models can be found in Appendix 7.

Subpixel prediction allows for fast and accurate inference

We compared the accuracy of our subpixel maxima layer to an integer-based maxima layer using the fly dataset. We found significant accuracy improvements across every downsampling configuration (Appendix Figure 5). Even with confidence maps at $\frac{1}{8}\times$ the resolution of the original image, error did not drastically increase compared to full-resolution predictions. Making predictions at such a downsampled resolution allows us to achieve very fast inference >1000 Hz while maintaining relatively high accuracy. Additionally, achieving fast pose estimation using CNNs typically relies on massively parallel processing on the GPU with large batches of data, which makes fast real-time inference challenging to accomplish. Our Stacked DenseNet model, with a batch size of one, can run inference at ~30-110Hz—depending on resolution (Appendix Figure 5a)—which could be further improved by reconfiguring the model with fewer parameters. This opens the door to *truly* real-time behavioral experiments with prediction errors similar to current state-of-the-art methods.

Predicting global geometry improves accuracy and reduces extreme errors

We find that forcing the pose estimation model to predict a hierarchical posture graph reduces prediction error (Appendix Figure 6), and because the feature maps for the posture graph can be removed from the final output during inference, this effectively improves prediction accuracy for free. Both the mean and variance of the error distributions were lower when predicting the posture graph, which suggests that learning global geometry both decreases error *on average* and helps to reduce *extreme* prediction errors. The overall effect size for this decrease in error is fairly small (<1 pixel average reduction in error), but based on the results from the zebra dataset, this modification more dramatically improves performance for datasets with higher-variance images and sparse posture graphs. These results also suggest that annotating multiple keypoints to incorporate an explicit signal for global information may help improve prediction accuracy for a specific body part of interest.



b

```
# initialize, train, and save a model
from deepposekit.io import TrainingGenerator
from deepposekit.models import StackedDenseNet

generator = TrainingGenerator('path/to/data.h5')
model = StackedDenseNet(generator)
model.compile(optimizer='adam', loss='mse')
model.fit(batch_size=16, n_workers=8)
model.save('path/to/model.h5')


# load the model and run inference on new data
from deepposekit.models import load_model

model = load_model('path/to/model.h5')
new_data = load_new_data('path/to/new/data')
predictions = model.predict(new_data)
```

Figure 4. Our methods estimate posture at 2×—or greater—the speed of ***Pereira et al.*** (***2019***) while achieving similar accuracy to ***Insafutdinov et al.*** (***2016***)—the model used by ***Mathis et al.*** (***2018***)—shown here as mean accuracy (inverse Euclidean error) for our most challenging dataset of multiple interacting Grévy's zebras (*E. grevyi*) filmed in the wild (**a**). Our software interface is designed to be straightforward but flexible. We include many options for expert users to customize model training with sensible default settings to make pose estimation as easy as possible for beginners. For example, training a model and running inference on new data requires writing only a few lines of code and specifying some basic settings (**b**).

Our models are fast and robust

Finally, we benchmarked our model implementations against the models from ***Pereira et al.*** (***2019***) and ***Insafutdinov et al.*** (***2016***). We find that our Stacked DenseNet model outperforms both LEAP (***Pereira et al., 2019***) and DeeperCut (***Insafutdinov et al., 2016***) in terms of speed while also achieving much higher accuracy than LEAP (***Pereira et al., 2019***) with similar accuracy to DeeperCut (***Insafutdinov et al., 2016***) (Figure 4a). We found that both the Stacked Hourglass and Stacked DenseNet models outperformed LEAP (***Pereira et al., 2019***) with >2× faster inference speeds and >3× higher mean accuracy. Not only were our models' average prediction error significantly improved, but also, importantly, the variance was lower—indicating that our models produced fewer extreme prediction errors. At $\frac{1}{4}\times$ resolution, our Stacked DenseNet implementation consistently achieved prediction accuracy comparable to that of ***Insafutdinov et al.*** (***2016***)—with <0.5-pixel increase in average error—while running inference at nearly 4× the speed and using only ~2% of the parameters— ~26 million vs. ~0.5 million. The inference speed could be further improved by using a $\frac{1}{8}\times$ output without much increase in error (Appendix Figure 5) or by further adjusting the hyperparameters to constrain the size of the model. Our Stacked Hourglass implementation followed closely behind this level of performance but consistently performed worse than our Stacked DenseNet model. We were also able to reproduce the results reported by ***Pereira et al.*** (***2019***) that LEAP and the Stacked Hourglass model from ***Newell et al.*** (***2016***) have similar average prediction error for the fly dataset. However, we also find that LEAP (***Pereira et al., 2019***) has much higher variance, which suggests it is more prone to extreme prediction errors—a problem for further data analysis. Detailed results of our model comparisons are shown in Appendix Figure 7.

Discussion

Here we have presented a new framework for estimating animal posture using deep learning models. We built on the state-of-the-art for individual pose estimation using convolutional neural networks to achieve fast inference without significantly reducing accuracy. Our pose estimation methods outperform currently-available methods from ***Mathis et al.*** (***2018***) (DeeperCut; ***Insafutdinov et al. 2016***) and ***Pereira et al.*** (***2019***) (LEAP) while also providing a simplified interface (Figure 4b) for using these advanced tools to measure animal behavior and locomotion. We tested our methods across a range of datasets from controlled laboratory environments with single individuals to challenging field situations with multiple interacting individuals and variable lighting conditions. We found that our methods perform well for all of these situations. We ran experiments to optimize our approach and discovered that some straightforward modifications can greatly improve speed and performance. Additionally, we demonstrated that these modifications improve not the just the average error but also help to reduce extreme prediction errors—a key determinant for the reliability of subsequent statistical analysis.

We highlighted important considerations when using CNNs for pose estimation and reviewed the progress of fully-convolutional regression models from the literature. Recent advancements for these models have been driven mostly by a strategy of adding more connections between layers to increase performance and efficiency (e.g. ***Jégou et al. 2017***). New fundamentally-different models (***Sabour et al., 2017***) and loss functions (***Chen et al., 2017***) may provide further performance improvements. Recent work (e.g. ***Weigert et al. 2018***; ***Roy et al. 2018***) has also shown that future progress may require more mathematically-principled approaches such as applying probabilistic concepts (***Kendall and Gal, 2017***) and Bayesian inference at scale (***Tran et al., 2018***).

Measuring behavior is an critical factor for many studies in neuroscience (***Krakauer et al., 2017***). Understanding the connections between brain activity and behavioral output requires detailed and objective descriptions of body posture that match the richness and resolution neural measurement technologies have provided for years (***Anderson and Perona, 2014***; ***Berman, 2018***; ***Brown and De Bivort, 2018***), which our methods and other deep-learning–based tools provide (***Mathis et al., 2018***; ***Pereira et al., 2019***). We have also demonstrated the possibility that our toolkit could be used for truly real-time inference, which allows for closed-loop experiments where sensory stimuli or optogenetic stimulation are controlled in response to behavioral measurements (e.g. ***Bath et al. 2014***; ***Stowers et al. 2017***). Using real-time measurements in conjunction with optogenetics or thermogenetics may be key to disentangling the causal structure of motor output from the brain—especially given that recent work has shown an animal's response to optogenetic stimulation can differ depending on the behavior it is currently performing (***Cande et al., 2018***). Real-time behavioral quantification is also particularly important as closed-loop virtual reality is quickly becoming an indispensable tool for studying sensorimotor relationships in individuals and collectives (***Stowers et al., 2017***).

Quantifying individual movement is essential for revealing the genetic (***Kain et al., 2012***; ***Ayroles et al., 2015***) and environmental (***Bierbach et al., 2017***; ***Akhund-Zade et al., 2019***) underpinnings of phenotypic variation in behavior—as well as the phylogeny of behavior (e.g. ***Berman et al. 2014b***). Measuring individual behavioral phenotypes requires tools that are robust, scaleable, and easy-to-use, and our approach offers the ability to quickly and accurately quantify the behavior of many individuals in great detail. When combined with tools for genetic manipulations (***Ran et al., 2013***; ***Doudna and Charpentier, 2014***), high-throughput behavioral experiments (***Alisch et al., 2018***; ***Werkhoven et al., 2019***), and behavioral analysis (e.g. ***Berman et al. 2014a***; ***Pereira et al. 2019***; ***Wiltschko et al. 2015***), our methods could help to provide the data resolution and statistical power needed for dissecting the complex relationships between genes, environment, and behavioral variation.

When used together with other tools for localization and tracking, our methods are capable of reliably measuring posture for multiple interacting individuals. The importance of measuring detailed representations of individual behavior when studying animal collectives has been well established (***Strandburg-Peshkin et al., 2013***; ***Rosenthal et al., 2015***; ***Strandburg-Peshkin et al., 2015***, ***2017***). Estimating body posture is an essential first step for unraveling the sensory networks that drive group coordination, such as vision-based networks measured via raycasting (***Strandburg-Peshkin et al., 2013***; ***Rosenthal et al., 2015***). Additionally, using body pose estimation in combination with computational models of behavior (e.g. ***Costa et al. 2019***, ***Wiltschko et al. 2015***) and unsupervised behavioral classification methods (e.g. ***Berman et al. 2014a***, ***Pereira et al. 2019***) may allow for further dissection of how information flows through collectives by revealing the networks of behavioral contagion across multiple timescales and sensory modalities.

When combined with unmanned aerial vehicles (UAVs; ***Schiffman 2014***) or other field-based imaging (***Francisco et al., 2019***), applying these methods to the study of individuals and groups in the wild can provide high-resolution behavioral data that goes beyond the capabilities of current GPS and accelerometry-based technologies (***Nagy et al., 2010***, ***2013***; ***Kays et al., 2015***; ***Strandburg-Peshkin et al., 2015***, ***2017***; ***Flack et al., 2018***)—especially for species that cannot be studied with tags or collars. Additionally, by applying these methods in conjunction with 3-D habitat reconstruction—using techniques such as photogrammetry—field-based studies can begin to integrate fine-scale behavioral measurements with the full 3-D environment in which the behavior evolved (e.g. ***Strandburg-Peshkin et al. 2017***; ***Francisco et al. 2019***). This combination of technologies could allow researchers to address questions about the behavioral ecology of animals that were previously impossible to answer.

In conclusion, we have presented a toolkit, called DeepPoseKit, for automatically measuring animal posture from images. Our methods are fast, robust, and widely applicable to a range of species and experimental conditions. When designing our framework we emphasized usability across our entire software interface, which we expect will help to make these advanced tools accessible to a wider range of researchers. The fast inference and real-time capabilities of our methods should also help further reduce barriers to previously intractable questions across many scientific disciplines—including neuroscience, ethology, and behavioral ecology—both in the laboratory and the field.

Author contributions

J.M.G and I.D.C conceived the idea for the project. J.M.G. and D.C. developed the software with input from H.N. J.M.G implemented the pose estimation models and developed the subpixel maxima algorithm. J.M.G. and D.C. developed the annotation GUI, data augmentation pipeline, and wrote the documentation. J.M.G., D.C. and H.N. designed the experiments. J.M.G. and D.C. ran the experiments. B.R.C., B.K., J.M.G., and I.D.C. conceived the idea to apply posture tracking to zebras. B.R.C. and B.K. provided the annotated zebra posture data. B.K., and L.L. helped with initial testing and improvement of the software interface. L.L. also made significant contributions to an earlier version of the manuscript. J.M.G. fit the linear models and made the figures. J.M.G. wrote the initial draft of the manuscript with input from H.N. and D.C., and all authors helped revise the manuscript.

Acknowledgements

We are indebted to Talmo Pereira et al. and Mathis et al. for making their software open-source and freely-available—this project would not have been possible without them. We also thank François Chollet, the Keras team, and Alexander Jung for their open source contributions, which provided the core programming interface for our work. We thank Vivek H. Sridhar, Michael L. Smith, and Joseph B. Bak-Coleman for their helpful discussions on the project. We also thank M.L.S. for the use of his GPU. We thank Felicitas Oehler for annotating the zebra posture data and Chiara Hirschkorn for assistance with filming the locusts and annotating the locust posture data. We thank Alex Bruttel, Christine Bauer, Jayme Weglarski, Dominique Leo, and loobio GmbH for providing technical support. We acknowledge the NVIDIA Corporation for their generous donations to our research. This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 748549. B.R.C. acknowledges support from the University of Konstanz Zukunftskolleg's Investment Grant program. I.D.C. acknowledges support from NSF Grant IOS-1355061, Office of Naval Research Grants N00014-09-1-1074 and N00014-14-1-0635, Army Research Office Grants W911NG-11-1-0385 and W911NF14-1-0431, the Struktur-und Innovationsfonds fur die Forschung of the State of Baden-Württemberg, the DFG Centre of Excellence 2117 "Centre for the Advanced Study of Collective Behaviour" (ID: 422037984), and the Max Planck Society.

Animal Ethics Statement

All procedures for collecting the zebra (*E. grevyi*) dataset were reviewed and approved by Ethikrat, the independent Ethics Council of the Max Planck Society. The zebra dataset was collected with the permission of Kenya's National Commission for Science, Technology and Innovation (NACOSTI/P/17/59088/15489 and NACOSTI/P/18/59088/21567) using drones operated by B.R.C. with the permission of the Kenya Civil Aviation Authority (authorization numbers: KCAA/OPS/2117/4 Vol. 2 (80), KCAA/OPS/2117/4 Vol. 2 (81), KCAA/OPS/2117/5 (86) and KCAA/OPS/2117/5 (87); RPAS Operator Certificate numbers: RPA/TP/0005 AND RPA/TP/000-0009).

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. https://www.tensorflow.org/, software available from tensorflow.org.

Akhund-Zade J, Ho S, O'Leary C, de Bivort BL. The effect of environmental enrichment on behavioral variability depends on genotype, behavior, and type of enrichment. bioRxiv. 2019; p. 557181.

Alisch T, Crall JD, Kao AB, Zucker D, de Bivort BL. MAPLE (modular automated platform for large-scale experiments), a robot for integrated organism-handling and phenotyping. eLife. 2018; 7:e37166.

Anderson DJ, Perona P. Toward a science of computational ethology. Neuron. 2014; 84(1):18–31.

Andriluka M, Iqbal U, Insafutdinov E, Pishchulin L, Milan A, Gall J, Schiele B. Posetrack: A benchmark for human pose estimation and tracking. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2018. p. 5167–5176.

Andriluka M, Pishchulin L, Gehler P, Schiele B. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2014. .

Ayinde BO, Zurada JM. Building efficient convnets using redundant feature pruning. arXiv preprint arXiv:180207653. 2018; .

Ayroles JF, Buchanan SM, O'Leary C, Skutt-Kakaria K, Grenier JK, Clark AG, Hartl DL, de Bivort BL. Behavioral idiosyncrasy reveals genetic control of phenotypic variability. Proceedings of the National Academy of Sciences. 2015; 112(21):6706–6711.

Badrinarayanan V, Kendall A, Cipolla R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. CoRR. 2015; abs/1511.00561. http://arxiv.org/abs/1511.00561.

Bath DE, Stowers JR, Hörmann D, Poehlmann A, Dickson BJ, Straw AD. FlyMAD: rapid thermogenetic control of neuronal activity in freely walking Drosophila. Nature methods. 2014; 11(7):756.

Berman GJ. Measuring behavior across scales. BMC biology. 2018; 16(1):23.

Berman GJ, Bialek W, Shaevitz JW. Predictability and hierarchy in Drosophila behavior. Proceedings of the National Academy of Sciences. 2016; 113(42):11943–11948.

Berman GJ, Choi DM, Bialek W, Shaevitz JW. Mapping the stereotyped behaviour of freely moving fruit flies. Journal of The Royal Society Interface. 2014; 11(99):20140672.

Berman GJ, Choi DM, Bialek W, Shaevitz JW. Mapping the structure of drosophilid behavior. bioRxiv. 2014; p. 002873.

Bierbach D, Laskowski KL, Wolf M. Behavioural individuality in clonal fish arises despite near-identical rearing conditions. Nature communications. 2017; 8:15361.

Boenisch F, Rosemann B, Wild B, Dormagen D, Wario F, Landgraf T. Tracking All Members of a Honey Bee Colony Over Their Lifetime Using Learned Models of Correspondence. Frontiers in Robotics and AI. 2018; 5:35. https://www.frontiersin.org/article/10.3389/frobt.2018.00035, doi: 10.3389/frobt.2018.00035.

Brown AE, De Bivort B. Ethology as a physical science. Nature Physics. 2018; p. 1.

Cande J, Namiki S, Qiu J, Korff W, Card GM, Shaevitz JW, Stern DL, Berman GJ. Optogenetic dissection of descending behavioral control in Drosophila. Elife. 2018; 7:e34275.

Cao Z, Simon T, Wei SE, Sheikh Y. Realtime multi-person 2d pose estimation using part affinity fields. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2017. p. 7291–7299.

Carpenter B, Lee D, Brubaker MA, Riddell A, Gelman A, Goodrich B, Guo J, Hoffman M, Betancourt M, Li P. Stan: A Probabilistic Programming Language. J Stat Softw. 2017; .

Cauchy A. Méthode générale pour la résolution des systemes d'équations simultanées. Comp Rend Sci Paris. 1847; 25(1847):536–538.

Chen Y, Shen C, Wei XS, Liu L, Yang J. Adversarial posenet: A structure-aware convolutional network for human pose estimation. In: *Proceedings of the IEEE International Conference on Computer Vision*; 2017. p. 1212–1221.

Chollet F, et al., Keras. GitHub; 2015. https://github.com/fchollet/keras.

Chollet F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2017. p. 1251–1258.

Costa AC, Ahamed T, Stephens GJ. Adaptive, locally linear models of complex dynamics. Proceedings of the National Academy of Sciences. 2019; 116(5):1501–1510. https://www.pnas.org/content/116/5/1501, doi: 10.1073/pnas.1813476116.

Crall JD, Gravish N, Mountcastle AM, Combes SA. BEEtag: a low-cost, image-based tracking system for the study of animal behavior and locomotion. PloS one. 2015; 10(9):e0136487.

Dell AI, Bender JA, Branson K, Couzin ID, de Polavieja GG, Noldus LP, Pérez-Escudero A, Perona P, Straw AD, Wikelski M, et al. Automated image-based tracking and its application in ecology. Trends in ecology & evolution. 2014; 29(7):417–428.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. . 2009; .

Doudna JA, Charpentier E. The new frontier of genome engineering with CRISPR-Cas9. Science. 2014; 346(6213):1258096.

Duane S, Kennedy AD, Pendleton BJ, Roweth D. Hybrid Monte Carlo. Phys Lett B. 1987; 195(2):216–222.

Dugas C, Bengio Y, Bélisle F, Nadeau C, Garcia R. Incorporating second-order functional knowledge for better option pricing. In: *Advances in neural information processing systems*; 2001. p. 472–478.

Flack A, Nagy M, Fiedler W, Couzin ID, Wikelski M. From local collective behavior to global migratory patterns in white storks. Science. 2018; 360(6391):911–914.

Francisco FA, Nührenberg P, Jordan AL. A low-cost, open-source framework for tracking and behavioural analysis of animals in aquatic ecosystems. bioRxiv. 2019; https://www.biorxiv.org/content/early/2019/03/09/571232, doi: 10.1101/571232.

Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016.

Graving JM, pinpoint: behavioral tracking using 2D barcode tags v0.0.1-alpha; 2017. https://doi.org/10.5281/zenodo.1008970, doi: 10.5281/zenodo.1008970.

Guizar-Sicairos M, Thurman ST, Fienup JR. Efficient subpixel image registration algorithms. Optics letters. 2008; 33(2):156–158.

He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2016. p. 770–778.

Hoffman MD, Gelman A. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. Journal of Machine Learning Research. 2014; 15(1):1593–1623.

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2017. p. 4700–4708.

Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2017. p. 7310–7311.

Insafutdinov E, Pishchulin L, Andres B, Andriluka M, Schiele B. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In: *European Conference on Computer Vision* Springer; 2016. p. 34–50.

Iqbal U, Milan A, Gall J. Posetrack: Joint multi-person pose estimation and tracking. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2017. p. 2011–2020.

Jégou S, Drozdzal M, Vázquez D, Romero A, Bengio Y. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. CoRR. 2017; abs/1611.09326. http://arxiv.org/abs/1611.09326.

Johnson M, Duvenaud DK, Wiltschko A, Adams RP, Datta SR. Composing graphical models with neural networks for structured representations and fast inference. In: *Advances in neural information processing systems*; 2016. p. 2946–2954.

Jolles JW, Boogert NJ, Sridhar VH, Couzin ID, Manica A. Consistent individual differences drive collective behavior and group functioning of schooling fish. Current Biology. 2017; 27(18):2862–2868.

Jung A, imgaug. GitHub; 2018. https://github.com/aleju/imgaug.

Kain J, Stokes C, Gaudry Q, Song X, Foley J, Wilson R, De Bivort B. Leg-tracking and automated behavioural classification in Drosophila. Nature communications. 2013; 4:1910.

Kain JS, Stokes C, de Bivort BL. Phototactic personality in fruit flies and its suppression by serotonin and white. Proceedings of the National Academy of Sciences. 2012; 109(48):19834–19839.

Kays R, Crofoot MC, Jetz W, Wikelski M. Terrestrial animal tracking as an eye on life and planet. Science. 2015; 348(6240):aaa2478.

Ke L, Chang MC, Qi H, Lyu S. Multi-Scale Structure-Aware Network for Human Pose Estimation. In: *The European Conference on Computer Vision (ECCV)*; 2018. .

Kendall A, Gal Y. What uncertainties do we need in bayesian deep learning for computer vision? In: *Advances in neural information processing systems*; 2017. p. 5574–5584.

Kiefer J, Wolfowitz J, et al. Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics. 1952; 23(3):462–466.

Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014; .

Klambauer G, Unterthiner T, Mayr A, Hochreiter S. Self-normalizing neural networks. In: *Advances in neural information processing systems*; 2017. p. 971–980.

Klibaite U, Berman GJ, Cande J, Stern DL, Shaevitz JW. An unsupervised method for quantifying the behavior of paired animals. Physical biology. 2017; 14(1):015006.

Klibaite U, Shaevitz JW. Interacting fruit flies synchronize behavior. bioRxiv. 2019; p. 545483.

Krakauer JW, Ghazanfar AA, Gomez-Marin A, MacIver MA, Poeppel D. Neuroscience needs behavior: correcting a reductionist bias. Neuron. 2017; 93(3):480–490.

Kuhn M, Johnson K. Applied predictive modeling, vol. 26. Springer; 2013.

LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015; 521(7553):436.

Li H, Xu Z, Taylor G, Studer C, Goldstein T. Visualizing the loss landscape of neural nets. In: *Advances in Neural Information Processing Systems*; 2018. p. 6391–6401.

Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2015. p. 3431–3440.

Markowitz JE, Gillis WF, Beron CC, Neufeld SQ, Robertson K, Bhagat ND, Peterson RE, Peterson E, Hyun M, Linderman SW, et al. The striatum organizes 3D behavior via moment-to-moment action selection. Cell. 2018; 174(1):44–58.

Mathis A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nature Neuroscience. 2018; https://www.nature.com/articles/s41593-018-0209-y.

Mathis A, Warren RA. On the inference speed and video-compression robustness of DeepLabCut. bioRxiv. 2018; https://www.biorxiv.org/content/early/2018/10/30/457242, doi: 10.1101/457242.

Mendes CS, Bartos I, Akay T, Márka S, Mann RS. Quantification of gait parameters in freely walking wild type and sensory deprived Drosophila melanogaster. elife. 2013; 2:e00231.

Nagy M, Akos Z, Biro D, Vicsek T. Hierarchical group dynamics in pigeon flocks. Nature. 2010; 464(7290):890.

Nagy M, Vásárhelyi G, Pettit B, Roberts-Mariani I, Vicsek T, Biro D. Context-dependent hierarchies in pigeons. Proceedings of the National Academy of Sciences. 2013; 110(32):13049–13054.

Nath T, Mathis A, Chen AC, Patel A, Bethge M, Mathis MW. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. bioRxiv. 2018; https://www.biorxiv.org/content/early/2018/11/24/476531, doi: 10.1101/476531.

Newell A, Yang K, Deng J. Stacked Hourglass Networks for Human Pose Estimation. CoRR. 2016; abs/1603.06937. http://arxiv.org/abs/1603.06937.

Pereira TD, Aldarondo DE, Willmore L, Kislin M, Wang SSH, Murthy M, Shaevitz JW. Fast animal pose estimation using deep neural networks. Nature methods. 2019; 16(1):117.

Pérez-Escudero A, Vicente-Page J, Hinz RC, Arganda S, De Polavieja GG. idTracker: tracking individuals in a group by automatic identification of unmarked animals. Nature methods. 2014; 11(7):743.

Pratt LY. Discriminability-based transfer between neural networks. In: *Advances in neural information processing systems*; 1993. p. 204–211.

Prechelt L. Automatic early stopping using cross validation: quantifying the criteria. Neural Networks. 1998; 11(4):761–767.

Ran FA, Hsu PD, Wright J, Agarwala V, Scott DA, Zhang F. Genome engineering using the CRISPR-Cas9 system. Nature protocols. 2013; 8(11):2281.

Robbins H, Monro S. A stochastic approximation method. The annals of mathematical statistics. 1951; p. 400–407.

Romero-Ferrero F, Bergomi MG, Hinz R, Heras FJ, de Polavieja GG. idtracker. ai: Tracking all individuals in large collectives of unmarked animals. arXiv preprint arXiv:180304351. 2018; .

Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention* Springer; 2015. p. 234–241.

Rosenthal SB, Twomey CR, Hartnett AT, Wu HS, Couzin ID. Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion. Proceedings of the National Academy of Sciences. 2015; 112(15):4690–4695.

Roy AG, Conjeti S, Navab N, Wachinger C. Bayesian QuickNAT: Model Uncertainty in Deep Whole-Brain Segmentation for Structure-wise Quality Control. CoRR. 2018; abs/1811.09800. http://arxiv.org/abs/1811.09800.

Sabour S, Frosst N, Hinton GE. Dynamic routing between capsules. In: *Advances in neural information processing systems*; 2017. p. 3856–3866.

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2018. p. 4510–4520.

Schiffman R, Drones flying high as new tool for field biologists. American Association for the Advancement of Science; 2014.

Stowers JR, Hofbauer M, Bastien R, Griessner J, Higgins P, Farooqui S, Fischer RM, Nowikovsky K, Haubensak W, Couzin ID, et al. Virtual reality for freely moving animals. Nature methods. 2017; 14(10):995.

Strandburg-Peshkin A, Farine DR, Couzin ID, Crofoot MC. Shared decision-making drives collective movement in wild baboons. Science. 2015; 348(6241):1358–1361.

Strandburg-Peshkin A, Farine DR, Crofoot MC, Couzin ID. Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. Elife. 2017; 6:e19505.

Strandburg-Peshkin A, Twomey CR, Bode NW, Kao AB, Katz Y, Ioannou CC, Rosenthal SB, Torney CJ, Wu HS, Levin SA, et al. Visual sensory networks and effective information transfer in animal groups. Current Biology. 2013; 23(17):R709–R711.

Todd JG, Kain JS, de Bivort BL. Systematic exploration of unsupervised methods for mapping behavior. Physical biology. 2017; 14(1):015002.

Tran D, Hoffman MW, Moore D, Suter C, Vasudevan S, Radul A. Simple, distributed, and accelerated probabilistic programming. In: *Advances in Neural Information Processing Systems*; 2018. p. 7609–7620.

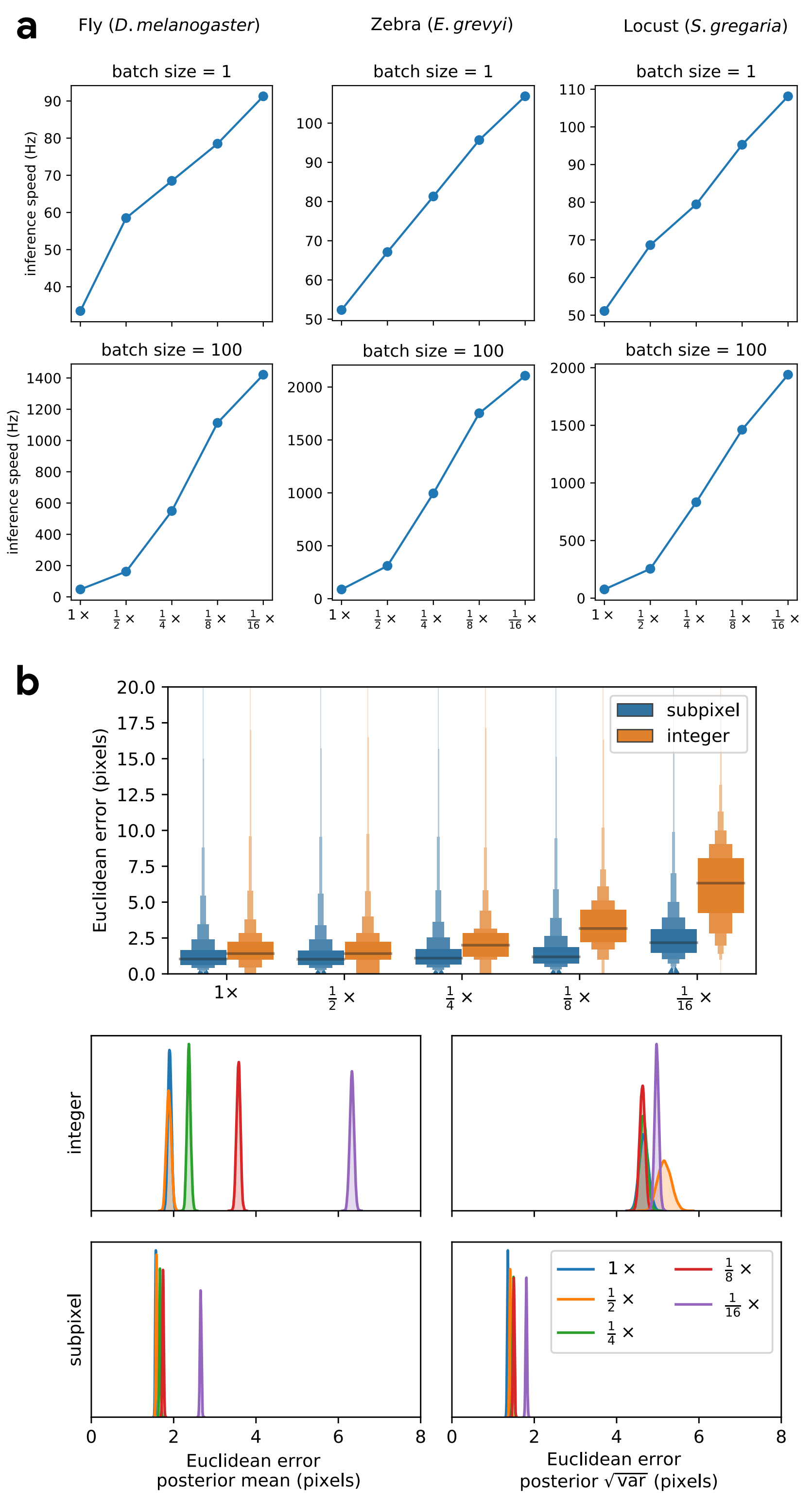
Uhlmann V, Ramdya P, Delgado-Gonzalo R, Benton R, Unser M. FlyLimbTracker: An active contour based approach for leg segment tracking in unmarked, freely behaving Drosophila. PLoS One. 2017; 12(4):e0173433.

Weigert M, Schmidt U, Boothe T, Müller A, Dibrov A, Jain A, Wilhelm B, Schmidt D, Broaddus C, Culley S, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. Nature methods. 2018; 15(12):1090.

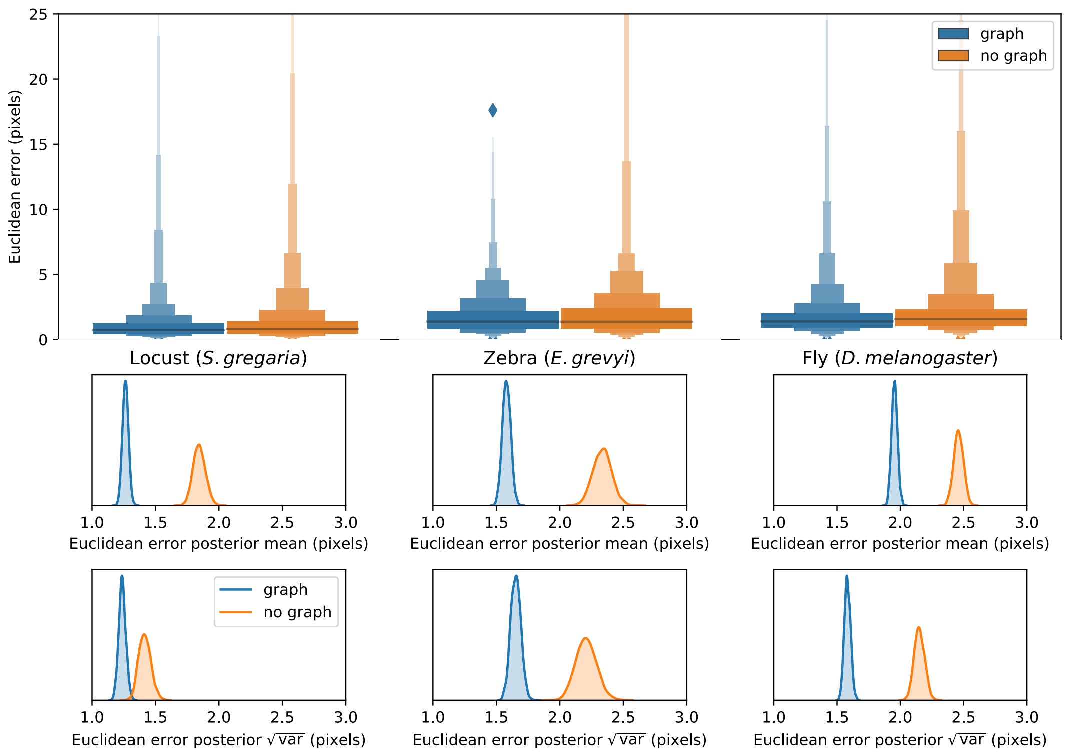
Werkhoven Z, Rohrsen C, Qin C, Brembs B, de Bivort B. MARGO (Massively Automated Real-time GUI for Object-tracking), a platform for high-throughput ethology. BioRxiv. 2019; p. 593046.

Wild B, Sixt L, Landgraf T. Automatic localization and decoding of honeybee markers using deep convolutional neural networks. CoRR. 2018; abs/1802.04557. http://arxiv.org/abs/1802.04557.

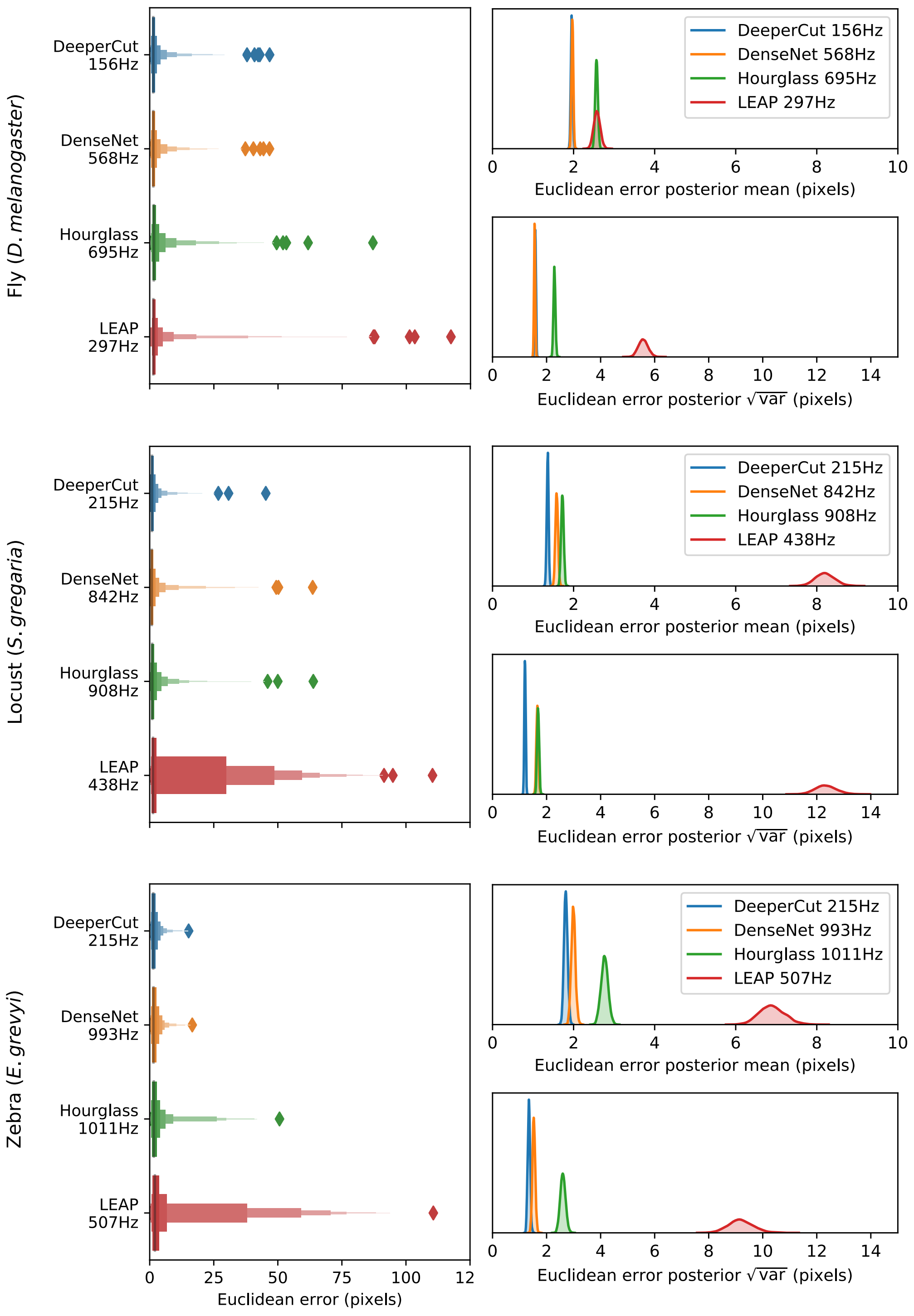
Wiltschko AB, Johnson MJ, Iurilli G, Peterson RE, Katon JM, Pashkovski SL, Abraira VE, Adams RP, Datta SR. Mapping sub-second structure in mouse behavior. Neuron. 2015; 88(6):1121–1135.



Appendix 0 Figure 5. Our subpixel maxima algorithm increases speed (**a**) without decreasing accuracy (**b**). Inference speed is fast and can be run in real-time on single images (batch size = 1) at ~30-110Hz. Plots show the inference speeds for our Stacked DenseNet model across downsampling configurations for each of our datasets (**a**). Prediction accuracy on the fly dataset is maintained across downsampling configurations (**b**). Letter-value plots (**top**) show the raw error distributions for each configuration. Visualizations of the posterior distributions for the mean and variance (**bottom**) illustrate statistical differences between the error distributions, where using subpixel maxima decreases both the mean and variance of the error distribution.



Appendix 0 Figure 6. Predicting the global geometry of the posture graph reduces error. Letter-value plots (**top**) show the raw error distributions for each experiment. Visualizations of the posterior distributions for the mean and variance (**bottom**) show statistical differences between the error distributions. Predicting the posture graph decreases both the mean and variance of the error distribution.



Appendix 0 Figure 7. Euclidean error distributions for each model across our three datasets. Letter-value plots (**left**) show the raw error distributions for each model. Histograms of the posterior distributions for the mean and variance (**right**) show statistical differences between the error distributions. Overall the LEAP model from ***Pereira et al.*** (***2019***) was the worst performer on every dataset in terms of both mean and variance. The DeeperCut model from ***Insafutdinov et al.*** (***2016***) was the best performer on every dataset followed closely by our implementation of Stacked Densenet (***Jégou et al., 2017***) and Stacked Hourglass (***Newell et al., 2016***). The posteriors for ***Insafutdinov et al.*** (***2016***) and our Stacked DenseNet model overlap almost completely for the fly dataset.

Appendix 1

Convolutional neural networks (CNNs)

Artificial neural networks like CNNs are complex, non-linear regression models that "learn" a hierarchically–organized set of parameters from real-world data via optimization. These machine learning models are now commonplace in science and industry and have proven to be surprisingly effective for a large number of applications where more conventional statistical models have failed (***LeCun et al., 2015***). For computer vision tasks, CNN parameters typically take the form of two-dimensional convolutional filters that are optimized to detect spatial features needed to model relationships between high-dimensional image data and some related variable(s) of interest, such as locations in space—e.g. posture keypoints—or semantic labels (***Long et al., 2015***; ***Badrinarayanan et al., 2015***).

Once a training set is generated (Appendix 2), a CNN model must be selected and optimized to perform the prediction task. CNNs are incredibly flexible with regard to how models are specified and trained, which is both an advantage and a disadvantage. This flexibility means models can be adapted to almost any computer vision task, but it also means the number of possible model architectures and optimization schemes is very large. This can make selecting an architecture and specifying hyperparameters a challenging process. However, most research on pose estimation has converged on a set of models that generally work well for this task (Appendix 3).

After selecting an architecture, the parameters of the model are set to an initial value and then iteratively updated to minimize some objective function, or *loss function*, that describes the difference between the model's predictive distribution and the true distribution of the data—in other words, the likelihood of the model's output is maximized. These parameter updates are performed using a modified version of the gradient descent algorithm (***Cauchy 1847***) known as *mini-batch stochastic gradient descent*—often referred to as simply *stochastic gradient descent* or *SGD* (***Robbins and Monro, 1951***; ***Kiefer et al., 1952***). SGD iteratively optimizes the model parameters using small randomly-selected subsamples, or *batches*, of training data. Using SGD allows the model to be trained on extremely large datasets in an iterative "online" fashion without the need to load the entire dataset into memory. The model parameters are updated with each batch by adjusting the parameter values in a direction that minimizes the error—where one round of training on the full dataset is commonly referred to as an *epoch*. The original SGD algorithm requires careful selection and tuning of hyperparameters to successfully optimize a model, but modern versions of the algorithm, such as *ADAM* (***Kingma and Ba, 2014***), automatically tune these hyperparameters, which makes optimization more straightforward.

The model parameters are optimized until they reach a convergence criterion, which is some measure of performance that indicates the model has reached a good location in parameter space. The most commonly used convergence criterion is a measure of predictive accuracy—often the loss function used for optimization—on a held-out *validation set*—a subsample of the training data not used for optimization—that evaluates the model's ability to generalize to new "out-of-sample" data. The model is typically evaluated at the end of each training epoch to assess performance on the validation set. Once performance on the validation set stops improving, training is usually stopped to prevent the model from overfitting to the training set—a technique known as *early stopping* (***Prechelt, 1998***).

Appendix 2

Collecting training data

Depending on the variability of the data, CNNs usually require thousands or tens of thousands of manually-annotated examples in order to reach human-level accuracy. However, in laboratory settings, sources of image variation like lighting and spatial scale can be more easily controlled, which minimizes the number of training examples needed to achieve accurate predictions.

This need for a large training set can be further reduced in a number of ways. Two commonly used methods include (1) *transfer learning*—using a model with parameters that are pre-trained on a larger set of images, such as the ImageNet database (***Deng et al., 2009***), containing diverse features (***Pratt, 1993***; ***Insafutdinov et al., 2016***; ***Mathis et al., 2018***)—and (2) *augmentation*— artificially increasing data variance by applying spatial and noise transformations such as flipping (mirroring), rotating, scaling, and adding different forms of noise or artificial occlusions. Both of these methods act as useful forms of *regularization*—incorporating a prior distribution—that allows the model to generalize well to new data even when the training set is small. Transfer learning incorporates prior information that images from the full dataset should contain statistical features similar to other images of the natural world, while augmentation incorporates prior knowledge that animals are bilaterally symmetric, can vary in their body size, position, and orientation, and that noise and occlusions sometimes occur.

Pereira et al. (***2019***) introduced two especially clever solutions for collecting an adequate training set. First, they cluster unannotated images based on pixel variance and uniformly sample images from each cluster, which reduces correlation between training examples and ensures the training data are representative of the entire distribution of possible images. Second, they use *active learning* where a CNN is trained on a small number of annotated examples and is then used to initialize keypoint locations for a larger set of unannotated data. These pre-initialized data are then manually corrected by the annotator, the model is retrained, and the unannotated data are re-initialized. The annotator applies this process iteratively as the training set grows larger until they are providing only minor adjustments to the pre-initialized data. This “human-in-the-loop”-style annotation expedites the process of generating an adequately large training set by reducing the cognitive load on the annotator—where the pose estimation model serves as a “cognitive partner”. Such a strategy also allows the annotator to automatically select new training examples based on the performance of the current iteration—where low-confidence predictions indicate examples that should be annotated for maximum improvement (Figure 1).

Of course, annotating image data requires software made for this purpose. ***Pereira et al.*** (***2019***) provide a custom annotation GUI written in MATLAB specifically designed for annotating posture using an active learning strategy. ***Mathis et al.*** (***2018***) originally did not provide an annotation tool, but recently added a Python-based GUI in an updated version of their software—including active learning and image sampling methods (see ***Nath et al. 2018***). Our framework also includes a Python-based GUI for annotating data with similar features to ***Mathis et al.*** (***2018***) and ***Pereira et al.*** (***2019***).

Appendix 3

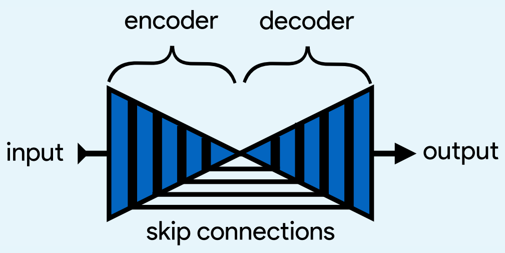
Fully-convolutional regression

For the task of pose estimation, a CNN is optimized to predict the locations of postural keypoints in an image. One approach is to use a CNN to directly predict the numerical value of each keypoint coordinate as an output. However, making predictions in this way removes real-world constraints on the model's predictive distribution by destroying spatial relationships within images, which negates many of the advantages of using CNNs in the first place.

CNNs are particularly good at transforming one image to produce another related image, or set of images, while preserving spatial relationships and allowing for translation-invariant predictions—a configuration known as a *fully-convolutional neural network* or *F-CNN* (**Long et al., 2015**). Therefore, instead of directly regressing images to coordinate values, a popular solution (**Newell et al., 2016**; **Insafutdinov et al., 2016**; **Mathis et al., 2018**; **Pereira et al., 2019**) is to optimize a F-CNN that transforms images to predict a stack of output images known as *confidence maps*—one for each keypoint. Each confidence map in the output volume contains a single, two-dimensional, symmetric Gaussian indicating the location of each joint, and the scalar value of the peak indicates the confidence score of the prediction—typically a value between 0 and 1. The confidence maps are then processed to produce the coordinates of each keypoint.

In the case of *multiple pose estimation* where an image contains many individuals, the global geometry of the posture graph is also predicted by training the model to produce *part affinity fields* (**Cao et al., 2017**)— vector fields drawn between joints in the posture graph—or *pairwise terms* (**Insafutdinov et al., 2016**)—vector fields of the conditional distributions between posture keypoints (e.g. $p(\text{foot}|\text{head})$). This allows multiple posture graphs to be disentangled from the image using graph partitioning as the vector fields indicate the probability of the connection between joints (see **Cao et al. 2017** for details).

Box 1. Encoder-decoder models

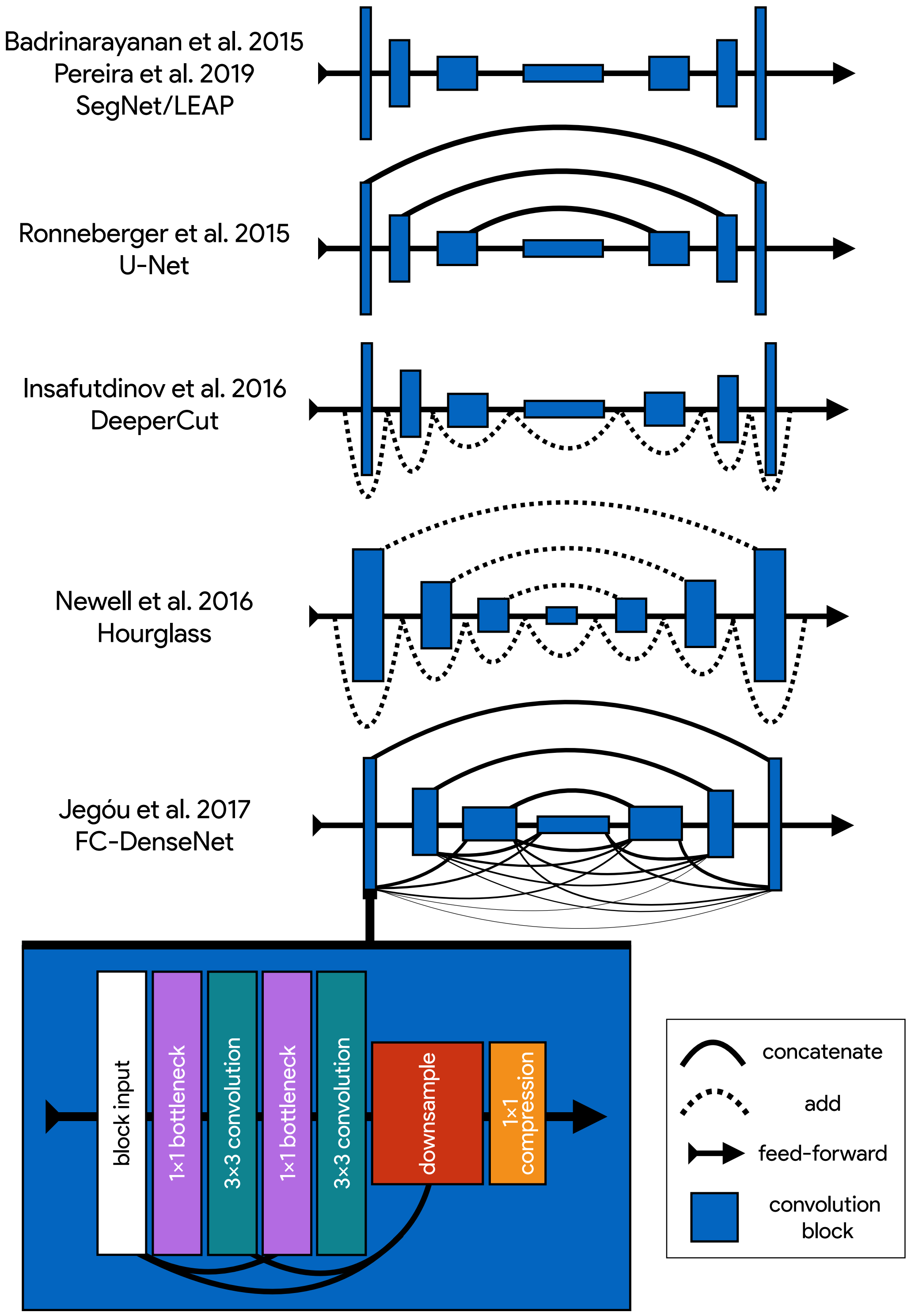


Box 1 Figure 1. An illustration of the basic encoder-decoder design. The encoder converts the input images into spatial features, and the decoder transforms spatial features to the desired output.

A popular type of F-CNN (Appendix 3) for solving posture regression problems is known as an *encoder-decoder* model (Figure 1), which first gained popularity for the task of semantic segmentation—a supervised computer vision problem where each pixel in an image is classified into a one of several labeled categories like "dog", "tree", or "road" (***Long et al., 2015***). This model is designed to repeatedly convolve and downsample input images in the bottom-up *encoder* step and then convolve and upsample the encoder's output in the top-down *decoder* step to produce the final output. Repeatedly applying convolutions and non-linear functions, or *activations*, to the input images transforms pixel values into higher-order spatial features, while downsampling and upsampling respectively increases and decreases the scale and complexity of these features.

Badrinarayanan et al. (***2015***) were the first to popularize a form of this model —known as *SegNet*— for semantic segmentation. However, this basic design is inherently limited because the decoder relies solely on the downsampled output from the encoder, which restricts the features used for predictions to those with the largest spatial scale and highest complexity. For example, a very deep network might learn a complex spatial pattern for predicting "grass" or "trees", but because it cannot directly access information from the earliest layers of the network, it cannot use the simplest features that plants are green and brown. Subsequent work by ***Ronneberger et al.*** (***2015***) improved on these problems with the addition of *residual* or *skip connections* between the encoder and decoder, where feature maps from encoder layers are concatenated to those decoder layers with the same spatial scale. This set of connections then allows the optimizer, rather than the user, to select the most relevant spatial scale(s) for making predictions.

Jégou et al. (***2017***) are the latest to advance the encoder-decoder paradigm. These researchers introduced a fully-convolutional version of ***Huang et al.***'s (2017a) *DenseNet* architecture known as a *fully-convolutional DenseNet*, or *FC-DenseNet*. FC-DenseNet's key improvement is an elaborate set of feed-forward residual connections where the input to each convolutional layer is a concatenated stack of feature maps from *all previous layers*. This densely-connected design was motivated by the insight that many state-of-the-art models learn a large proportion of redundant features. Most CNNs are not designed so that the final output layers can access all feature maps in the network simultaneously, and this limitation causes these networks to "forget" and "relearn" important features as the input images are transformed to produce the output. In the case of the incredibly popular ResNet-101 (***He et al., 2016***) nearly 40% of the features can be classified as redundant (***Ayinde and Zurada, 2018***). A densely-connected architecture has the advantages of reduced feature redundancy, increased feature reuse, enhanced feature propagation from early layers to later layers, and subsequently, a *substantial reduction* in the number of parameters needed to achieve state-of-the-art results (***Huang et al., 2017a***). Recent work has also shown that DenseNet's elaborate residual connections also have the pleasant side-effect of convexifying the loss landscape during optimization (***Li et al., 2018***), which allows for faster optimization and increases the likelihood of reaching a good optimum.



Appendix 3 Figure 1. An illustration showing the progression of encoder-decoder architectures from the literature—ordered by performance from top to bottom (see Appendix 3 Box 1 for further details). Most advances in performance have come from adding connections between layers in the network, culminating in FC-DenseNet from ***Jégou et al.*** (***2017***). Lines in each illustration indicate connections between convolutional blocks with the thickness of the line indicating the magnitude of information flow between layers in the network. The size of each convolution block indicates the relative number of feature maps (width) and spatial scale (height). The callout for FC-DenseNet (***Jégou et al. 2017***; **bottom-left**) shows the elaborate set of skip connections within each densely-connected convolutional block as well as our additions of bottleneck and compression layers (described by ***Huang et al. 2017a***) to increase efficiency (Appendix 8)

Appendix 4

Individual vs. multiple pose estimation

Most recent state-of-the-art methods for posture estimation now focus on simultaneously estimating the pose of multiple individuals in an image (e.g. ***Cao et al. 2017***)—known as *multiple pose estimation*. However, the majority of work on multiple pose estimation has not adequately solved the tracking problem of linking individual data across frames in a video, especially after visual occlusions—although recent work has attempted to address this problem (***Iqbal et al., 2017***; ***Andriluka et al., 2018***). Reliably tracking individuals is important for most behavioral studies, and there are a number of diverse methods already available for solving this problem (***Pérez-Escudero et al., 2014***; ***Crall et al., 2015***; ***Graving, 2017***; ***Romero-Ferrero et al., 2018***; ***Wild et al., 2018***; ***Boenisch et al., 2018***). Therefore, to avoid solving an already-solved problem, the work we describe in this paper is purposefully limited to *individual pose estimation* where each image contains only a single focal individual—which may be localized and cropped from a larger multi-individual image.

We created a top-down posture estimation framework that can be easily adapted to any data collection workflow, which could include any method for localizing and tracking individuals. Limiting our methods in this way also simplifies the pose detection problem and the cognitive task of creating annotated data. Additionally, because individual pose estimation is such a well-studied problem in computer vision, we can build on the state-of-the-art for this task (see Appendices 3 and 5 for details).

Appendix 5

The state of the art for individual pose estimation

The current state of the art for individual posture estimation is ***Newell et al.*** (***2016***)— as well as the many iterations of this design (e.g. ***Ke et al. 2018***, ***Chen et al. 2017***; also see benchmark results from ***Andriluka et al. 2014***). ***Newell et al.*** (***2016***) employ what they call a *Stacked Hourglass* network (Appendix 3 Figure 1), which consists of a series of multi-scale encoder-decoder *hourglass* modules connected together in a feed-forward configuration (Figure 2). The main novelties these researchers introduce include (1) stacking multiple hourglass networks together for repeated top-down-bottom-up inference, (2) using convolutional blocks based on the ResNet architecture (***He et al., 2016***) with residual connections between the input and output of each block, and (3) using residual connections between the encoder and decoder (similar to ***Ronneberger et al. 2015***) with residual blocks in between. ***Newell et al.*** (***2016***) also apply a technique known as *intermediate supervision* (Figure 2) where the loss function used for model training is applied to the output of each hourglass as a way of improving optimization across the model's many layers. Recent work by ***Jégou et al.*** (***2017***) has further improved on this encoder-decoder design (see Appendix 3 Box 1 and Appendix 3 Figure 1), but to the best of our knowledge, the model introduced by ***Jégou et al.*** (***2017***) has not been previously applied to pose estimation.

Appendix 6

Overparameterization and the limitations of LEAP

Overparameterization is a key limitation for many pose estimation methods, and addressing this problem is critical for high-performance applications. ***Pereira et al.*** (***2019***) approached this problem by designing their LEAP model after the model from ***Badrinarayanan et al.*** (***2015***), which is a straighforward encoder-decoder design (Appendix 3 Figure 1; Appendix 3 Box 1). They benchmarked their model on posture estimation tasks for laboratory animals and compared performance with the more-complex Stacked Hourglass model from ***Newell et al.*** (***2016***). They found their smaller, simplified model achieved equal or better median accuracy with dramatic improvements in inference speed up to 185 Hz. However, ***Pereira et al.*** (***2019***) first rotationally and translationally aligned each image to improve performance, and their reported inference speeds do not include this computationally expensive preprocessing step. Additionally, rotationally and translationally aligning images is not always possible when the background is complex or highly-variable—such as in field settings—or the study animal has a non-rigid body. This limitation makes LEAP (***Pereira et al., 2019***) unsuitable in many cases. While their approach is simple and effective for a multitude of experimental setups, the LEAP model from ***Pereira et al.*** (***2019***) is also implicitly limited in the same ways as ***Badrinarayanan et al.***'s SegNet model (see Appendix 3 Box 1 for details). LEAP cannot make predictions using multiple spatial scales and is not robust to data variance such as rotations (***Pereira et al., 2019***).

Appendix 7

Fitting linear models with Stan

We estimated the joint posterior $p(\theta_\mu, \theta_\phi | X, y)$ for each model using the No-U-Turn Sampler (NUTS; ***Hoffman and Gelman 2014***), a self-tuning variant of the Hamiltonian Monte Carlo (HMC) algorithm (***Duane et al., 1987***), implemented in Stan (***Carpenter et al., 2017***). We drew HMC samples using 4 independent Markov chains consisting of 1,000 warm-up iterations and 1,000 sampling iterations for a total of 4,000 sampling iterations. To speed up sampling, we randomly subsampled 20$\%$ of the data from each replicate when fitting each linear model, and we fit each model 5 times to ensure the results were consistent. All models converged without any signs of pathological behavior. We performed a posterior predictive check by visually inspecting predictive samples to assess model fit. For our priors we chose relatively uninformative distributions $\theta_\mu \sim Cauchy(0, 5)$ and $\theta_\phi \sim Cauchy(0, 10)$, but we found that the choice of prior generally did not have an effect on the final result due to the large amount of data used to fit each model.

Appendix 8

Stacked DenseNet

Our Stacked DenseNet model consists of an initial 7×7 convolutional layer with stride 2, to efficiently downsample the input resolution—following ***Newell et al.*** (***2016***)—followed by a stack of densely-connected hourglasses with intermediate supervision (Appendix 5) applied at the output of each hourglass. We also include hyperparameters for the bottleneck and compression layers described by ***Huang et al.*** (***2017a***) to make the model as efficient as possible. These consist of applying a 1×1 convolution to inexpensively compress the number of feature maps before each 3×3 convolution as well as when downsampling and upsampling (see ***Huang et al. 2017a*** and Appendix 3 Figure 1 for details).

Model hyperparameters

For our Stacked Hourglass model we used a block size of 64 filters (64 filters per 3×3 convolution) with a bottleneck factor of 2 (64/2 = 32 filters per 1×1 bottleneck block). For our Stacked DenseNet model we used a growth rate of 48 (48 filters per 3×3 convolution), a bottleneck factor of 1 (1×growth rate = 48 filters per 1×1 bottleneck block), and a compression factor of 0.5 (feature maps compressed with 1×1 convolution to $0.5m$ when upsampling and downsampling, where m is the number of feature maps). For our Stacked DenseNet model we also replaced the typical configuration of batch normalization and ReLU activations (***Goodfellow et al., 2016***) with the more recently developed self-normalizing SELU activation function (***Klambauer et al., 2017***), as we found this modification increased inference speed. For LEAP (***Pereira et al., 2019***) we used a 1× resolution output with integer-based global maxima because we wanted to compare our more complex models with LEAP in the original configuration described by ***Pereira et al.*** (***2019***). Additionally, applying our subpixel maxima algorithm at high resolution reduces inference speed compared to integer-based maxima, so this would bias our speed comparisons.

Our implementation of DeeperCut

Because the original DeeperCut model from ***Insafutdinov et al.*** (***2016***) was not implemented in Keras (a requirement for our pose estimation framework), our implementation of this model does not exactly match the description in the paper. Implementing this model directly in our framework is important to ensure model training and data augmentation are identical when making comparisons. Nevertheless we assume our version is nearly identical in performance due to the very similar architecture and number of parameters—a ResNet-50 (***He et al., 2016***) encoder pretrained on the ImageNet database (***Deng et al., 2009***) with a matching ResNet decoder. Additionally, pretrained feature detectors in the Keras framework have a lower bound on input image size (224×224), so we were forced to add an upsampling layer before the encoder to double the resolution of the input when evaluating the model. Therefore speed comparisons between this model and the others may be biased. To account for this, our implementation of ***Insafutdinov et al.*** (***2016***) takes advantage of our fast subpixel maxima algorithm to increase speed, so our overall comparisons are reasonable regardless of these constraints. Our reported inference speeds for our datasets also match well with results from ***Mathis and Warren*** (***2018***) who evaluated the inference speed of DeeperCut (***Insafutdinov et al., 2016***) for multiple image sizes.

Appendix 9

Depthwise-separable convolutions for memory-limited applications

In an effort to maximize model efficiency, we also experimented with replacing 3×3 convolutions in our model implementations with 3×3 depthwise-separable convolutions —first introduced by ***Chollet*** (***2017***) and now commonly used in fast, efficient "mobile" CNNs (e.g. ***Sandler et al. 2018***). In theory this modification should both reduce the memory footprint of the model and increase inference speed. However we found that, while this does drastically decrease the memory footprint of our already memory-efficient models, it slightly decreases accuracy and does not improve inference speed, so we opt for a full 3×3 convolution instead. We suspect that this discrepancy between theory and application is due to inefficient implementations of depthwise-separable convolutions in many popular deep learning frameworks, which will hopefully improve in the near future. At the moment we include this option as a hyperparameter for our Stacked DenseNet model, but we recommend using depthwise-separable convolutions only for applications that require a small memory footprint such as training on a lower-end GPU with limited memory or running inference on a mobile device.