Computer Vision and Deep Learning Methods for Measuring and Modeling Animal Behavior

Doctoral thesis for obtaining the academic degree

Doctor of Natural Sciences
Dr.rer.nat.

submitted by

Graving, Jacob M.

at the

Faculty of Sciences

Department of Biology

Konstanz, 2020

Contents

Summary

The study of animal behavior is a fundamental pursuit for answering scientific questions across a variety of fields — including neuroscience, psychology, ecology, genetics, and evolution. While the task of collecting accurate and complete behavioral data has typically always been difficult, laborious, and subjective, in recent years there has been rapid progress in methods for automatically quantifying behavior objectively and at scale. This progress has been primarily driven by the emergence of new computational hardware, software, and algorithms for measuring behavior. In order to reveal core insights about how animals organize their behavior with the increased quality and resolution of these data comes the need for new methods for data-driven modeling. Here, in this thesis, I focus on computational tools—the development of new algorithms and software—for measuring and modeling behavior using methods from computer vision, deep learning, Bayesian inference, and probabilistic programming, while also synthesizing these approaches with ideas from other relevant areas such as information theory, nonlinear dynamics, and statistical physics. First, I developed a barcode tracking system for automated behavioral studies where the location and identity of individuals can be reliably tracked over several weeks (or potentially longer) using conventional computer vision (Chapter 1). Next, I developed general-purpose deep learning methods for measuring animal posture — any set of user-selected body parts — in the laboratory and field (Chapter 2). Finally, I introduce methods for using these posture data to model behavior with techniques from machine learning and Bayesian statistical inference (Chapter 3). Together these methods reduce barriers to measuring and modeling animal behavior and allow researchers to answer scientific questions that were previously intractable.

Zussamenfassung

Das Studium des Verhaltens von Tieren ist ein grundlegendes Unterfangen zur Beantwortung wissenschaftlicher Fragen in einer Vielzahl von Bereichen — einschließlich Neurowissenschaften, Psychologie, Ökologie, Genetik und Evolution. Während die Aufgabe, genaue und vollständige Verhaltensdaten zu sammeln, in der Regel immer schwierig, mühsam und subjektiv war, gab es in den letzten Jahren rasche Fortschritte bei den Methoden zur automatischen objektiven und maßstabsgetreuen Quantifizierung von Verhalten. Dieser Fortschritt wurde in erster Linie durch das Aufkommen neuer rechnergestützter Hardware, Software und Algorithmen zur Verhaltensmessung vorangetrieben. Um zentrale Erkenntnisse darüber zu gewinnen, wie Tiere ihr Verhalten mit der verbesserten Qualität und Auflösung dieser Daten organisieren, werden neue Methoden zur datengesteuerten Modellierung benötigt. In dieser Dissertation konzentriere ich mich in dieser Arbeit auf rechnergestützte Werkzeuge - die Entwicklung neuer Algorithmen und Software - zur Messung und Modellierung von Verhalten unter Verwendung von Methoden aus den Bereichen Computersehen, Deep-Learning, Bayes'sche Inferenz und probabilistische Programmierung, wobei ich diese Ansätze auch mit Ideen aus anderen relevanten Bereichen wie Informationstheorie, nichtlineare Dynamik und statistische Physik zusammenführe. Zuerst entwickelte ich ein Strichcode-Verfolgungssystem für automatisierte Verhaltensstudien, bei dem der Aufenthaltsort und die Identität von Personen über mehrere Wochen (oder potenziell länger) mit konventionellem Computersehen zuverlässig verfolgt werden kann (Kapitel 1). Als Nächstes entwickelte ich Mehrzweck-Deep-Learning-Methoden für die Messung der Haltung von Tieren — jede Menge vom Benutzer ausgewählter Körperteile — im Labor und vor Ort (Kapitel 2). Schließlich stelle ich Methoden zur Verwendung dieser Haltungsdaten zur Verhaltensmodellierung mit Techniken des maschinellen Lernens und der statistischen Bayes'schen Inferenz vor (Kapitel 3). Zusammen verringern diese Methoden die Barrieren bei der Messung und Modellierung des Verhaltens von Tieren und ermöglichen es den Forschern, wissenschaftliche Fragen zu beantworten, die zuvor unlösbar waren.

Introduction

Measuring and modeling behavior are fundamental problems for the study of ecology and neuroscience (Berman, 2018; Brown & De Bivort, 2018). In general, these methodological areas seek to develop techniques capable of answering a straightforward set of questions: Where are individuals located in space? Who are those individuals? And what are they doing? Perhaps unsurprisingly, these three fundamental questions overlap with similar unsolved problems in computer vision (Dell et al., 2014; LeCun et al., 2015) including object detection and tracking, facial recognition and object classification, as well as pose estimation and action recognition. Despite the concrete, real-world nature of these problems, only very recently have the computational algorithms for solving these tasks reached the same level of quality as that of manual human annotation (Goodfellow et al., 2016; LeCun et al., 2015). This increase in quality is largely due to the democratization and wide-spread adoption of deep learning algorithms, a class of computational models that are capable of automatically detecting complex patterns and can scale to arbitrarily large and high-dimensional datasets (Goodfellow et al., 2016; LeCun et al., 2015). Consequently, a large body of computer science literature with general-purpose algorithms for solving these previously-unsolved tasks now exists, and this progress is only just beginning to be adapted to study animal behavior (Graving et al., 2019a; Günel et al., 2019; A. Mathis et al., 2018; Pereira et al., 2019) — which brings with it many additional unsolved problems (Graving et al., 2019a; A. Mathis, Schneider, et al., 2020; M. W. Mathis & Mathis, 2020).

The overarching goal of this thesis has been to contribute to this growing body of work by developing general-purpose, open-source software and algorithms that allow researchers to apply computational tools to measure and model behavior as well as to illustrate the power of these tools to address questions that were previously unanswerable. Over the course of my PhD, I built on the existing deep learning and computer vision literature to develop new software and algorithms for solving these problems. This involved collaborations with other researchers to apply these methods to detect, track, and estimate the body posture of animals both in the

laboratory and the field. Additionally, I developed methods for summarizing and interpreting these data using probabilistic deep learning models, which draw on fundamental ideas from Bayesian statistics and information theory. Together these methods have allowed researchers to measure behavior in experimental scenarios that were previously intractable and have helped to make meaningful biological inferences using large, complex, high-dimensional data sets.

Just a couple decades ago, the use of computer vision for the study of animal behavior was relatively uncommon (Dell et al., 2014). However, as computational hardware and software have matured and became more accessible, so have the methods used to quantify movement and extract insights from behavioral experiments. Studying behavior with computer vision methods has become increasingly common in recent years, and the field of animal behavior has evolved rapidly in response (Anderson & Perona, 2014; Berman, 2018; Brown & De Bivort, 2018). This is an extremely exciting and active research area with researchers continuously advancing a range of methods to automatically measure behavior in almost any scenario (Anderson & Perona, 2014; Dell et al., 2014; Graving et al., 2019a; M. W. Mathis & Mathis, 2020). For example, many of the conventional approaches developed only a few years ago are now almost completely obsolete. Some of the most prevalent approaches now focus on the use of deep learning for the measurement and modeling of animal locomotion (Graving et al., 2019a; Graving & Couzin, 2020; Luxem et al., 2020; M. W. Mathis & Mathis, 2020) — including automated motion capture and tracking of animal movements in the laboratory and in the wild (Francisco et al., 2020; Graving et al., 2019a; Nath et al., 2019a).

Deep learning has begun to revolutionize science as we know it (LeCun et al., 2015). Rapid advances in hardware and software, combined with a research community focused on open-source and open-access philosophies, have democratized this class of machine learning algorithms. Consequently, this has made the use of deep learning tools commonplace across the natural sciences. These general-purpose algorithms have been applied to a wide array of problems in computer vision, natural language processing, and audio recognition (reviewed by LeCun et al. 2015). With the right data, a deep learning model has the potential to solve nearly any computational task with relatively minimal effort—even those tasks once thought to always require human-level intelligence. The study of animal behavior is no different, and deep learning has dramatically impacted this interdisciplinary field in a short period of time (Graving et al., 2019a; M. W. Mathis & Mathis, 2020). Deep learning has already revolutionized the study of animal behavior as a tool for both quantifying behavior (Graving et al., 2019a; A. Mathis et

al., 2018; Pereira et al., 2019; Romero-Ferrero et al., 2019) as well as understanding behavior through data-driven modeling (Graving & Couzin, 2020; M. Johnson et al., 2016; Luxem et al., 2020) and will continue to advance the study of behavior for the foreseeable future.

Here, in this thesis, I provide an update on the status of computer vision-based methods in the behavioral sciences. I focus on a few of the most prominent methods in this area of research, highlight some of the most promising applications for these technologies, and introduce several advances in the use of computer vision and deep learning-based methods to automatically measure animal locomotion and body posture. In Chapter 1, I discuss methods for tracking the behavior of animals with conventional computer vision techniques. I provide an overview of existing methods and discuss the motivations for using computer vision over other methods for behavioral quantification. Then I software for marker-based spatial localization and individual identification using 2D barcode tags (Graving, 2017), and, in collaboration with colleagues, apply this software to automatically track large groups of individuals for long-term behavioral studies in captive bird populations (Alarcón-Nieto et al., 2018). In this work, we also demonstrate how the data can be used to reconstruct social networks. This system has already been used to reveal fundamental insights about the behavior of these bird populations (Maldonado-Chaparro et al., 2018), and will continue to do so in the future.

In Chapter 2, I develop image-based approaches using deep learning to measure detailed information about animal body posture. These methods were only recently adapted from the computer science literature on human pose estimation and have seen rapid progress since then. I discuss how these approaches compare with conventional methods for behavioral quantification, and I review currently available methods for deep learning-based pose estimation. Additionally, I identify several gaps in these existing methods and introduce a deep learning-based software toolkit, called DeepPoseKit, with general-purpose methods for the automated measurement of animal body posture. Within this toolkit, several technical advances are introduced to the field of pose estimation that address many of the limitations with existing methods, and, together with colleagues, we compare these methods with previous techniques and apply them to new datasets (Graving et al., 2019a). These methods have already been applied to ongoing research projects. For example, I am collaborating with colleagues to study energy saving strategies in schooling fish as well as collective predator detection in herds of ungulates in the wild — two unsolved research topics that would not have been tractable without these methods.

In Chapter 3, I discuss methods for decomposing these detailed postural data into their

underlying components by identifying stereotyped behavioral motifs, or behavioral modes, directly from data. To accomplish this, I introduce a new deep learning model, called VAE-SNE, which serves as a tool for the general-purpose analysis and interpretation of high-dimensional data. I then demonstrate the utility of this algorithm for different tasks across multiple domains in the life sciences. In particular, I use VAE-SNE to perform unsupervised action recognition by automatically segmenting high-dimensional behavioral data into known stereotyped actions, such as locomotion, grooming etc., by automatically compressing and clustering high-dimensional body posture dynamics.

Throughout the thesis, I discuss how modern computer vision-based methods, together with other computational technologies, such as virtual reality and drone-based imaging, allow for systems for real-time, automated, and efficient measurement and modeling of complex behavioral patterns, such as collective and social behavior, both in laboratory and field settings. In addition to providing an overview of current methods that utilize deep learning for studying behavior, I discuss the future of deep learning as a set of tools for measuring and modeling the behavior of animals across multiple scales of biological complexity as well as how we might evolve, and improve upon, existing methods.

Chapter 1

An automated barcode tracking system for behavioural studies in birds

Gustavo Alarcón-Nieto*, Jacob M. Graving*, James A. Klarevas-Irby*, Adriana A. Maldonado-Chaparro, Inge Mueller, Damien R. Farine

***equal contribution**

Adapted from: Alarcón-Nieto, G.*, Graving, J. M.*, Klarevas-Irby, J. A.*, Maldonado-Chaparro, A. A., Mueller, I., Farine, D. R. (2017). An automated barcode tracking system for behavioural studies in birds. bioRχiv, 201590 under a CC-BY-4.0-NC-ND International License

Published as: Alarcón-Nieto, G.*, Graving, J. M.*, Klarevas-Irby, J. A.*, Maldonado-Chaparro, A. A., Mueller, I., Farine, D. R. (2018). An automated barcode tracking system for behavioural studies in birds. Methods in Ecology and Evolution, 9(6), 1536-1547. © 2018 Methods in Ecology and Evolution © 2018 British Ecological Society

1.1 Abstract

1. Recent advances in technology allow researchers to automate the measurement of animal behaviour. These methods have multiple advantages over direct observations and manual data input as they reduce bias related to human perception and fatigue, and deliver more extensive and complete datasets that enhance statistical power. One major challenge that automation can overcome is the observation of many individuals at once, enabling whole-group or whole-population tracking.

2. We provide a detailed description of an automated system for tracking birds. Our system uses printed, machine-readable codes mounted on backpacks. This simple, yet robust, tagging system can be used simultaneously on multiple individuals to provide data on bird identity, position and directionality. Furthermore, because the backpacks are printed on paper, they are very lightweight. We show that our method is reliable, relatively easy to implement and monitor, and with proper handling, has proved to be safe for the birds over long periods of time.

3. We describe the deployment procedure of this system for a captive population of songbirds. We test different camera options, and discuss their advantages and disadvantages. In particular, we highlight how using single-board computers to control the frequency and duration of image capture makes this system affordable and adaptable to a range of study systems and research questions.

4. The ability to automate the measurement of individual positions has the potential to significantly increase the power of both observational and experimental studies. The system can capture both detailed interactions (using video recordings) and repeated observations (e.g. once per second for the entire day) of individuals over long timescales (months or potentially years). This approach opens the door to tracking life-long relationships among individuals, while also capturing fine-scale differences in behaviour.

1.2 Introduction

Studying behaviour is central to addressing a broad range of research questions in the fields of neurobiology, ecology and evolutionary biology. Nevertheless, collecting accurate and complete behavioural data remains a challenging task (Crall, Gravish, Mountcastle, & Combes, 2015).

Although direct observation is still an important method for gathering data, a variety of automated methods are now frequently used to accelerate data collection and reduce the effects of human intervention. Video recording has become common practice for studying both captive (Ihle, Kempenaers, & Forstmeier, 2015; Nagy et al., 2013; Perez-Escudero, Vicente-Page, Hinz, Arganda, & de Polavieja, 2014; Rojas Mora, Forstmeier, & Fusani, 2014; Togasaki et al., 2005) and wild organisms (Scheibe, Eichhorn, Wiesmayr, Schonert, & Krone, 2008; Togasaki et al., 2005). However, manually measuring behaviour from photos or videos is extremely time consuming and may still have the same limitations as direct observations, such as cognitive bias and fatigue. Manually identifying individuals is also challenging, which limits the use of this approach to species with individually distinct features (Perez-Escudero et al., 2014). Recent advances in automated, image-based tracking methods have solved these issues in a variety of ways. Unfortunately, many of these solutions rely on complex, computationally intense algorithms, often require keeping animals in simplistic, unnatural environments, and may not reliably preserve identities over long periods of time or across contexts (Perez-Escudero et al., 2014). One alternative, which has been explored in a few recent studies (e.g. Mersch, Crespi, & Kelle, 2013; Nagy et al., 2013) is to fit machine-readable tags to individuals, allowing for faster, more reliable tracking. This method offers exciting new opportunities, such as studying social behaviour in complex, naturalistic environments, over long timescales, and across multiple behavioural contexts. Here, we provide details of how to implement such a system for songbirds.

The development of methods for tracking individuals plays an important role in our ability to study animals. In addition to the limitations of human observers to process multiple streams of information simultaneously (such as the actions of several individuals in a group), many studies still rely on using relatively small datasets to estimate broad patterns. One example is the use of focal follows, where a single individual is tracked for a period and all of its interactions with others are recorded. While doing so, all the interactions among others are not recorded. This means that even with very intensive monitoring, the maximum number of dyadic observations that can be made is $N - 1$, where N is the number of individuals present. Sparseness in the resulting datasets can impact the ability to successfully test hypotheses (Farine & Strandburg-Peshkin, 2015). Furthermore, these studies can suffer from temporal autocorrelation (most data on a focal is collected within a short period of time; Whitehead, 2008). Studies that cannot extract data with sufficient resolution also lead to concerns about the use of animals in research if they cannot robustly test the hypothesis, as sparse data collection can heighten the rates of true and

false positives.

Multiple technologies enable more detailed tracking of individuals than what is possible by manual observation. For example, modern studies of migration and habitat-use commonly rely on satellite and radio tags to estimate individual position over time (e.g. Abedi-Lartey, Dechmann, Wikelski, Scharf, & Fahr, 2016; Klaassen et al., 2014; Rotics et al., 2017). When combined with accelerometers and other sensors, these tags can also measure fine-scale behaviour and physiology (Wang, Smith, & Wilmers, 2017). Recent developments even allow for collecting high-resolution trajectory data over large areas (Kays, Crofoot, Jetz, & Wikelski, 2015), which can be used to infer social interactions in mobile, group-living species (Strandburg-Peshkin, Farine, Couzin, & Crofoot, 2015). Despite their advantages, such systems are generally very costly and the tags are often too heavy for many species to carry, which limits the number of individuals that can be tracked and the size of the animals that can be studied.

An increasingly common method for tracking smaller animals is Passive Integrated Transponder (PIT) tags (Boarman, Beigel, Goodlett, & Sazaki, 1998). These tags provide a unique identity when in range of a radio frequency identification antenna. PIT tags are lightweight, inexpensive and require no battery power, enabling large-scale deployment over long periods of time, and they can be used in both laboratory (Boogert, Farine, & Spencer, 2014; Farine, Spencer, & Boogert, 2015; Griffith, Holleley, Mariette, Pryke, & Svedin, 2010; Weissbrod et al., 2013) and field conditions (Adelman, Moyers, Farine, & Hawley, 2015; Aplin et al., 2015; Bonter & Bridge, 2011; Broderick & Godley, 1999; Farine, Aplin, Garroway, Mann, & Sheldon, 2014; König et al., 2015; Mariette et al., 2011; Steinmeyer, Mueller, & Kempenaers, 2013). Although many individuals can be tagged, the antennas can only detect one individual at a time and only at fixed focal locations, such as nest boxes (Santema, Schlicht, Schlicht, & Kempenaers, 2017; Schlicht, Valcu, & Kempenaers, 2015), feeders (Firth, Sheldon, & Farine, 2016), or puzzle-boxes (Aplin et al., 2015), which limits resolution for assessing interactions among individuals.

Machine vision hardware and software allow for automated, image-based tracking of animals (Dell et al., 2014; Jolles, Boogert, Sridhar, Couzin, & Manica, 2017; Perez-Escudero et al., 2014; Rosenthal, Twomey, Hartnett, Wub, & Couzin, 2015). However, when tracking groups of animals, maintaining individual identities can be difficult (Perez-Escudero et al., 2014). Some algorithms can identify unmarked individuals (e.g. Berger-Wolf et al., 2015), even using subtle differences in coloration (e.g. Perez-Escudero et al., 2014), although these methods are less effective if such features change over time (e.g. bird feathers move). Several studies on social insects have used

machine-recognizable 2D barcodes (hereafter barcodes; Crall et al., 2015; Greenwald, Segre, & Feinerman, 2015; Mersch et al., 2013) with a unique pattern of black and white squares that can be identified and matched to a dictionary of known codes. Insects are good models for using such markers because these barcodes can be directly glued onto their bodies, and they can be applied to hundreds of individuals simultaneously because the tags are inexpensive to make (using only waterproof paper). Similar approaches have been used on birds (Nagy et al., 2013), but few details are available on their implementation and long-term impact on individuals.

Barcodes represent a major advance in data quality at a comparatively low cost, enabling researchers to simultaneously collect data from multiple individuals, assess interactions and associations in different contexts, and conduct experiments in complex, naturalistic environments. Here, we describe how to implement such a tracking system for songbirds in a captive experimental setup, which allows for tracking of individuals' positions and orientations over time. We particularly focus on the design and deployment procedures for backpack-mounted barcodes to ensure bird safety and reliable data collection, as well as the required monitoring and maintenance of the system over long periods of time. We discuss the materials used, different camera systems for capturing image data, and other considerations associated with data collection, with a particular focus on how to implement this system cheaply and effectively. We then provide details on the process of extracting data from the images, and what software is available for this purpose, highlighting opportunities for automation of the entire processing pipeline. Finally, we discuss potential behaviours that can be measured, using such a system and possible applications in further studies.

1.3 Methods

1.3.1 Study population

We tested our barcode tracking system on domesticated zebra finches Taeniopygia guttata. The zebra finch is a model species widely used in behavioural studies (Boogert et al., 2014; David, Auclair, & Cezilly, 2011; Farine et al., 2015; Kriengwatana, Spierings, & ten Cate, 2016; Mariette & Griffith, 2012; Ruploh, Bischof, & von Engelhardt, 2014; Schuett, Dall, & Royle, 2011; Wuerz & Kruger, 2015). They are social birds, living in colonies of 50–100 individuals (Zann, 1994), and in captivity can be kept in large groups, which makes them a suitable organism to test our tracking system.

We tested our system on two flocks of domesticated zebra finches, held in separate indoor aviaries in the Max Planck Institute for Ornithology in Radolfzell, Germany, with indoor aviary lighting turned on from 8.00 until 18.00 hr. Each flock was held in a 2 $\times$ 2 $\times$ 2-m metal-mesh cage and provided with a complex arrangement of branches, feeders, drinking water, a bathing tray and wood chips as floor cover. We supplied both millet seeds and water ad libitum, except during food-based assays (see below). No nesting material or nest boxes were available during the length of our trials to prevent the birds from breeding. Each flock consisted of 28 adult individuals in 1:1 sex ratio. We tested several prototype backpacks between September and November 2016. From December 2016 through to the end of March 2017, we fit the backpacks described in this paper to all members of the flocks (except those that could not take a backpack, see below). Birds therefore carried backpacks for up to 4 months, with some individuals carrying backpacks continuously over a period of up to 7 months. Each bird was also fitted with leg bands for identification, consisting of a numbered closed metal band and two plastic bands in a colour combination that was unique in each aviary. This study was conducted under Ethics Permit 35-9185.81/G16/73 issued by the state of Baden-Württemberg, Germany.

1.3.2 Barcode tracking system

The barcode tracking system consists of three components: (1) a backpack fitted with a barcode; (2) recording device(s), and (3) processing software and hardware. In this section, we describe the design of the backpack (i.e. structure carrying the barcode), its fitting procedure (i.e. deployment) and the monitoring and maintenance of the codes.

Backpack design

Backpacks consist of three main parts: the backpack structure and tag mount, the tray, and the straps (Figure 1.1). We constructed the structure using 70 10-mm strips of waterproof and tearproof paper (Xerox®-Premium Never Tear-95 μm). We built this structure by laser printing templates on an A4 sheet of paper (Figure 1.1a, template provided in Supporting Information 1). Each template was cut out, folded and glued into a loop to form the tag mount (Figure 1.1b,f), which provided a raised surface to keep the barcode above the feathers. We then 3D-printed black plastic trays (Figure 1.1c, Supporting Information 2) into which we glued the barcodes (printed on the same type of paper as the backpacks, Figure 1.1d). The black plastic is an important feature as it reinforces the border that frames the barcode and prevents the birds from

damaging the edges, which makes the code unreadable by the software. We glued this tray with the code onto the backpack mount (Figure 1.1f). Although a well-deployed backpack should keep this tray behind the wing joints, we rounded the external corners of the tray (Figure 1.1c) to prevent injuries and wing rubbing.

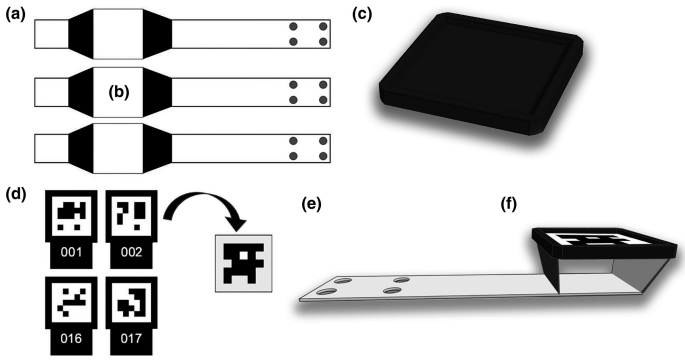

Figure 1.1. Components and assembly of the backpack-mounted barcodes. (a) Template layout. (b) Mount area where plastic trays (c) are glued. (d) Barcode layout and cutout to be glued on the tray. (e) Assembled backpack with tray and code raised on the mount (f)

Backpacks include a front strip of paper that fits between the scapulae of the bird, into which we punched four round holes (c. 1-mm diameter, Figure 1.1e) to attach the elastic string that formed the straps of the backpack around the bird (Figure 1.2). For each backpack, we used a single piece of string 25-cm long which we looped through the rear holes on the paper, crossed under the backpack, tied on the front holes, and kept the leads loose to allow for individual adjustment during deployment. For zebra finches, we used a 28 $\times$ 6-mm front strip, a 10 $\times$ 10 mm mount raised 6 mm, and 10-mm square tray. Backpacks weighed c. 0.27 g when fully assembled, including the straps.

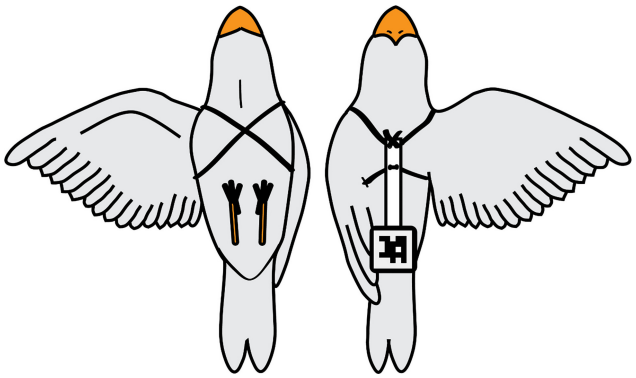

Figure 1.2. Left: bottom view of bird with backpack. The string sits in front and behind the wings and crosses on the chest of the bird. Right: Top view. The string is tied at the anterior end of the backpack which goes between the scapulae, and the mount with the barcode sits on the rump, behind the wing joints

Backpack deployment

The general procedure for fitting backpacks includes the following steps: (1) Catching, measuring and recording health status of each bird; (2) Backpack fitting; (3) Observation during acclimation; (4) Releasing and monitoring birds in their permanent housing; and (5) Periodic health checks.

Once we confirmed the birds were in good health (step 1), we fit a completely assembled backpack to each bird (step 2). We pre-tied the string on the backpack with a simple slipknot and then pulled the straps over the bird's head until the front strip sat on the interscapular area, carefully pulling each wing through the looped straps. We found that the best fit was achieved when the leading edge of the raised section was below the elbow joint of the wing, and the trailing edge was above the rump (Figure 1.2). Once the backpack was in its final position, we tightened the string around the body, adjusting according to the size of each bird. The tightness must be firm enough to hold the backpack in position while preventing the bird to put its feet or toes inside of the loop, but also loose enough to allow the birds to fly and move freely, and to avoid blocking the crop. In their final position, the straps should sit midway between the body and the surface of the feathers. The front strip and the string loops eventually become covered by the feathers, while only the mount with the plastic tray and the barcode are visible. The mount must be positioned behind all the wing bones and joints, where only feathers can be in contact with it.

After fitting the backpacks, we placed subjects in a small observation cage (step 3) to monitor their behaviour. This step is critical for animal safety. We monitored birds for up to 1 hr until we were sure that they behaved normally, that is, with no observable hindering of movement. A well-fitted backpack allows the animal to move freely, with no interference for flying, walking, landing or perching. Most birds tried to pull the backpacks or the straps off during this period. In our experience, the intensity and duration of this behaviour was not necessarily a signal of an ill-fitting backpack and, on the contrary, preening helped to accommodate all the new elements. We found that the acclimation process worked better when subjects were kept in small groups (2–5) and in a separate room with no people present, as it reduced stress and allowed for allopreening, feeding and undisturbed movement. Once we considered the deployment procedure was successful, we made a final check of the adjustments, secured the knot near the neck of the bird a second small knot and cyanoacrylate glue, and cut any excess string from the leads. When necessary, we also trimmed some covert feathers around the mount to prevent any obstructions on the codes that might hinder detection.

Every time we observed a bird with hindered movement or unusual behaviour that could

be related to the backpack (i.e. foot or toes trapped in a loose strap, incapable of flight, or unbalanced perching), we checked and readjusted the straps using blunt-tip tweezers. In some cases, if a bird's movement did not improve after the adjustment, we completely removed the backpack, let the bird rest to reduce stress, and observed it without the backpack before trying another deployment. After a second attempt, a few birds (4 of 58) still showed suboptimal performance despite having a well-fitted backpack and appearing to be in good health. Our testing suggests that fewer than 5% of subjects will never acclimate to the backpacks.

Backpack monitoring

We monitored the birds regularly, either during our experiments or during care-taking activities, and constantly looked for unusual behaviour. This monitoring is important to prevent injuries or detect early symptoms of health issues, either related to the backpacks or otherwise. In our experience, most of the signs that could suggest ill-fitted tags occurred within the first 2 days of observation after deployment and were addressed promptly. Importantly, some issues were only detectable when birds were settled in their permanent housing environment where they could fly much more extensively. We also monitored the birds by assessing the tracking data to identify individuals that were outliers in the number of detections (suggesting they behaved differently to others). The main issue we found arose after release into large aviaries was the backpack rubbing on the body or wings of the bird. Symptoms of this included bald spots on wings or neck, reduced movement or difficulty flying. These were addressed immediately by ensuring the backpack mount (and tray) were correctly fitted (i.e. not crooked and positioned away from the wings). However, in some cases, when the problem persisted, we completely removed the backpack, let the bird rest, and observed its behaviour without the backpack.

1.3.3 Camera systems

Barcodes can be detected using either photos or video. The choice largely depends on the research question to be addressed, as well as the scale of data collection and its associated processing and storage requirements. In this section, we provide details on the necessary considerations for implementing a camera system, and details of our experience using several implementations, including high-resolution photos and video from action cameras, computer-controlled DSLR cameras and the programmable camera module for the Raspberry Pi. We also discuss the pros and cons of each system for different types of research questions.

Code size and capture

For adequate detection and recognition of individual birds in photos or videos, the size of the barcodes should be at least 20 pixels per side in the captured image data (Crall et al., 2015), but this can vary depending on tag design and camera hardware. Detectability of tags can be improved, using high-resolution cameras, reducing the distances between the codes and the camera (either physically or by using zoom lenses), or increasing the physical size of the deployed tags (which is limited by the study organism). Other considerations such as lens distortion, sharpness, and depth of field must be considered depending on the setup and area being captured. Lens distortion can be partially corrected via software, but this correction reduces the effective resolution of the images, especially for wide-angle lenses (Figure 1.3). Depth of field is an important consideration if the birds can perch at different heights, although issues can be avoided by having all perches on the same plane. Finally, the camera shutter speed needs to be chosen carefully. Slow shutter speeds result in blurred or overexposed codes and, thus, failed detections. To prevent these problems, exposure time should be set as short as possible while ensuring that contrast and noise levels are adequate for the software to successfully read the codes. We found that darker images had greater detectability as they increased the clarity of the edges within the barcodes by reducing bleeding of the white areas of the barcode into the black areas.

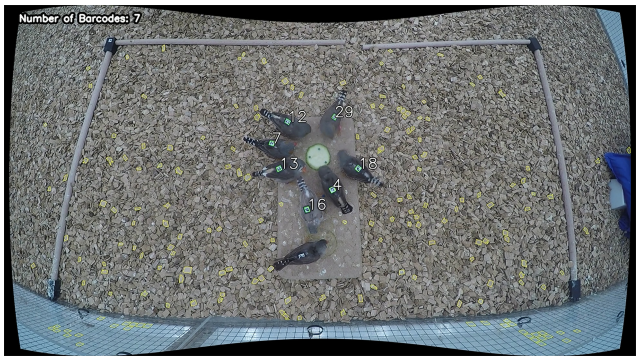

Figure 1.3. Barcode detections in a feeding context in one video frame (from a GoPro camera). The food source in the centre of the arena was pinned to a wooden board to prevent birds from moving it out of the frame. The tracking algorithm detected barcodes on the back of each individual despite being on a complex background (wood chips). The yellow polygons are objects that were detected as candidate barcodes but did not match any known identities. The bird near the bottom of the image was not detected because its wings covered the barcode in this frame. Individual 16 was oriented away from the food. The black edges around image illustrate the software correction we used to partially compensate for wide-angle lens distortion

Photos or video?

Imaging technology requires a trade-off between spatial and temporal resolution as hardware is always limited in its capacity to transmit and process signals. The choice between using photos or videos when capturing image data depends on the spatial or temporal resolution required to answer the research question. Video offers higher temporal resolution (typically 24 Hz) at the expense of spatial resolution (most hardware is limited to 3,840 $\times$ 2,160-pixels), making it more suitable for studying behaviours that occur over short periods of time and in confined areas (e.g. allopreening, aggressive interactions, mating displays or copulation). Photos offer increased spatial resolution ($>$50 megapixels for some DSLR models) at the cost of temporal resolution (most cameras are limited to $<$15 Hz at full resolution), making them ideal for assessing behaviours in larger, open areas that do not require high-frequency measurements (e.g. feeding, bathing, co-perching, co-feeding, etc.) and for quantifying associations over longer timescales (i.e. social networks, pair formation, etc.). Current imaging technologies vary widely in frame rate, image resolution, and file sizes, and different camera setups can be adapted for data collection depending on the research question and project budget. We implemented and tested three types of recording devices.

Action cameras

We used GoPro Hero 4 action cameras to record video of the birds in a feeding arena 90 $\times$ 50 cm on the floor of the aviaries (Figure 1.3). We set the cameras to run continuously until the battery was depleted (c. 45 min) and chose a resolution of 1,920 $\times$ 1,080 pixels (1,080p) at 24 Hz to limit file size, reduce processing time, maximize battery life, and prevent the camera from overheating. We created a 3D-printed arm to attach the camera to the side of the cage, 50 cm above the feeding arena and manually started recordings immediately after providing birds with a high-value food patch (to attract them to this focal area of the camera).

These cameras produced adequate image quality but had noticeable distortion due to the wide-angle fixed lenses. We manipulated the resulting images to reduce distortion before running the detection code (see “extracting data from images”). At 1,080p, we observed that the codes were sharp enough for detection, although at 24 Hz some frames suffer from image blurriness when birds moved. At 1,080p resolution, we generated a 4-GB file every 15–17 min of video, which is the maximum file size supported by the cameras. This means that in a 45-min recording

session, we had to process three videos and store at least 12 GB. Limitations of this setup include the need to manually operate the cameras, restricted recording time due to battery life or large file size, and limited options for automating the entire system. Some of these problems, such as limited storage, lens distortion, and lack of automation, have been mitigated in newer models of the GoPro Hero series, as well as other brands of action cameras.

Digital SLR cameras

We briefly tested data collection using four Canon EOS1200 DSLR Cameras with 18–55 mm lenses for recording video or still images. We connected these cameras to Raspberry Pi 3 single-board computers to control the image capture frequency. We placed the cameras at the top of the aviaries facing directly down. The cameras were set to capture one image at 1/200-s every 10 min to measure the position of birds sitting on perches made from natural branches. These cameras can deliver high quality images up to 5,184 $\times$ 3,456 pixels (18 megapixels) and the zoom lenses allow for easy accommodation to different distances and to cover either small or large areas. However, this model had a loud mechanical shutter, which visibly disturbed the birds in the enclosed aviary space. Rather than modifying or removing the shutters, we abandoned this setup in favour of cameras with electronic shutters and no moving parts. In video mode, DSLR cameras can record high-resolution video (1,080p) which is sufficient for collecting detailed movement data. Unfortunately, the recording time for many models is limited to 30 min.

Single-board computers with camera

We used Raspberry Pi 3 Model Bs (Raspberry Pi Foundation), each fitted with an 8-megapixel Camera Module V2 (RS Components Ltd and Allied Electronics Inc.), to record photos of birds on perches (Figure 1.4). We installed two of these on top of each aviary, covering most of the perch system without overlap. To record the birds present, we set the system to capture one image every 10 s, from dawn to dusk. In our experience, one of the most important advantages of this system is the possibility of programming automation scripts via the picamera software package (Jones, 2013) for Python (Python Software Foundation, available at http://www.python.org). This approach gives the user fine-scale control over the quantity, sampling frequency, and spatial resolution of photos and videos. In combination with standard networking protocols like Secure Shell, these features allow for a fully automated pipeline that includes image capture, with file transfer, processing, and data storage when networked to a more powerful host computer.

Another important advantage of these computers is their low cost, especially if the system requires multiple cameras per aviary or across multiple replicas in an experimental setup. Among the disadvantages of this system is the inconsistent quality of the camera modules (a small proportion of our cameras were unable to produce sharp images). To deal with this, we tested the cameras on the experimental setup and avoided using those that seemed to be defective. Although these camera modules provide a large depth of field, they require manual focusing, which can be difficult and is often inconvenient (Table 1).

Figure 1.4. Barcode detections in a social perching context (photographed with the Raspberry Pi camera module). The software can easily detect visible codes in complex aviary environments and extract information about important interactions, such as direct body contact (individuals 55 and 66), that many tracking algorithms would fail to detect

Table 1.1. Summary of the pros and cons of different camera implementations tested

Camera system	**Advantages**	**Disadvantages**
DSLR (Canon EOS 1200)	Image quality. Availability of lenses and accessories. Suitable for video or photo. Low-light performance.	Noisy mechanical shutter (this model). Video limited to 30 min. Bulky. Costly.
Action Cameras (GoPro Hero 4)	Compact size. Image resolution and quality. Suitable for video or photo. Wide angle lens suitable for small spaces.	Lens distortion. Limited battery life. No display (this model). Manual operation (this model).
Single-board computers/Camera module (Raspberry Pi 3 Model B with Camera V2)	Inexpensive. Compact size. Suitable for video or photo. Programmable. Network access. Expected improvements and software updates. No battery life limitation.	Lower image quality. Fixed recording area. Difficult manual focusing.

1.3.4 Extracting data from videos and images

Once videos or images are recorded, the next step is to extract location data from the barcodes contained in the image data. Several software libraries are available to accomplish this (Crall et al., 2015; Garrido-Jurado, Muñoz-Salinas, Madrid-Cuevas, & Marín-Jiménez, 2016; Graving 2017; Wang & Olson, 2016), and each provides its own set of barcodes. These software libraries all extract the identity, location, and orientation of each tag from image data. In our study, we used the software library pinpoint by (Graving 2017), which is a based on the work of Garrido-Jurado et al. (2016).

Code detection

The detection algorithm finds the identity matrix of the barcode using the contrasting white and black edges between the barcode and the black frame of the plastic tray on the backpack. Images are binarized using an adaptive (spatially localized) thresholding algorithm, which allows for uneven lighting, and candidate barcodes are detected based on their geometry, which allows for complex backgrounds. Once a candidate barcode is detected, the identity matrix is extracted from the pixel data and compared to known identities stored in a tag dictionary. The tracking algorithm can reliably detect the codes at arbitrary angles, even when they are not completely perpendicular to the central-axis of the camera lens. The software provides the identity and Cartesian coordinates for the corners of each detected barcode with sub-pixel resolution, which can be used to calculate the orientation of the code (note the importance of fitting the code in the right direction on the birds).

1.3.5 Example data analyses

To briefly demonstrate the use of this automated approach to data collection and analysis, we studied the foraging behaviour of individual zebra finches at a high-quality food source and constructed foraging networks based on high-resolution movement data measured using our system. Social networks are particularly challenging to study using manual observation because they require measuring the behaviour of most or all individuals simultaneously. To achieve this, we created an arena 90 $\times$ 50 cm on the floor of each of the two aviaries and provided birds with an ephemeral high-quality food resource (a slice of zucchini/courgette) twice per day (around 9.00 and 16.00 hr). We used a barcode to record the centroid of the resource, which

was subsequently removed to allow birds unobstructed access to food. Birds were fasted for an hour before the experiment to ensure they were motivated to feed, and their access to the food resource was captured on video using the GoPro Hero 4 camera fitted 50 cm above the food (see above). We collected data on the two aviaries for 58 days, between December 15, 2016 and March 29, 2017.

We extracted feeding association data, representing the propensity for individuals to synchronize their feeding and tolerate one another at the food source. We recorded the identity of individuals detected at the food for every video frame by defining a feeding zone with respect to the centroid of the food resource. A feeding event was recorded when a barcode was detected within a 154-pixel (or c. 8-cm) radius of the resource centroid, and the bird was facing the food (i.e. the centroid was within the 180° zone in front of the bird) (Figure 1.3). Once we identified the individuals in every frame and classified feeding events, we constructed a weighted, undirected social network representing the co-feeding relationships among individuals (represented as nodes) in each flock. We accomplished this by transforming our data into a matrix of pairwise associations using a simple ratio index (SRI) for every pair of individuals in each flock (see Farine & Whitehead, 2015). Here, the edge weight between two individuals (SRI_{ij}) is the probability of observing individuals i and j feeding together given that either i or j has been detected. This calculation is simply given by the following:

$$\mathrm{SRI}_{ij} = \frac{x_{ij}}{n}, \tag{1.1}$$

for $i, j = 1, \ldots, N$ and $i \neq j$, where N is the total number of individuals in the flock, x_{ij} is the number of frames in which individuals i and j were feeding together, and n is the total number of frames where either i or j was detected (alone or together).

1.4 Results

1.4.1 Barcode deployment and maintenance

We deployed backpacks on 58 zebra finches (Figure 1.5), which required about 3 min of handling per individual, plus observation and monitoring time. All the deployed backpacks lasted throughout the experimental period (4 months or more) without causing any injuries to the birds. However, minor maintenance was required as backpacks and codes showed some wearing due

to grooming and allopreening (see backpack-mount in Figure 1.5). Common issues included loss of ink on and around the barcodes, weakened paper around the front holes, and unglued mounts. We also noticed that, in a few cases, the straps lost elasticity after 4 months and appeared loose. More commonly, we observed that debris (i.e. food remains or excrement) on the barcode obstructed its detection. Every time we detected one of these issues, we addressed it immediately to guarantee both safety of the birds and quality and continuity of the data collection. For any minor issues, we carefully cleaned the codes to remove debris, or covered the ink-less spots with black ink permanent markers. For extensive damage on the mount or the surface of the barcode, we removed and replaced the mount keeping the strip and the elastic string on the bird, thus reducing manipulation and acclimation time. In cases that required a whole new backpack, we repeated the process of the first deployment.

Figure 1.5. Male zebra finch with a barcode backpack c. 4 months after fitting. Note the wearing on the backpack structure caused by grooming behaviours

1.4.2 Detection

We recorded 48 hr of video at the feeding arenas using GoPro Cameras and recorded photos using Raspberry Pi cameras. The detection software identified 52.05% of the barcodes (i.e. birds) present in 100 randomly selected frames from the GoPro footage. This percentage was improved to 64.58% after simple linear interpolation of positions across short gaps of missing data (e.g. 1–24 frames of video). From the photos of birds perched on the aviary branch system, the software detected 60.4% of the individuals present in 100 randomly sampled images

captured, using the Raspberry Pi cameras. The most common reasons for failed-detections were motion blur and feathers temporarily obscuring parts of the code (e.g. Figure 1.3). Motion blur was most prominent in GoPro videos but is easily prevented in photos using shutter speeds of 1/1000-s or faster (which can even identify birds in flight, Figure 1.6). Trimming some feathers around the code during backpack deployment can increase the number of detections, although we expect the current rate of data collection to be sufficient for most applications and research questions.

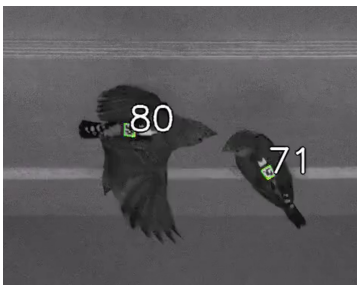

Figure 1.6. Example of a zebra finch (individual 80) being detected in flight (with 1/2000-s shutter speed)

1.4.3 Example data analyses

Using image data collected with a GoPro mounted over the food arena, we were able to distinguish birds consuming the resource from those present in the frame but not feeding (Figure 1.7). For example, from a single 45-min observation period, as shown in Figure 1.7, we recorded 74,960 records of individual positions. These records also contain many potential interactions. We demonstrate that the data on the co-presence of individuals at a food source can be used to generate social networks (Figure 1.8), a powerful approach used in many studies of animal behaviour for which extensive observation data are required.

1.5 Discussion

We present a method for recording the behaviour of captive birds using backpack-mounted barcodes, image capture, and computer detection. With proper deployment, manipulation and monitoring, we have shown that this system is safe for the birds, durable, and capable of delivering extensive data from multiple individuals simultaneously. These position and orientation data can be used to assess multiple types of behaviours, associations and interactions among

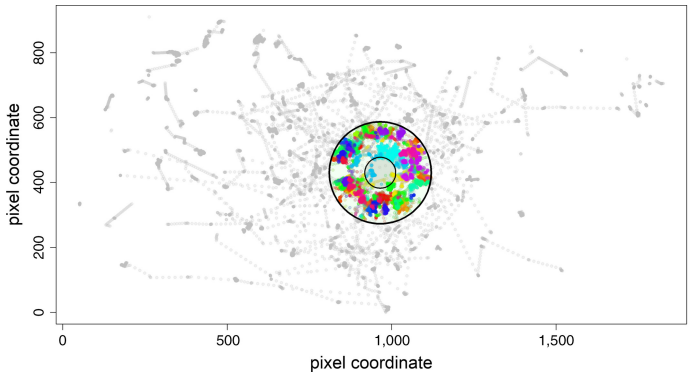

Figure 1.7. Detection of feeding behaviour characterized by proximity and directionality to the food source. The inner circle represents the outline of the food source, and the outer circle represents the 154-pixel boundary for birds to considered to be "at food." Coloured dots represent detections of different individuals within the "at food" zone. Grey dots are birds present and identified in the frame but not actively feeding. The positions of birds away from the centre of the frame are less accurate due to lens distortion (see Figure 1.3)

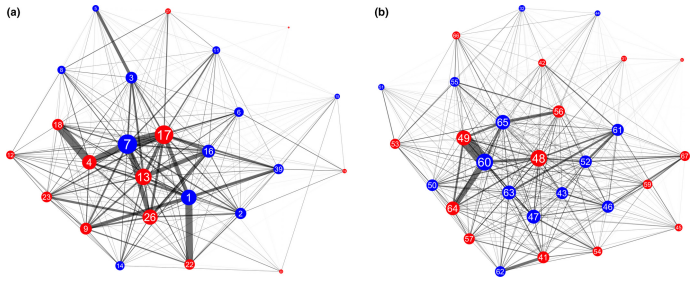

Figure 1.8. Affiliative networks generated with co-feeding data extracted from barcodes detected at a food source using a video camera in (a) flock 1 and (b) flock 2. Each node (circle) represents an individual. Red nodes represent males, and blue nodes represent females. The size of the node represents the individual's degree in the network, a measure of centrality computed by summing the weights of all the edges connected to it. The thickness of the line represents the strength of the association between each pair of individuals

individuals. This system presents several advantages to more commonly-implemented methods. In particular, it is adaptable to different contexts and research questions, being possible to vary the temporal resolution (photos or video) and the area covered without requiring any additional markers to birds. For general purposes, the use of Raspberry Pi single-board computers and camera modules makes this method affordable, enabling high-throughput data collection that increases sample sizes and statistical power. Our example analyses demonstrate that the barcode-based approach can generate similar data to what is often collected using PIT tags (Figure 1.8), but also provides much richer information on movements and spatial location within patches (Figure 1.7). We found that the backpack system simplified the data analysis because we were certain about the co-occurrence of birds at the same food source (i.e. captured in the

same frame), instead of having to infer co-occurrences from sequences of detections using pattern-recognition algorithms (e.g. Psorakis et al., 2015).

We tested the application of different camera setups and behavioural contexts, including video for feeding arenas and photos in co-perching scenarios. Cameras could be fitted in various locations, including bathing areas, nest boxes, and potentially in open areas to capture birds in flight (e.g. Figure 1.6). The decision on the type of camera and on video or photos will depend on each research question. For example, researchers could choose video for recording aggressive interactions or other behaviours that involve movement, or capture photos every few seconds to capture affiliative data for the purposes of studying social networks, pair formation, or group stability. The type of data provided by these barcodes also provides new opportunities for analysis. Using machine learning, it will be possible to automatically classify behaviours and interactions over extended periods of time while also minimizing manual annotation by a human observer (Robie, Seagraves, Egnor, & Branson, 2017), thereby avoiding bias and fatigue. Such approaches have been developed for studying other organisms (Kabra, Robie, Rivera-Alba, Branson, & Branson, 2013), which use data that are similar to what our system generates.

Our backpack-based barcode method has potential to be extended to diverse range of systems. Although we only collected data during daylight hours, barcodes could easily be detected in low-light conditions and many commercially available infrared cameras can image the black-and-white codes without visible light (using infrared lights). While most birds are not very active at night, there is increasing evidence that many important behaviours happen early in the morning (Bonter, Zuckerberg, Sedgwick, & Hochachka, 2013). Such behaviours could easily be captured with this barcode system but would be almost impossible to study using manual observations or video as it is difficult to identify coloured leg bands. Future applications include using barcodes to identify individuals interacting with a device (e.g. a feeder or a puzzle box). To date, such systems have mostly relied on using PIT tags (e.g. Aplin et al., 2015), which limits sampling to a single individual at once. In social species, individuals often congregate, and a barcode system can facilitate multiple simultaneous detections and quantify relative positions of individuals to one-another and to the device. The implementation of “real-time” detection could allow for algorithms that control devices in response to the behaviour of birds, such as allowing only a maximum number of individuals in one area or selectively dispensing food to particular individuals (as performed by Firth, Voelkl, Farine, & Sheldon, 2015). Barcodes could provide a powerful interface between individuals and experimental devices, not only by being able to

provide tailored responses (such as individual learning algorithms, Morand-Ferron, Hamblin, Cole, Aplin, & Quinn, 2015), but also, unlike almost any other system, by capturing information about who else is present when particular events occur.

Although we have discussed the multiple advantages, the limitations of the system must be also considered. While backpacks and barcodes can last for more than 4 months, permanent monitoring was required to assure safety of the birds and adequate delivery of data. Grooming and allopreening caused some wear on the backpacks and codes, and this sometimes led to impaired movement of the birds. Detecting and addressing such issues is important for both safety of the birds and continuity of the data collection. Additionally, there are unavoidable issues that reduce detectability, like fast movement, codes tilted due to extreme body position, and wings or feathers partially covering the trays. The current design of the backpacks addresses these issues well and delivers consistent detection. Additional concerns related to camera systems, such as storage, resolution, lens distortion or lighting, can be solved for specific research circumstances.

A key question that requires further investigation is whether these backpacks will be suitable for field deployment. We found that, in zebra finches, we could detect most issues within the first 1–2 days. However, few field studies are amenable to keeping birds in captivity to allow such monitoring. Thus, field applications may be limited to species that either have well-established protocols for fitting backpacks in the field or those in which individuals can be easily monitored (e.g. territorial species). We believe that there is a danger that small songbirds could entangle their backpacks in small branches, particularly if backpacks become loose over time. Finally, our aviaries had artificial lighting that remained constant during daytime. Researchers conducting outdoor studies, with natural lighting conditions, must consider the changing environment (i.e. sun position and cloud coverage) to avoid unusable images due to the differences in light quality from dawn/dusk to noon. For example, sun shining directly on the white tag will make the code invisible to the camera, while a setup designed for sunny conditions would create completely black photos under cloudy conditions. The use of infra-red cameras and infra-red lighting is one way to overcome this challenge.

Our backpack-mounted barcode system could revolutionize data collection in a range of experimental systems. We have demonstrated that it can be implemented safely and cheaply. Further, it has the ability to collect extensive data across many individuals simultaneously and the flexibility to address diverse research questions. With simple software modifications, the

system can also be integrated into active devices that interface directly with individuals, which will prove to be an extremely powerful experimental approach.

1.6 Data availability

The raw tracking data files, including the distance from the food patch, that are used for the example data analyses can be found at https://doi.org/10.17617/3.19.

Chapter 2

DeepPoseKit: a software toolkit for fast and robust animal pose estimation using deep learning

Jacob M. Graving, Daniel Chae, Hemal Naik, Liang Li, Benjamin Koger, Blair R. Costelloe, and Iain D. Couzin

Adapted from: Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., Couzin, I. D. (2019). DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. eLife, 8, e47994

2.1 Abstract

Quantitative behavioral measurements are important for answering questions across scientific disciplines—from neuroscience to ecology. State-of-the-art deep-learning 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 and robustness. Here we introduce a new easy-to-use software toolkit, *DeepPoseKit*, that addresses these problems using an efficient multi-scale deep-learning model, called *Stacked DenseNet*, and a fast GPU-based peak-detection algorithm for estimating keypoint locations with subpixel precision. These advances improve processing speed $>2\times$ with no loss in accuracy compared to currently-available methods. We demonstrate the versatility of our methods 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.

2.2 Introduction

Understanding the relationships between individual behavior, brain activity (reviewed by Krakauer et al. 2017), and collective and social behaviors (J. W. Jolles et al., 2017; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; Rosenthal et al., 2015; Strandburg-Peshkin et al., 2013) is a central goal of the behavioral sciences—a field that spans disciplines from neuroscience to psychology, ecology, and genetics. Measuring and modeling 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 & Perona, 2014; Berman, 2018; Brown & 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., 2016, 2014a; Costa et al., 2019; M. Johnson et al., 2016; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; Markowitz et al., 2018; Stephens et al., 2011; Todd et al., 2017; Wiltschko et al., 2015). Tools for automatically measuring animal movement represent a vital first step toward developing unified theories of behavior across

scales (Berman, 2018; Brown & 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 latest contributions to the growing toolbox for quantitative behavioral analysis are from A. Mathis et al. (2018) and Pereira et al. (2019), who make use of a popular type of machine learning model called *convolutional neural networks*, or *CNNs* (LeCun et al. 2015; Appendix A.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 and robustness, which may limit their practical applications. To address these problems, we introduce a new software toolkit, called *DeepPoseKit*, with methods that are fast, robust, and easy-to-use. We run experiments using multiple datasets to compare our new methods with those from A. Mathis et al. (2018) and Pereira et al. (2019), and we find that our approach offers considerable improvements. These results also demonstrate the flexibility of our toolkit for both laboratory and field situations and exemplify the wide applicability of our methods across a range of species and experimental conditions.

2.2.1 Measuring animal movement with computer vision

Collecting high-quality behavioral data is a challenging task, and while direct observations are important for gathering qualitative data about a study system, a variety of automated methods for quantifying movement have become popular in recent years (Anderson & Perona, 2014; Dell et al., 2014; Kays et al., 2015). Methods like video monitoring and recording help to accelerate data collection and reduce the effects of human intervention, but the task of manually scoring videos is time consuming and suffers from the same limitations as direct observation, namely observer bias and mental fatigue. Additionally, due to limitations of human observers' ability to process information, many studies that rely on manual scoring use relatively small datasets to estimate experimental effects, which can lead to increased rates of statistical errors. Studies that lack the statistical resolution to robustly test hypotheses (commonly called "power" in frequentist statistics) also raise concerns about the use of animals for research, as statistical errors caused by sparse data can impact researchers' ability to accurately answer scientific questions. These limitations have led to the development of automated methods for quantifying behavior using advanced imaging technologies (Dell et al., 2014) as well as sophisticated tags and collars

with GPS, accelerometry, and acoustic-recording capabilities (Kays et al., 2015). Tools for automatically measuring the behavior of individuals now play a central role in our ability to study the neurobiology and ecology of animals, and reliance on these technologies for studying animal behavior will only increase in the future.

The rapid development of computer vision hardware and software in recent years has allowed for the use of automated image-based methods for measuring behavior across many experimental contexts (Dell et al., 2014). Early methods for quantifying movement with these techniques required highly-controlled laboratory conditions. However, because animals exhibit different behaviors depending on their surroundings (Akhund-Zade et al., 2019; Francisco et al., 2020; Strandburg-Peshkin et al., 2017), laboratory environments are often less than ideal for studying many natural behaviors. Most conventional computer vision methods are also limited in their ability to accurately track groups of individuals over time, but nearly all animals are social at some point in their life and exhibit specialized behaviors when in the presence of conspecifics (Francisco et al., 2020; J. W. Jolles et al., 2017; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; Rosenthal et al., 2015; Strandburg-Peshkin et al., 2013; Versace et al., 2019). These methods also commonly track only the animal's center of mass, which reduces the behavioral output of an individual to a two-dimensional or three-dimensional particle-like trajectory. While trajectory data are useful for many experimental designs, the behavioral repertoire of an animal cannot be fully described by its aggregate locomotory output. For example, stationary behaviors, like grooming and antennae movements, or subtle differences in walking gaits cannot be reliably detected by simply tracking an animal's center of mass (Berman et al., 2014a; Wiltschko et al., 2015).

Together these factors have driven the development of software that can accurately track the positions of marked (Boenisch et al., 2018; Crall et al., 2015; Graving, 2017; Wild et al., 2018) or unmarked (Pérez-Escudero et al., 2014; Romero-Ferrero et al., 2019) individuals as well as methods that can quantify detailed descriptions of an animal's posture over time (Berman et al., 2014a; A. Mathis et al., 2018; Pereira et al., 2019; Stephens et al., 2011; Wiltschko et al., 2015). Recently these advancements have been further improved through the use of deep learning, a class of machine learning algorithms that learn complex statistical relationships from data (LeCun et al., 2015). Deep learning has opened the door to accurately tracking large groups of marked (Boenisch et al., 2018; Wild et al., 2018) or unmarked (Romero-Ferrero et al., 2019) individuals and has made it possible to measure the body posture of animals in nearly any context—including in the wild (Nath et al., 2019a)—by tracking the positions of user-defined body

parts (A. Mathis et al., 2018; Pereira et al., 2019). These advances have drastically increased the quality and quantity, as well as the diversity, of behavioral data that are potentially available to researchers for answering scientific questions.

2.2.2 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 (J. Kain et al., 2013; Mendes et al., 2013), attaching intrusive physical markers to the study animal (J. 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 (J. 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 near human-level accuracy in almost any scenario by manually annotating data (Figure 2.1)—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 2.3). State-of-the-art machine learning methods, like CNNs, use these training data to parameterize a model describing the statistical relationships 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 A.1–A.5.

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 labor-intensive task made less time-consuming by techniques described in Appendix A.2.

A. 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 *fully-convolutional neural network* or *F-CNN* (Long et al. 2015; Appendix A.3) often referred to as an *encoder-decoder* model (Appendix A.3 Box A.4). 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 2.3), 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.

While deep-learning models typically need large amounts of training data, both A. Mathis et al. (2018) and Pereira et al. (2019) have demonstrated that near human-level accuracy can be achieved with few training examples (Appendix A.2). In order to ensure generalization to large datasets, both groups of researchers introduced ideas related to iteratively refining the training set used for model fitting (A. Mathis et al., 2018; Pereira et al., 2019). In particular, Pereira et al. (2019) describe a technique known as *active learning* where a trained model is used to initialize new training data and reduce annotation time (Appendix A.2). A. Mathis et al. (2018) describe multiple techniques that can be used to further refine training data and minimize errors when making predictions on the full dataset. Simple methods to accomplish this include filtering data or selecting new training examples based on confidence scores or the entropy of the confidence maps from the model output. Nath et al. (2019a) also introduced the use temporal derivatives (i.e., speed and acceleration) and autoregressive models to identify outlier frames, which can then be labeled to refine the training set or excluded from further analysis on the final dataset (Figure 2.1).

2.2.3 Pose estimation models and the speed-accuracy trade-off

A. Mathis et al. (2018) developed their pose estimation model, which they call *DeepLabCut*,

by modifying a previously-published model called *DeeperCut* (Insafutdinov et al., 2016). The DeepLabCut model (A. Mathis et al., 2018), like the DeeperCut model, is built on the popular *ResNet* architecture (He et al., 2016)—a state-of-the-art deep-learning model used for image classification. This choice is advantageous because the use of a popular architecture allows for incorporating a pre-trained encoder to improve performance and reduce the number of required training examples (A. Mathis et al., 2018), known as *transfer learning* (Pratt 1993; Appendix A.2)—although, as will be seen, transfer learning appears to offer little improvement over a randomly-initialized model. However, this choice of of a pre-trained architecture is also disadvantageous as the model is *overparameterized* with >25 million parameters. Overparameterization allows the model to make accurate predictions, but this may come with the cost of slow inference. To alleviate these effects, work from A. Mathis & Warren (2018) showed that inference speed for the DeepLabCut model (A. Mathis et al., 2018) can be improved by decreasing the resolution of input images, but this is achieved at the expense of accuracy.

With regard to model design, Pereira et al. (2019) implement a modified version of a model called *SegNet* (Badrinarayanan et al., 2015), which they call *LEAP* (LEAP Estimates Animal Pose), that attempts to limit model complexity and overparameterization with the goal of maximizing inference speed (see Appendix A.5)—however, our comparisons in this paper suggest Pereira et al. (2019) achieved only limited success compared to the DeepLabCut model (A. Mathis et al., 2018). The LEAP model 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 training routine for the LEAP model introduced by Pereira et al. (2019) requires computationally expensive preprocessing that is not practical for many datasets, which makes it unsuitable for a wide range of experiments (see Appendix A.5 for more details).

Together the methods from A. Mathis et al. (2018) and Pereira et al. (2019) represent the two extremes of a phenomenon known as the *speed-accuracy trade-off* (J. Huang et al., 2017)—an active area of research in the machine learning literature. A. Mathis et al. (2018) prioritize accuracy over speed by using a large overparameterized model, 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 accuracy (e.g., Chollet 2017; G. Huang et al. 2017; 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 A.3 Box A.4) by increasing connectivity between the model's many layers across multiple spatial scales (Appendix A.3 Figure A.6). Multi-scale inference implicitly allows the model to simultaneously integrate large-scale global information, such as the lighting, image background, or the orientation of the focal individual's body trunk; information from intermediate scales like anatomical geometry related to cephalization and bilateral symmetry; and fine-scale local information that could include differences in color, texture, or skin patterning for specific body parts. This multi-scale design gives the model capacity to learn the hierarchical relationships between different spatial scales and efficiently aggregate them into a joint representation when solving the posture estimation task (see Box A.4 and Appendix A.3 Figure A.6 for further discussion)

2.2.4 Individual vs. multiple pose estimation

Most work on human pose estimation now focuses on estimating the pose of multiple individuals in an image (e.g., Cao et al. 2017). For animal pose estimation, the methods from Pereira et al. (2019) are limited to estimating posture for single individuals—known as *individual pose estimation*—while the methods from A. Mathis et al. (2018) can also be extended to estimate posture for multiple individuals simultaneously—known as *multiple pose estimation*. However, the majority of work on multiple pose estimation, including A. Mathis et al. (2018), has not adequately solved the tracking problem of linking individual posture data across frames in a video, especially after visual occlusions, which are common in many behavioral experiments—although recent work has attempted to address this problem (Andriluka et al., 2018; Iqbal et al., 2017). Additionally, as the name suggests, the task of multiple pose estimation requires exhaustively annotating images of multiple individuals—where every individual in the image must be annotated to prevent the model from learning conflicting information. This type of annotation task is even more laborious and time consuming than annotations for individual pose estimation and the amount of labor increases proportionally with the number of individuals in each frame, which makes this approach intractable for many experimental systems.

Reliably tracking the position of individuals over time is important for most behavioral studies, and there are a number of diverse methods already available for solving this problem (Boenisch

et al., 2018; Crall et al., 2015; Graving, 2017; Pérez-Escudero et al., 2014; Romero-Ferrero et al., 2019; Wild et al., 2018). Therefore, to avoid solving an already-solved problem of tracking individuals and to circumvent the cognitively complex task of annotating data for multiple pose estimation, 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 cropped from a larger multi-individual image after localization and tracking. We introduce a top-down posture estimation framework that can be readily adapted to existing behavioral analysis workflows, which could include any method for localizing and tracking individuals.

The additional step of localizing and tracking individuals naturally increases the processing time for producing posture data from raw image data, which varies depending on the algorithms being used and the number of individuals in each frame. While tracking and localization may not be practical for all experimental systems, which could make our methods difficult to apply "out-of-the-box", the increased processing time from automated tracking algorithms is a reasonable trade-off for most systems given the costly alternative of increased manual labor when annotating data. This trade-off seems especially practical when considering that the posture data produced by most multiple pose estimation algorithms still need to be linked across video frames to maintain the identity of each individual, which is effectively a bottom-up method for achieving the same result. Limiting our methods to individual pose estimation also simplifies the pose detection problem as processing confidence maps produced by the model does not require computationally-expensive local peak detection and complex methods for grouping keypoints into individual posture graphs (e.g., Cao et al. 2017; Insafutdinov et al. 2016; Appendix A.3). Additionally, because individual pose estimation is such a well-studied problem in computer vision, we can readily build on state-of-the-art methods for this task (see Appendices A.3 and A.4 for details).

2.3 Results

Here we introduce fast, flexible, and robust pose estimation methods, with a software interface—a high-level programming interface (API) and graphical user-interface (GUI) for annotations—that emphasizes usability. Our methods build on the state-of-the-art for individual pose estimation (Newell et al. 2016; Appendix A.4), convolutional regression models (Jégou et al. 2017; Appendix A.3 Box A.4), and conventional computer vision algorithms (Guizar-Sicairos et al., 2008) to

improve 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 peak detection for estimating keypoint locations with subpixel precision—even at low spatial resolutions. We also discuss a modification to incorporate a hierarchical posture graph for learning the multi-scale geometry between keypoints on the animal's body, which increases accuracy when training pose estimation models. We ran experiments to optimize our approach and compared our new models to the models from A. Mathis et al. (2018) (DeepLabCut) and Pereira et al. (2019) (LEAP) in terms of speed, accuracy, training time, and generalization ability. We benchmarked these models using three image datasets recorded in the laboratory and the field—including multiple interacting individuals that were first localized and cropped from larger, multi-individual images (see "Methods" for details).

2.3.1 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 2.1) with a custom GUI for creating annotated training data with active learning similar to Pereira et al. (2019; Appendix A.2), as well as a flexible pipeline for data augmentation (Jung 2018; Appendix A.2; shown in Figure 2.3), model training and evaluation (Figure 2.3; Appendix A.1), and running inference on new data. We designed our high-level programming interface using the same guidelines from Keras (Chollet et al., 2015) to allow the user to go from idea to result 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.

2.3.2 Our pose estimation models

To achieve the goal of "fast animal pose estimation" introduced by Pereira et al. (2019), while maintaining the robust predictive power of models like DeepLabCut (A. Mathis et al., 2018), we implemented two fast pose estimation models that extend the state-of-the-art model 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 the DeepLabCut model (A. Mathis et al., 2018) and LEAP model (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 A.3 Box A.4) while optimizing our model hyperparameters for efficiency. Our first model is a novel implementation of *FC-DenseNet* from Jégou et al. (2017; Appendix A.3 Box A.4) arranged in a stacked configuration similar to Newell et al. (2016; Appendix A.4). We call this new model Stacked DenseNet, and to the best of our knowledge, this is the first implementation of this model architecture in the literature—for pose estimation or otherwise. Further details for this model are available in Appendix A.7. Our second model is a modified version of the *Stacked Hourglass* model from Newell et al. (2016; Appendix A.4) 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).

2.3.3 Subpixel keypoint prediction on the GPU allows for fast and accurate inference

In addition to implementing our efficient pose estimation models, we developed a new method to process 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 (Insafutdinov et al., 2016; A. Mathis et al., 2018). However, CPU processing is not ideal because moving large arrays of data between the GPU and CPU can be 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 (Pereira et al., 2019). 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 of keypoint locations even if the model's confidence maps are dramatically smaller than the resolution of the input image. We compared the accuracy of our subpixel maxima layer to an integer-based maxima layer using the fly dataset from Pereira et al. (2019) (see "Methods"). We found significant accuracy improvements across every downsampling configuration (Appendix A Figure A.1a). 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 for confidence maps at such a downsampled resolution allows us to achieve very fast inference >1000 Hz while maintaining high accuracy (Appendix A Figure A.1b).

We also provide speed comparisons with the other models we tested and find that our Stacked DenseNet model with our subpixel peak detection algorithm is faster than the DeepLabCut model (A. Mathis et al., 2018) for both offline (batch size = 100) and real-time speeds (batch size = 1). While we find that our Stacked DenseNet model is faster than the LEAP model (Pereira et al., 2019) for offline processing (batch size = 100), the LEAP model (Pereira et al., 2019) is significantly faster for real-time processing (batch size = 1). Our Stacked Hourglass model (Newell et al., 2016) is about the same or slightly faster than Stacked DenseNet for offline speeds (batch size = 100), but is much slower for real-time processing (batch size = 1). Achieving fast pose estimation using CNNs typically relies on massively parallel processing on the GPU with large batches of data or requires downsampling the images to increase speed, which increases error (A. Mathis & Warren, 2018). These factors make fast and accurate real-time inference challenging to accomplish. Our Stacked DenseNet model, with a batch size of one, can run inference at $\sim$30-110Hz—depending on the resolution of the predicted confidence maps (Appendix A Figure A.1b). These speeds are faster than the DeepLabCut model (A. Mathis et al., 2018) and could be further improved by downsampling the input image resolution or reconfiguring the model with fewer parameters. This allows our methods to be flexibly used for real-time or closed-loop behavioral experiments with prediction errors similar to current state-of-the-art methods.

2.3.4 Learning multi-scale geometry between keypoints improves accuracy and reduces extreme errors

Minimizing extreme prediction errors is important to prevent downstream effects on any further behavioral analysis (Seethapathi et al., 2019)—especially in the case of analyses based on time-frequency transforms like those from Berman et al. (2016, 2014a); Klibaite et al. (2017); Klibaite & Shaevitz (2019); Todd et al. (2017) and Pereira et al. (2019) where high magnitude errors can cause inaccurate behavioral classifications. While effects of these extreme errors can be minimized using post-hoc filters and smoothing, these post-processing techniques can remove relevant high-frequency information from time-series data, so this solution is less than ideal. One way to minimize extreme errors when estimating posture is to incorporate multiple spatial scales when making predictions (e.g., Y. Chen et al. 2017). Our pose estimation models are implicitly capable of using information from multiple scales (see Appendix A.3 Box A.4), 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 keypoint locations, a hierarchical graph of edges describing the multi-scale 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 2.3), (2) a set of confidence maps for individual limbs (e.g., left leg, right arm, etc.; Figure 2.6), (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 2.3). 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.

We find that training our Stacked DenseNet model to predict a hierarchical posture graph reduces keypoint prediction error (Appendix A Figure A.2), 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 multi-scale 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. Predicting the posture graph may be especially useful for animals with long slender appendages such as insect legs and antennae where prediction errors are likely to occur due to occlusions and natural variation in the movement of these body parts. These results also suggest that annotating multiple keypoints to incorporate an explicit signal for multi-scale information may help improve prediction accuracy for a specific body part of interest.

2.3.5 Stacked DenseNet is fast and robust

We benchmarked our new model implementations against the models (Pereira et al., 2019) and A. Mathis et al. (2018). We find that our Stacked DenseNet model outperforms both the LEAP model (Pereira et al., 2019) and the DeepLabCut model (A. Mathis et al., 2018) in terms of speed while also achieving much higher accuracy than the LEAP model (Pereira et al., 2019) with similar accuracy to the DeepLabCut model (A. Mathis et al. 2018; Figure 2.4a). We found that both the Stacked Hourglass and Stacked DenseNet models outperformed the LEAP model (Pereira et al., 2019). Notably our Stacked DenseNet model achieved approximately $2\times$ faster inference speeds with $3\times$ 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 model consistently achieved prediction accuracy nearly identical to the DeepLabCut model (A. Mathis et al., 2018) while running inference at nearly $2\times$ the speed and using only $\sim 5\%$ of the parameters—$\sim$1.5 million vs. $\sim$26 million. Detailed results of our model comparisons are shown in Figure 2.4-Figure supplement 2.5.

While the Stacked DenseNet model used for comparisons is already fast, inference speed could be further improved by using a $\frac{1}{8}\times$ output without much increase in error (Appendix A Figure A.1) or by further adjusting the hyperparameters to constrain the size of the model. Our Stacked Hourglass implementation followed closely behind the performance of our Stacked DenseNet model and the DeepLabCut model (A. Mathis et al., 2018) but consistently performed more poorly than our Stacked DenseNet model in terms of prediction accuracy, so we excluded this model from further analysis. We were also able to reproduce the results reported by Pereira et al. (2019) that the LEAP model and the Stacked Hourglass model (Newell et al., 2016) have

similar average prediction error for the fly dataset. However, we also find that the LEAP model (Pereira et al., 2019) has much higher variance, which suggests it is more prone to extreme prediction errors—a problem for further data analysis.

2.3.6 Stacked DenseNet trains quickly and requires few training examples

To further compare models, we used our zebra dataset to assess the training time needed for our Stacked DenseNet model, the DeepLabCut model (A. Mathis et al., 2018), and the LEAP model (Pereira et al., 2019) to reach convergence as well as the amount of training data needed for each model to generalize to new data from outside the training set. We find that our Stacked DenseNet model, the DeepLabCut model (A. Mathis et al., 2018), and the LEAP model (Pereira et al., 2019) all fully converge in just a few hours and reach reasonably high accuracy after only an hour of training (Appendix A Figure A.3). However, it appears that our Stacked DenseNet model tends to converge to a good minimum faster than both the DeepLabCut model (A. Mathis et al., 2018) and the LEAP model (Pereira et al., 2019).

We also show that our Stacked DenseNet model achieves good generalization with few training examples and without the use of transfer learning (Appendix A Figure A.4). These results demonstrate that, when combined with data augmentation, as few as five training examples can be used as an initial training set for labelling keypoints with active learning (Figure 2.1). Additionally, because our analysis shows that generalization to new data plateaus after approximately 100 labeled training examples, it appears that 100 training examples is a reasonable minimum size for a training set—although the exact number will likely change depending the variance of the image data being annotated. To further examine the effect of transfer learning on model generalization, we compared performance between the DeepLabCut model (A. Mathis et al., 2018) initialized with weights pretrained on the ImageNet database (Deng et al., 2009) vs. the same model with randomly-initialized weights (Appendix A Figure A.4). As postulated by A. Mathis et al. (2018), we find that transfer learning does provide some benefit to the DeepLabCut model's ability to generalize. However, the effect size of this improvement is small with a mean reduction in Euclidean error of <0.5 pixel. Together these results indicate that transfer learning is helpful, but not required, for deep learning models to achieve good generalization with limited training data.

2.4 Discussion

Here we have presented a new software toolkit, called DeepPoseKit, 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 reducing accuracy or generalization ability. Our new pose estimation model, called Stacked DenseNet, offers considerable improvements (Figure 2.4a; Figure supplement 2.5) over the models from A. Mathis et al. (2018) (DeepLabCut) and Pereira et al. (2019) (LEAP), and our software framework also provides a simplified interface (Figure 2.4b) 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 and require few training examples to achieve good predictive performance on new data—without the use of transfer learning. We ran experiments to optimize our approach and discovered that some straightforward modifications can greatly improve speed and accuracy. 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.

While our results offer a good-faith comparison of the available methods for animal pose estimation, there is inherent uncertainty that we have attempted to account for but may still bias our conclusions. For example, deep learning models are trained using stochastic optimization algorithms that give different results with each replicate, and the Bayesian statistical methods we use for comparison are explicitly probabilistic in nature. There is also great variability across hardware and software configurations when using these models in practice (A. Mathis & Warren, 2018), so performance may change across experimental setups and datasets. Additionally, we demonstrated that some models may perform better than others for specific applications (Figure 2.4 supplement 2.5), and to account for this, our toolkit offers researchers the ability to choose the model that best suits their requirements—including the LEAP model (Pereira et al., 2019) and the DeepLabCut model (A. Mathis et al., 2018).

We highlighted important considerations when using CNNs for pose estimation and reviewed the progress of fully-convolutional regression models from the literature. The latest 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). Future progress for this class of models may require better loss functions (Y. Chen et al., 2017; Goodfellow et al., 2014; J. Johnson et al., 2016; R. Zhang et al., 2018) that more explicitly model the spatial dependencies within a scene, models that incorporate the temporal structure of the data (Seethapathi et al., 2019), and more mathematically-principled approaches (e.g., Roy et al. 2018; Weigert et al. 2018) such as the application of formal probabilistic concepts (Kendall & Gal, 2017) and Bayesian inference at scale (Tran et al., 2018).

Measuring behavior is a 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 & Perona, 2014; Berman, 2018; Brown & De Bivort, 2018), which our methods and other deep-learning–based tools provide (A. Mathis et al., 2018; Pereira et al., 2019). We have also demonstrated the possibility that our toolkit could be used for 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 (Ayroles et al., 2015; Brown et al., 2013; J. S. Kain et al., 2012) and environmental (Akhund-Zade et al., 2019; Bierbach et al., 2017; Versace 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 (Doudna & Charpentier, 2014; Ran et al., 2013), high-throughput behavioral experiments (Alisch et al., 2018; Javer et al., 2018; Werkhoven, Rohrsen, et al., 2019), and behavioral analysis (e.g., Berman et al. 2014a; 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 (e.g., Boenisch et al. 2018; Crall et al. 2015; Graving 2017; Pérez-Escudero et al. 2014; Romero-Ferrero et al. 2019; Wild et al. 2018), 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 (Rosenthal et al., 2015; Strandburg-Peshkin et al., 2015, 2017, 2013). 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 (Rosenthal et al., 2015; Strandburg-Peshkin et al., 2013). 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 groups by revealing the networks of behavioral contagion across multiple timescales and sensory modalities. While we have provided a straightforward solution for applying existing pose estimation methods to measure collective behavior, there still remain many challenging scenarios where these methods would fail. For example, tracking posture in a densely-packed bee hive or school of fish would require novel solutions to deal with the 3-D nature of individual movement, which includes maintaining individual identities and dealing with the resulting occlusions that go along with imaging these types of biological systems.

When combined with unmanned aerial vehicles (UAVs; Schiffman 2014) or other field-based imaging (Francisco et al., 2020), 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 (Flack et al., 2018; Kays et al., 2015; Nagy et al., 2010; Nagy, Vásárhelyi, et al., 2013; Strandburg-Peshkin et al., 2015, 2017)—especially for species that are impractical to study with tags or collars. Additionally, by applying these methods in conjunction with 3-D habitat reconstruction—using techniques from photogrammetry (Francisco et al., 2020; Strandburg-Peshkin et al., 2017)—field-based studies can begin to integrate fine-scale behavioral measurements with the full 3-D environment in which the behavior evolved. Future advances will likely allow for the calibration and synchronizaton of imaging devices across multiple UAVs (e.g. Price et al. 2018; Saini et al. 2019). This would make it possible to measure the full 3-D posture of wild animals (e.g. Zuffi et al. 2019) in scenarios where fixed camera systems (e.g. Nath et al. 2019a) would not be tractable, such as during migratory

or predation events. When combined, these technologies could allow researchers to address questions about the behavioral ecology of animals that were previously impossible to answer.

Computer vision algorithms for measuring behavior at the scale of posture have rapidly advanced in a very short time; nevertheless, the task of pose estimation is far from solved. There are hard limitations to this current generation of pose estimation methods that are primarily related to the requirement for human annotations and user-defined keypoints—both in terms of the number of keypoints, the specific body parts being tracked, and the inherent difficulty of incorporating temporal information into the annotation and training procedure. Often the body parts chosen for annotation are an obvious fit for the experimental design and have reliably-visible reference points on the animal's body that make them easy to annotate. However, in many cases the required number and type of body parts needed for data analysis may not so obvious—such as in the case of unsupervised behavior classification methods (Berman et al., 2014a; Pereira et al., 2019). Additionally, the reference points for labeling images with keypoints can be hard to define and consistently annotate across images, which is often the case for soft or flexible-bodied animals like worms and fish. Moreover, due to the laborious nature of annotating keypoints, the current generation of methods also rarely takes into account the natural temporal structure of the data, instead treating each video frame as a statistically independent event, which can lead to extreme prediction errors (reviewed by Seethapathi et al. 2019). Extending these methods to track the full three-dimensional posture of animals also typically requires the use of multiple synchronized cameras (Günel et al., 2019; Nath et al., 2019a), which increases the cost and complexity of creating an experimental setup, as well as the manual labor required for annotating a training set, which must include labeled data from every camera view.

These limitations make it clear that fundamentally-different methods may be required to move the field forward. Future pose estimation methods will likely replace human annotations with fully-articulated volumetric 3-D models of the animal's body (e.g., Zuffi et al. 2019, 2017), and the 3-D scene will be learned in an unsupervised or weakly-supervised way (e.g., Jaques et al. 2019; Zuffi et al. 2019), where the shape, position, and posture of the animal's body, the camera position and lens parameters, and the background environment and lighting conditions will all be jointly learned directly from 2-D images by a deep-learning model (Valentin et al., 2019; Zuffi et al., 2019). These *inverse graphics models* (Kulkarni et al., 2015; Sabour et al., 2017; Valentin et al., 2019) will likely take advantage of recently-developed differentiable graphics engines that allow 3-D rendering parameters to be straightforwardly controlled using computationally-efficient

optimization methods (Valentin et al., 2019; Zuffi et al., 2019). After optimization, the volumetric 3-D timeseries data predicted by the deep learning model could be used directly for behavioral analysis or specific keypoints or body parts could be selected for analysis post-hoc. In order to more explicitly incorporate the natural statistical properties of the data, these models will also likely rely on the use of perceptual (J. Johnson et al., 2016; R. Zhang et al., 2018; Zuffi et al., 2019) and adversarial (Y. Chen et al., 2017; Goodfellow et al., 2014; Kanazawa et al., 2019) loss functions that incorporate spatial dependencies within the scene rather than modeling each video frame as a set of statistically independent pixel distributions—as is the case with current methods that use factorized likelihood functions such as pixel-wise mean squared error (e.g., Pereira et al. 2019) or cross-entropy loss (e.g., A. Mathis et al. 2018). Because there would be limited or no requirement for human-provided labels, these models could also be easily modified to incorporate the temporal structure of the data using autoregressive representations (e.g., Kanazawa et al. 2019; Kumar et al. 2019; Oord et al. 2016; Van den Oord et al. 2016), rather than modeling the scene in each video frame as a statistically independent event. Together these advances could lead to larger, higher-resolution, more reliable behavioral datasets that could revolutionize our understanding of relationships between behavior, the brain, and the environment.

In conclusion, we have presented a new toolkit, called DeepPoseKit, for automatically measuring animal posture from images. We combined recent advances from the literature to create methods that are fast, robust, and widely applicable to a range of species and experimental conditions. When designing our framework we emphasized usability across the 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.

2.5 Methods

We ran three main 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 $1\times$ to $\frac{1}{16}\times$ the input resolution using our Stacked DenseNet model. Next, we tested if training our Stacked DenseNet model to predict the multi-scale 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) (LEAP) and A. Mathis et al. (2018) (DeepLabCut), which we also implemented in our framework (see Appendix A.7 for details). We assessed both the inference speed and prediction accuracy of each model as well as training time and generalization ability. When comparing these models we incorporated the relevant improvements from our experiments—including subpixel maxima and predicting multi-scale geometry between keypoints—unless otherwise noted (see Appendix A.7).

While we do make comparisons to the DeepLabCut model (A. Mathis et al., 2018) we do not use the same training routine as A. Mathis et al. (2018) and Nath et al. (2019a), who use binary cross-entropy loss for optimizing the confidence maps in addition to the location refinement maps described by Insafutdinov et al. (2016). We made this modification in order to hold the training routine constant for each model while only varying the model itself. However, we find that these differences between training routines effectively have no impact on performance when the models are trained using the same dataset and data augmentations (Appendix A.7 Figure A.7). We also provide qualitative comparisons to demonstrate that, when trained with our DeepPoseKit framework, our implementation of the DeepLabCut model (A. Mathis et al., 2018) appears to produce fewer prediction errors than the original implementation from A. Mathis et al. (2018); Nath et al. (2019a) when applied to a novel video (Appendix A.7 Figure A.7-Figure supplements A.8 and A.9; Appendix A.7 Figure A.7-video A.10).

2.5.1 Datasets

We performed experiments using the vinegar or "fruit" fly (*Drosophila melanogaster*) dataset (Figure 2.6-video 2.7) 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*) recorded in a laboratory setting (Figure 2.6-video 2.8), and herds of Grévy's zebras (*Equus grevyi*) recorded in the wild (Figure 2.6-video 2.9). The locust and zebra datasets are particularly challenging for pose estimation as they feature multiple interacting individuals—with focal individuals centered in the frame—and the latter with highly-variable environments and lighting conditions. These datasets are freely-available from `https://github.com/jgraving/deepposekit-data` (Graving et al., 2019b).

Our locust dataset consisted of a group of 100 locusts in a circular plastic arena 1-m in

diameter. The locust group was recorded from above using a high-resolution camera (Basler ace acA2040-90umNIR) and video recording system (Motif, loopbio GmbH). Locusts were localized and tracked using 2-D barcode markers (Graving, 2017) attached to the thorax with cyanoacrylate glue, and any missing localizations ($< 0.02\%$ of the total dataset) between successful barcode reads were interpolated with linear interpolation. Our zebra dataset consisted of variably sized groups in the wild recorded from above using a commercially-available quadcopter drone (DJI Phantom 4 Pro). Individual zebra were localized using custom deep-learning software based on Faster R-CNN (Ren et al., 2015) for predicting bounding boxes. The positions of each zebra were then tracked across frames using a linear assignment algorithm (Munkres, 1957) and data were manually verified for accuracy.

After positional tracking, the videos were then cropped using the egocentric coordinates of each individual and saved as separate videos—one for each individual. The images used for each training set were randomly selected using the k-means sampling procedure (with k=10) described by Pereira et al. (2019) (Appendix A.2) to reduce correlation between sampled images. After annotating the images with keypoints, we rotationally and translationally aligned the images and keypoints using the central body axis of the animal in each labeled image. This step allowed us to more easily perform data augmentations (see "Model training") that allow the model to make accurate predictions regardless of the animal's body size and orientation (see Appendix A.5). However, this preprocessing step is not a strict requirement for training, and there is no requirement for this preprocessing step when making predictions on new unlabeled data, such as with the methods described by Pereira et al. (2019) (Appendix A.5). Before training each model we split each annotated dataset into randomly selected training and validation sets with 90% training examples and 10% validation examples, unless otherwise noted. The details for each dataset are described in Table 2.1.

Table 2.1. 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	900	9	Multiple	This paper

2.5.2 Model training

For each experiment, we set our model hyperparameters to the same configuration for our Stacked DenseNet and Stacked Hourglass models. Both models were trained with $\frac{1}{4}\times$ resolution outputs and a stack of two networks with two outputs where loss was applied (see Figure 2.3). 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 as the best configuration will depend on the application. The details of the hyperparameters we used for each model are described in Appendix A.7.

To make our posture estimation tasks closer to realistic conditions, incorporate prior information (Appendix A.2), and properly demonstrate the robustness of our methods to rotation, translation, scale, and noise, we applied various augmentations to each data set during training (Figure 2.3). All models were trained using data augmentations that included random flipping, or mirroring, along both the horizontal and vertical image axes with each axis being independently flipped by drawing from a Bernoulli distribution (with $p = 0.5$); random rotations around the center of the image drawn from a uniform distribution in the range [-180°, +180°); random scaling drawn from a uniform distribution in the range [90%, 110%] for flies and locusts and [75%, 125%] for zebras (to account for greater size variation in the data set); and random translations along the horizontal and vertical axis independently drawn from a uniform distribution with the range [-5%, +5%]—where percentages are relative to the original image size. After performing these spatial augmentations we also applied a variety of noise augmentations that included additive noise—i.e., adding or subtracting randomly-selected values to pixels; dropout—i.e., setting individual pixels or groups of pixels to a randomly-selected value; blurring or sharpening—i.e., changing the composition of spatial frequencies; and contrast ratio augmentations—i.e, changing the ratio between the highest pixel value and lowest pixel value in the image. These augmentations help to further ensure robustness to shifts in lighting, noise, and occlusions. See Appendix A.2 for further discussion on data augmentation.

We trained our models (Figure 2.3) using mean squared error loss optimized using the ADAM optimizer (Kingma & 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.

2.5.3 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, due to the small size of the datasets used for making comparisons, we elected to use only a validation set for model evaluation, as using an overly small training or test set can bias assessments of a model's predictive performance (Kuhn & Johnson, 2013). 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 data 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, unless otherwise noted. Our hardware consisted of a Dell Precision Tower 7910 workstation (Dell, Inc.) running Ubuntu Linux v18.04 with 2$\times$ Intel Xeon E5-2623 v3 CPUs (8 cores, 16 threads at 3.00GHz), 64GB of RAM, a Quadro P6000 GPU and a Titan Xp GPU (NVIDIA Corporation). We used both GPUs (separately) for training models and evaluating predictive performance, but we only used the faster Titan Xp GPU for benchmarking inference speeds and training time. While the hardware we used for development and testing is on the high-end of the current performance spectrum, 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 ($\pm$ 10%) when using the same GPU; however, training speed depends more heavily other hardware components like the CPU and hard disk.

2.5.4 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 $h(x) = \log\left(1 + e^{\mathbf{x}}\right)$—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 accuracy 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 A.6.

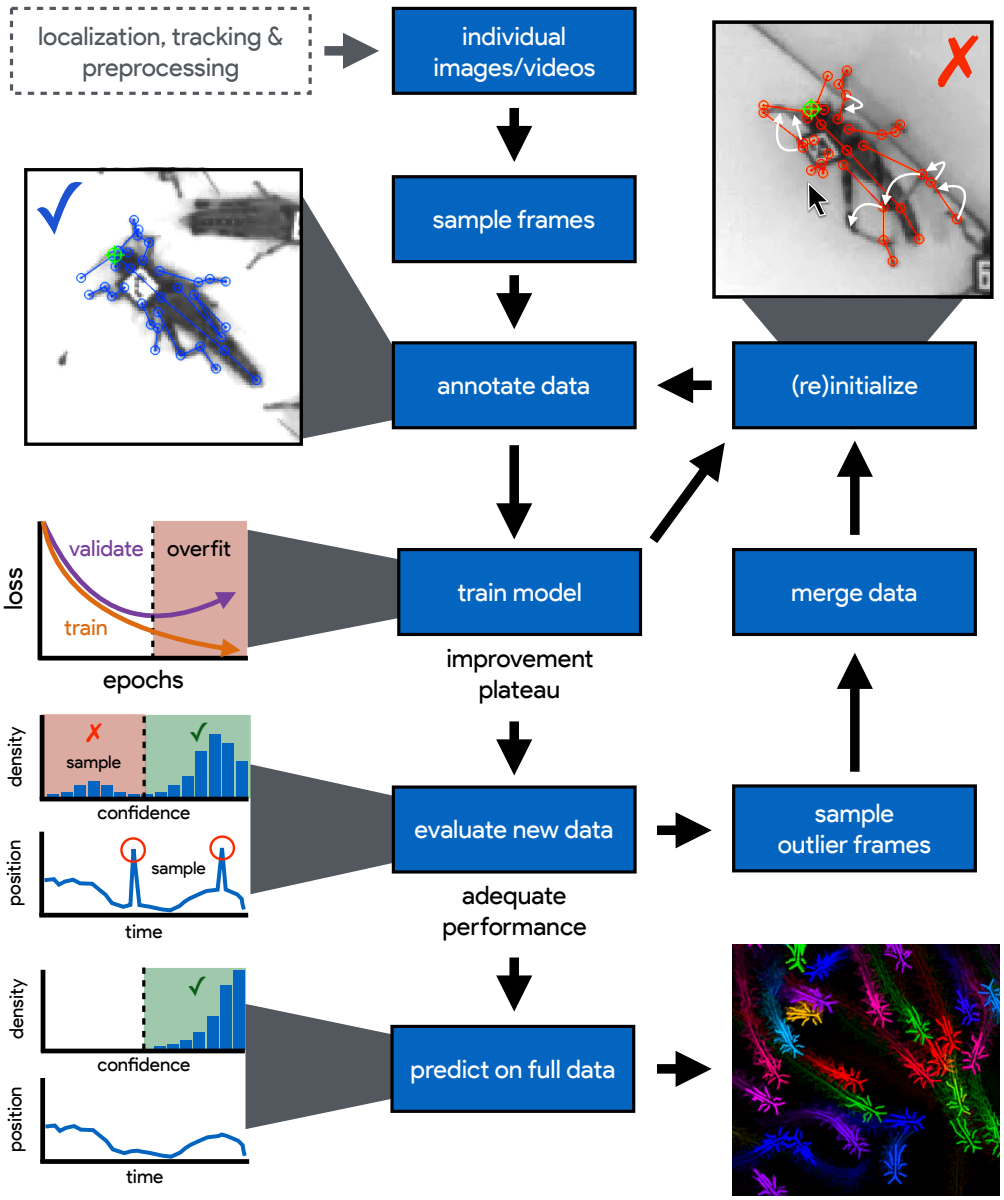

Figure 2.1. An illustration of the workflow for DeepPoseKit. Multi-individual images are localized, tracked, and preprocessed into individual images, which is not required for single-individual image datasets. An initial image set is sampled, annotated, and then iteratively updated using the active learning approach described by Pereira et al. (2019) (see Appendix A.2). As annotations are made, the model is trained (Figure 2.3) with the current training set and keypoint locations are initialized for unannotated data to reduce the difficulty of further annotations. This is repeated until there is a noticeable improvement plateau for the initialized data—where the annotator is providing only minor corrections—and for the validation error when training the model (Appendix A Figure A.4). New data from the full dataset are evaluated with the model, and the training set is merged with new examples that are sampled based on the model's predictive performance, which can be assessed with techniques described by A. Mathis et al. (2018); Nath et al. (2019a) for identifying outlier frames and minimizing extreme prediction errors—shown here as the distribution of confidence scores predicted by the model and predicted body part positions with large temporal derivatives—indicating extreme errors. This process is repeated as necessary until performance is adequate when evaluating new data. The pose estimation model can then be used to make predictions for the full data set, and the data can be used for further analysis.

Figure 2.2. A visualization of the posture data output for a group of locusts (5× speed) `https://youtu.be/hCa2zaoUWhs`

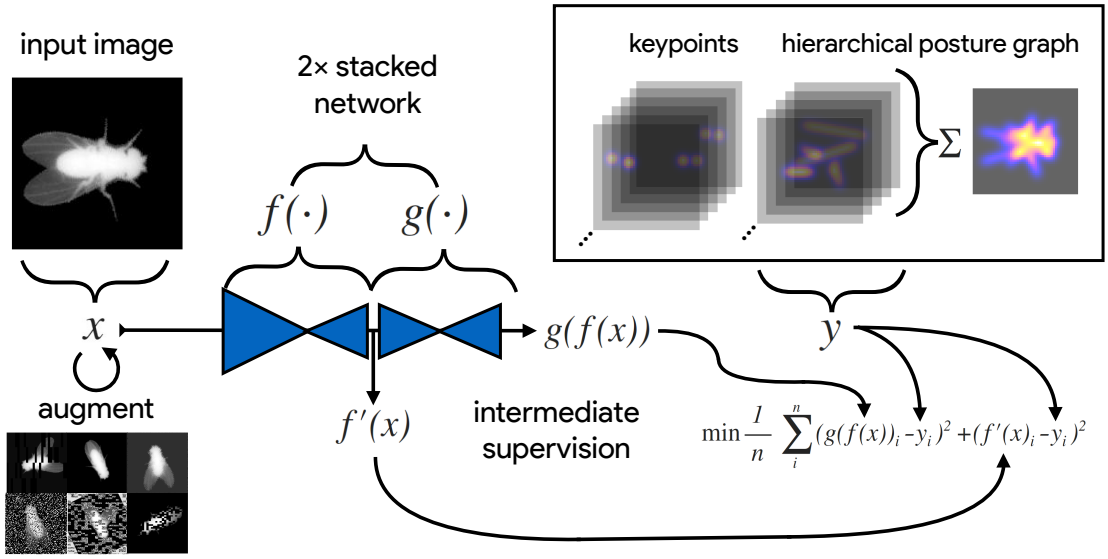

Figure 2.3. An illustration of the model training process for our Stacked DenseNet model in DeepPoseKit (see Appendix A.1 for details about training models). Input images x (**top-left**) are augmented (**bottom-left**) with various spatial transformations (rotation, translation, scale, etc.) followed 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 networks $g(f(x))$ 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 A.4). 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.

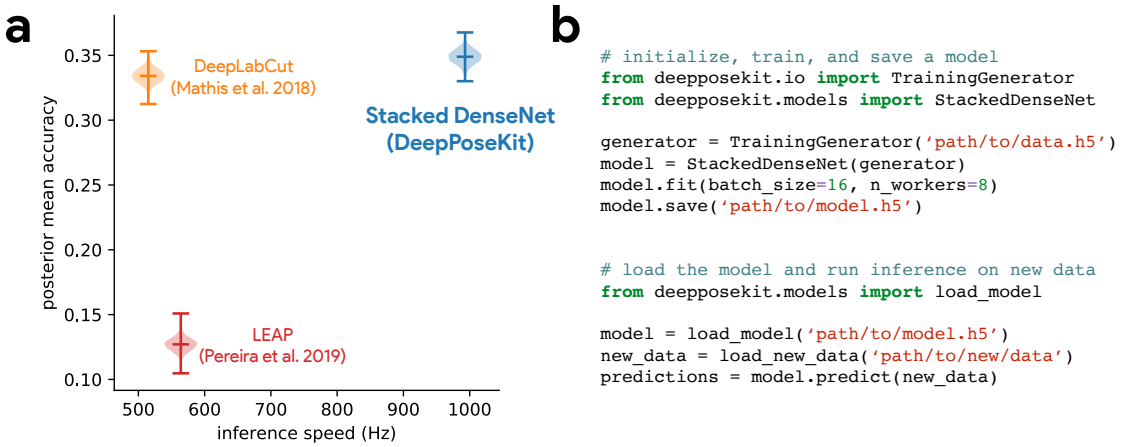

Figure 2.4. Our Stacked DenseNet model estimates posture at approximately 2×—or greater—the speed of the LEAP model (Pereira et al., 2019) and the DeepLabCut model (A. Mathis et al., 2018) while also achieving similar accuracy to the DeepLabCut model (A. Mathis et al., 2018)—shown here as mean accuracy $(1 + \text{Euclidean error})^{-1}$ for our most challenging dataset of multiple interacting Grévy's zebras (*E. grevyi*) recorded in the wild (**a**). See Figure 2.4 supplement 2.5 for further details. 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**).

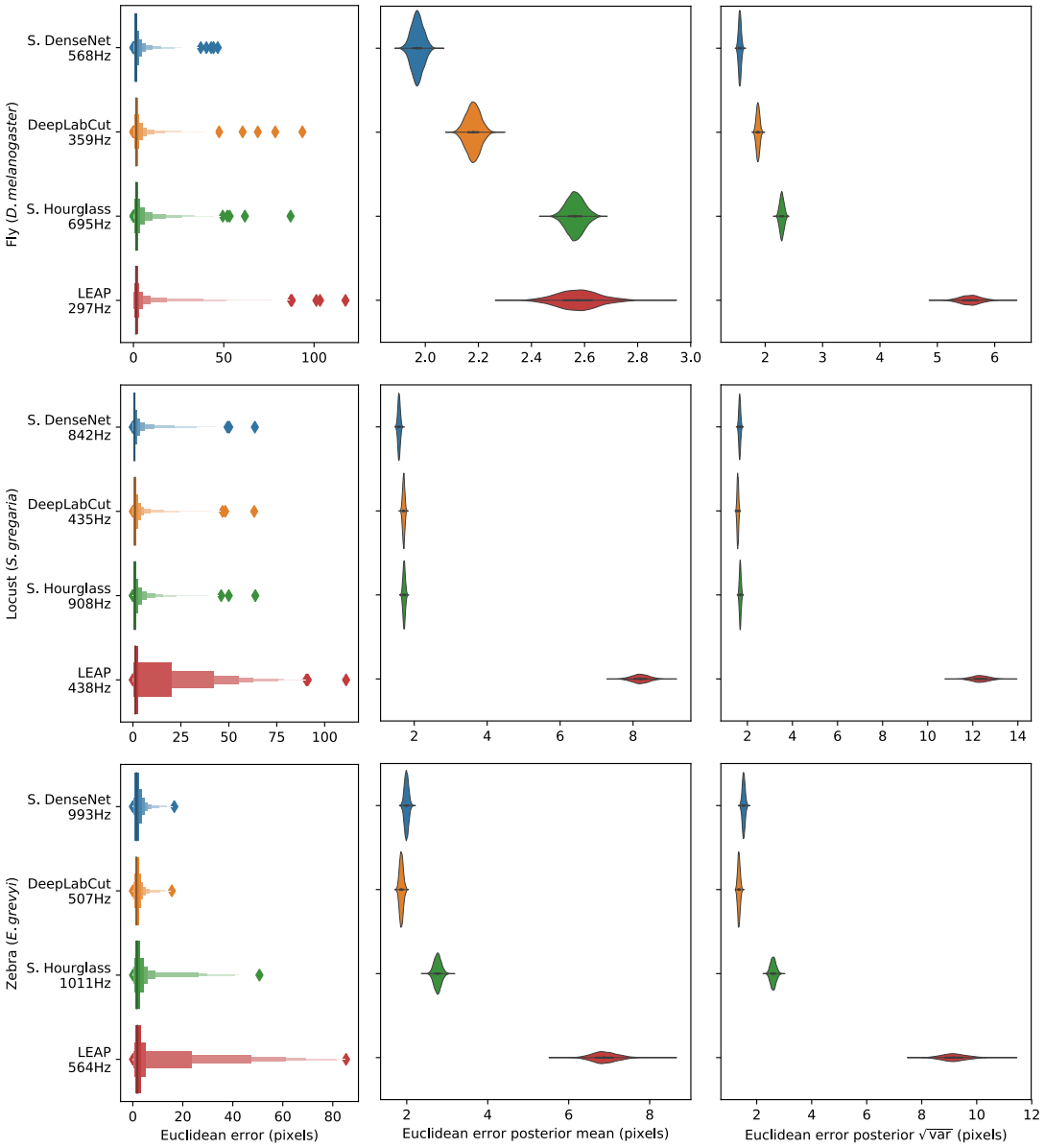

Figure 2.5. Euclidean error distributions for each model across our three datasets. Letter-value plots (**left**) show the raw error distributions for each model. Violinplots of the posterior distributions for the mean and variance (**right**) show statistical differences between the error distributions. Overall the LEAP model (Pereira et al., 2019) was the worst performer on every dataset in terms of both mean and variance. Our Stacked Densenet model was the best performer for the fly dataset, while our Stacked DenseNet model and the DeepLabCut model (A. Mathis et al., 2018) both performed equally well on the locust and zebra datasets. The posteriors for the DeepLabCut model (A. Mathis et al., 2018) and our Stacked DenseNet model are highly overlapping for these datasets, which suggests they are not statistically discernible from one another. Our Stacked Hourglass model (Newell et al., 2016) performed equally to the DeepLabCut model (A. Mathis et al., 2018) and our Stacked DenseNet model for the locust dataset but performed slightly worse for the fly and zebra datasets.

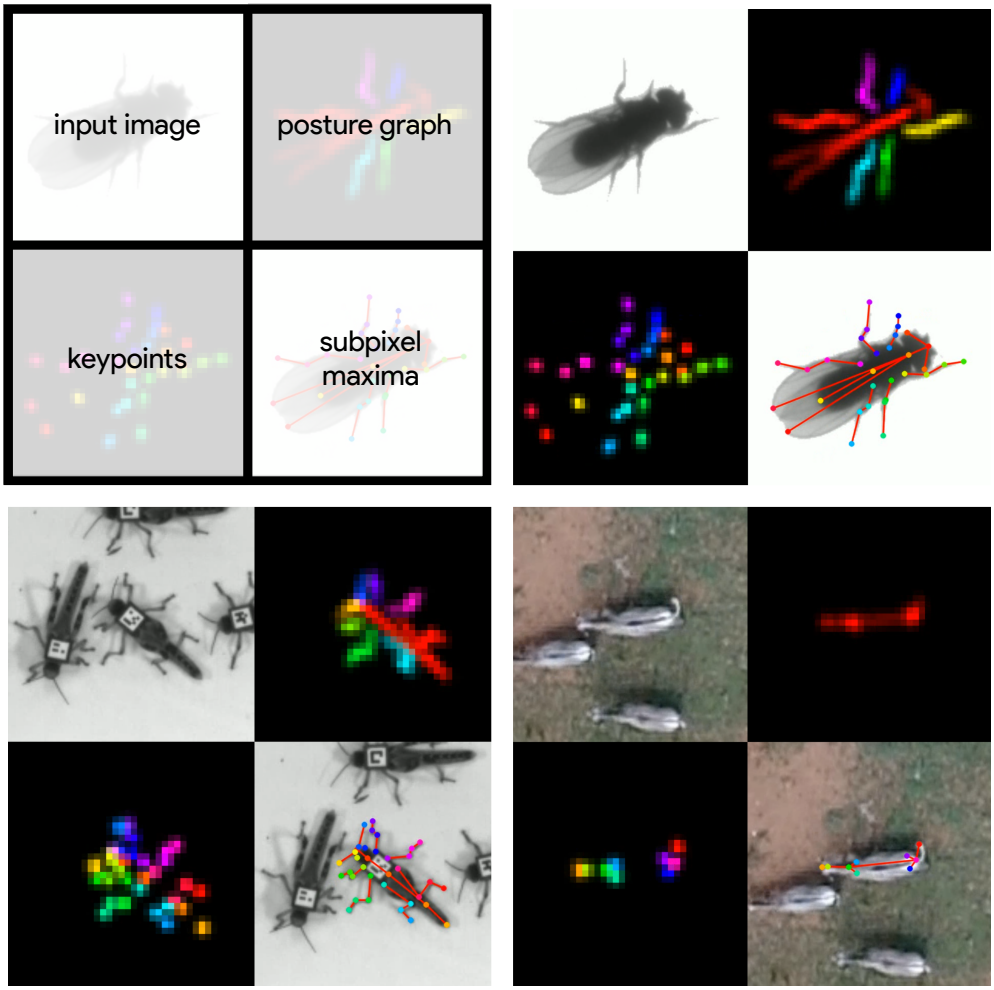

Figure 2.6. A visualization of the datasets we used to evaluate our methods (Table 2.1). For each dataset, confidence maps for the keypoints (bottom-left) and posture graph (top-right) are illustrated using different colors for each map. These outputs are from our Stacked DenseNet model at $\frac{1}{4}\times$ resolution.

Figure 2.7. A video of a behaving fly from Pereira et al. (2019) with pose estimation outputs visualized `https://youtu.be/lsnex6k4NRs`

Figure 2.8. A video of a behaving locust with pose estimation outputs visualized. `https://youtu.be/b0DyyLP_Czk`

Figure 2.9. A video of a behaving Grévy's zebra with pose estimation outputs visualized. `https://youtu.be/dSjaphoGHAY`

Chapter 3

VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering

Jacob M. Graving, and Iain D. Couzin

Adapted from: Graving, J. M., Couzin, I. D. (2020). VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering. bioRχiv, 207993 under a CC-BY-4.0 International License

3.1 Abstract

Scientific datasets are growing rapidly in scale and complexity. Consequently, the task of understanding these data to answer scientific questions increasingly requires the use of compression algorithms that reduce dimensionality by combining correlated features and cluster similar observations to summarize large datasets. Here we introduce a method for both dimension reduction and clustering called VAE-SNE (variational autoencoder stochastic neighbor embedding). Our model combines elements from deep learning, probabilistic inference, and manifold learning to produce interpretable compressed representations while also readily scaling to tens-of-millions of observations. Unlike existing methods, VAE-SNE simultaneously compresses high-dimensional data and automatically learns a distribution of clusters within the data — without the need to manually select the number of clusters. This naturally creates a multi-scale representation, which makes it straightforward to generate coarse-grained descriptions for large subsets of related observations and select specific regions of interest for further analysis. VAE-SNE can also quickly and easily embed new samples, detect outliers, and can be optimized with small batches of data, which makes it possible to compress datasets that are otherwise too large to fit into memory. We evaluate VAE-SNE as a general purpose method for dimensionality reduction by applying it to multiple real-world datasets and by comparing its performance with existing methods for dimensionality reduction. We find that VAE-SNE produces high-quality compressed representations with results that are on par with existing nonlinear dimensionality reduction algorithms. As a practical example, we demonstrate how the cluster distribution learned by VAE-SNE can be used for unsupervised action recognition to detect and classify repeated motifs of stereotyped behavior in high-dimensional timeseries data. Finally, we also introduce variants of VAE-SNE for embedding data in polar (spherical) coordinates and for embedding image data from raw pixels. VAE-SNE is a robust, feature-rich, and scalable method with broad applicability to a range of datasets in the life sciences and beyond.

3.2 Introduction

Modern scientific research generates large, high-resolution datasets that are complex and high-dimensional, where a single observation from an experimental system can contain measurements describing hundreds, or thousands, of features. For example, neuroscientists measure electrical activity across thousands of individual neurons simultaneously (Jun et al., 2017; Stringer,

Pachitariu, Steinmetz, Carandini, & Harris, 2019; Stringer, Pachitariu, Steinmetz, Reddy, et al., 2019) — even across the entire brain (Ahrens et al., 2012, 2013); cell biologists and bioinformaticians routinely sequence the transcriptome for thousands of genes across large populations of single cells (Becht et al., 2019; La Manno et al., 2018; Linderman et al., 2019; Samusik et al., 2016); behavioral scientists measure the high-dimensional body posture dynamics of animals and humans (Bala et al., 2020; Berman et al., 2014a; Cande et al., 2018; Chambers et al., 2019; Costa et al., 2019; Ebbesen & Froemke, 2020; Graving et al., 2019a; Günel et al., 2019; J. Kain et al., 2013; Karashchuk et al., 2020; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; A. Mathis et al., 2018; Nath et al., 2019b; Pereira et al., 2019; Stephens et al., 2011, 2008; Wiltschko et al., 2015); and evolutionary ecologists measure complex morphological patterns across sizeable collections of animal specimens (I. C. Cuthill et al., 2017; J. F. H. Cuthill et al., 2019; Ezray et al., 2019; Wham et al., 2019; Q. Zhang et al., 2019). While there are many benefits to measuring real-world systems accurately and completely for answering scientific questions, this added complexity poses problems for conventional data analysis methods — especially those commonly used in the life sciences, like linear models (Bolker et al., 2009) — that are designed for small, low-dimensional datasets and typically rely on simplified models with strong, often unrealistic, assumptions for making statistical inferences.

To deal with the complexity of modern data, researchers in many fields have begun to use machine-learning methods known as *dimensionality reduction* and *clustering* to help interpret large, high-dimensional datasets. These algorithms distill correlated features down to a smaller set of components (dimensionality reduction) or group large subsets of observations into a smaller set of classes based on similarity (clustering). Together these methods offer scientists a way to *compress* data, where compression is typically performed with the goal of reducing the size and complexity of a dataset while making only minimal, or very general, a priori assumptions about the true distribution of the data. Because these algorithms derive their compressed representations directly from the structure of the data itself, without human supervision, they are typically known as *unsupervised learning* algorithms.

Across many scientific disciplines, unsupervised algorithms are rapidly becoming a commonly-used tool for visualizing and interpreting high-dimensional data distributions as well as summarizing large datasets with coarse-grained descriptions and identifying specific subpopulations and regions of interest within the data for further downstream analysis. Researchers have applied these methods to demonstrate how the brain organizes behavior (Berman et al., 2016; Billings

et al., 2017; Brown et al., 2013; Cande et al., 2018; Costa et al., 2019; Markowitz et al., 2018; Stephens et al., 2011, 2008; Stringer, Pachitariu, Steinmetz, Carandini, & Harris, 2019; Stringer, Pachitariu, Steinmetz, Reddy, et al., 2019; Wiltschko et al., 2015); describe how cells grow and develop over time (La Manno et al., 2018); document new and rare types of cells (Grün et al., 2015; Linderman et al., 2019); gain insights into cancer treatment (Tirosh et al., 2016); and reveal fundamental principles of evolution (J. F. H. Cuthill et al., 2019; Ezray et al., 2019; Wham et al., 2019). Therefore, as scientists begin to regularly rely on these algorithms for analyzing complex datasets, the task of ensuring the quality, robustness, and utility of the compressed representations they produce is an issue of considerable importance — as is the ability to scale these methods to increasingly large datasets.

While existing methods for dimension reduction produce high-quality compressed representations (Becht et al., 2019; Kobak & Linderman, 2019), they typically lack features for identifying groups of similar data (i.e., learned clusters; but see Pezzotti et al. 2016; Robinson & Pierce-Hoffman 2020), and despite much progress to improve scalability of existing algorithms (Linderman et al., 2017, 2019; McInnes et al., 2018), some of the most widely-used methods are still limited in their ability to scale beyond a few million observations without specialized, high-performance hardware — especially in the case of large, out-of-core datasets that cannot fit into memory. Recent applications of deep learning (Goodfellow et al., 2016), and deep generative models in particular (Appendix B.1.1; Kingma & Welling 2013; Rezende et al. 2014), have begun to address these issues (Ding et al., 2018; Ding & Regev, 2019; Szubert et al., 2019). Nevertheless, even with the low memory and computational cost of deep learning methods that can be trained with small batches of data on consumer-grade hardware, these new algorithms are still significantly slower to fit to data than more popular methods because they require costly nearest neighbor or pairwise distance calculations (Becht et al., 2019; Ding et al., 2018; Szubert et al., 2019). The majority of these methods also do not provide any built-in mechanism for detecting outliers, which could potentially bias any downstream results and cause statistical errors when testing hypotheses.

There has also been a flurry of recent work on advanced methods for clustering data (e.g., Campello et al. 2013; Fogel et al. 2019; Guo et al. 2017; Jiang et al. 2016; McInnes et al. 2017; Robinson & Pierce-Hoffman 2020; Xie et al. 2016; Yang et al. 2019; and numerous others), including efficient methods that rely on deep learning and deep generative models. However, the vast majority of these methods impose strong assumptions about the shape of the clusters and

require the user to manually select the number of clusters fitted to the data — or, alternatively, involve complex computations that do not scale well to large datasets. Determining how many clusters to fit is typically a non-trivial, unintuitive, and computationally-intensive task for datasets where the number of clusters is not known a priori (Fang & Wang, 2012; Milligan & Cooper, 1985; Pham et al., 2005; Todd et al., 2017). Many recently proposed clustering algorithms are also only evaluated with relatively small “toy" datasets, such as the MNIST handwritten digit database (LeCun et al., 2010), where the data typically have very little noise, no outliers, and the number of clusters is often known a priori. This lack of rigorous real-world assessment casts doubt on the practical utility of these algorithms in cases where datasets have a large number of observations, are naturally noisy or contain outliers, and the number of clusters is unknown, such as those commonly used in the natural sciences.

Here we aim to address many of the limitations outlined above and unify some of the key methodological concepts from previous work into a single modeling framework. To accomplish this, we introduce a deep generative model for both dimensionality reduction and clustering. We then compare our model with existing methods for dimensionality reduction, and importantly, to ensure that it has practical utility, we demonstrate the application of our method using empirical examples with real-world data from multiple domains. In comparison to existing dimension reduction methods, our proposed method produces low-dimensional data representations with similar, or better, quality while also offering several key improvements. Notably, our approach provides the ability to scale to datasets containing tens-of-millions of observations without specialized, high-performance hardware and automatically learns an interpretable cluster distribution from the data without any manual tuning or expensive computations to determine the number of clusters. Together these results demonstrate that our proposed method is a robust, feature-rich, and scalable tool for data analysis and is widely-applicable to a variety of tasks.

3.3 Results

We make three main contributions in this paper: (**1**) First, we introduce a deep generative model for both dimensionality reduction and clustering called variational autoencoder stochastic neighbor embedding (VAE-SNE; Fig. 3.1; Methods). VAE-SNE can produce a variety of different compressed representations and readily scales to out-of-core datasets with tens-of-millions of observations. Our model builds on numerous ideas from past work by synthesizing methods from

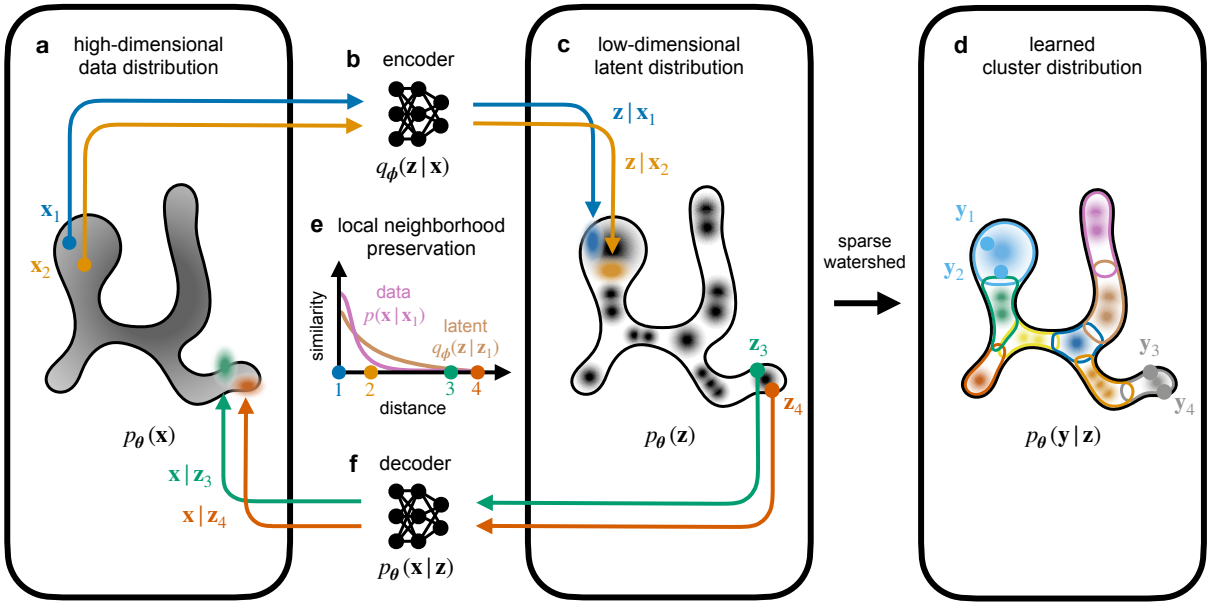

Figure 3.1. Overview of the VAE-SNE model. **a**-**f**, Observed samples from a high-dimensional data distribution $\mathbf{x} \sim p(\mathbf{x})$ (**a**) are probabilistically embedded (**b**) into a low-dimensional latent distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ (**c**) using an encoder deep neural network $\mathrm{DNN}_{\boldsymbol{\phi}} : \mathbf{x} \rightarrow \mathbf{z}$ to generate an approximate latent posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. Samples from the latent distribution $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ or $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z})$ (**c**) are then transformed (**f**) using a generative decoder deep neural network $\mathrm{DNN}_{\boldsymbol{\theta}} : \mathbf{z} \rightarrow \mathbf{x}$ to probabilistically reconstruct the high-dimensional data distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Given a set of observed high-dimensional data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ the model parameters for the encoder and decoder $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$ are optimized so that the approximate posterior for the encoder matches the true posterior from the generative decoder as best as possible, or $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, which then creates a functional mapping between the high-dimensional and low-dimensional distributions. To improve local structure preservation during optimization, pairwise distances between vectors in the high-dimensional and low-dimensional space are optimized using pairwise similarity kernels (**e**), a probability density function of distance, so that the local neighborhoods around each observation match as best as possible, or $p(\mathbf{x}|\mathbf{x}_i) \approx q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{z}_i)$. This preferentially weights the preservation of local neighborhoods over global relationships by assigning more probability mass to nearby neighbors during optimization. The prior for the latent distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ is also a learned Gaussian mixture distribution (**c**) that is jointly optimized with the encoder and decoder to fit the observed data and can be used to cluster the latent distribution (**d**) into a small set of discrete classes $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})$ — where highly-overlapping modes (mixture components) within the distribution are automatically merged into the same class label using sparse watershed assignment (Methods; Todd et al. 2017)

a class of generative models known as variational autoencoders (VAEs; Kingma & Welling 2013), the popular dimensionality reduction algorithm (*t*-distributed) stochastic neighbor embedding (SNE/t-SNE;Hinton & Roweis 2003; van der Maaten & Hinton 2008) and its many extensions (Chien & Hsu, 2017; Ding et al., 2018; van der Maaten, 2009; Wang & Wang, 2016), as well as recent advances in variational inference (Burda et al., 2015; Cremer et al., 2017; Dilokthanakul

et al., 2016; Kingma et al., 2014; Tomczak & Welling, 2017) and clustering methods (Todd et al., 2017). (**2**) Second, we apply VAE-SNE, and a variety of other popular dimensionality reduction methods, to compress real-world datasets from different domains (Fig. 3.2). We then quantitatively assess how each algorithm performs in preserving important aspects of the data — including information about local, global, and temporal structure. We also assess generalization to new, out-of-sample data and compare processing speeds for each algorithm. Additionally, we show how the likelihood score produced by VAE-SNE can be used to detect outliers when embedding out-of-sample data. (**3**) Third, we show how VAE-SNE can be used to automatically cluster large datasets into a small set of interpretable classes. As a practical example, we apply VAE-SNE to a dataset of 21.1 million observations describing the high-dimensional body posture dynamics of a commonly-used model organism — the fruit fly (*Drosophila melanogaster*) — to automatically discretize these data into motifs of stereotyped behavior for further analysis (Fig. 3.3; Berman et al. 2014a; Pereira et al. 2019). These results illustrate how VAE-SNE can be used as a type of automated ethogram for describing the full behavioral repertoire of animals (reviewed by Anderson & Perona 2014; Berman 2018; Brown & De Bivort 2018; Datta et al. 2019), while also providing several advantages over existing methods for this task.

Our approach (Fig. 3.1; Methods) builds on VAEs as a base model for performing dimensionality reduction (Appendix B.1.1), which, like other types of autoencoders (Hinton & Salakhutdinov, 2006), model high-dimensional data using two deep neural networks: one to encode data to a compressed latent representation, and another to decode the latent vectors and reconstruct the data. However, VAEs are distinct from other autoencoders in that the encoder is used to parameterize continuous distributions of latent vectors — from which latent vectors are then probabilistically sampled — rather than embedding each high-dimensional observation as a single point in the latent space. This type of model offers an attractive dimensionality reduction framework because the objective function (Appendix B.1.2) naturally imparts a trade-off between the complexity of the encoded description and the overall accuracy of the decoded reconstruction (Alemi et al., 2016). However, these models suffer from multiple long-standing issues including a phenomenon known as *posterior collapse* (Alemi et al., 2017; Dieng, Kim, et al., 2019) where the latent coordinate space becomes arbitrarily organized and no longer preserves any statistical features of the high-dimensional data distribution. There has been a string of recent work to address these issues including some relatively straightforward solutions (Dieng, Kim, et al., 2019; Higgins et al., 2016) that achieve varying levels of success, as well as new objective functions

that involve regularizing the mutual information between the high-dimensional data and latent distribution (e.g., Rezaabad & Vishwanath 2019; Zhao et al. 2017; reviewed by Poole et al. 2019).

For VAE-SNE, we provide an effective solution to this problem with the addition of a stochastic neighbor regularizer (Appendix B.2; Chien & Hsu 2017; Ding et al. 2018; van der Maaten 2009; van der Maaten & Hinton 2008) that optimizes pairwise similarity kernels between the high- and low-dimensional distributions to strengthen local neighborhood preservation and more explicitly retain a useful representation. We also draw on other theoretical and practical improvements from the literature to enhance the performance of VAE-SNE (Methods). For example, we use a Gaussian mixture prior for learning the latent distribution (Dilokthanakul et al., 2016; Kingma et al., 2014; Tomczak & Welling, 2017). This choice of distribution allows for better local structure preservation and, when combined with sparse watershed assignment to merge overlapping mixture components (Fig. 3.1; Methods; Todd et al. 2017), serves as a flexible method for clustering data — without the need to manually define the number of clusters or impose strong assumptions about cluster shape. We employ several other advances to further improve structure preservation. For instance, we apply a perplexity annealing technique (Kobak & Berens, 2019) to slowly decay the size of the local neighborhoods optimized by the model during training, which helps to preserve structure across multiple scales. Moreover, we extensively optimize the algorithms underlying our model by applying parallel computations on the CPU and GPU that dramatically improve processing speed compared to previous work (Ding et al., 2018).

In addition to our three main contributions, we further extend VAE-SNE to demonstrate its flexibility as a framework for dimensionality reduction. To accomplish this, we introduce a von Mises-Fisher variant of VAE-SNE (Appendix B.3.1; Fig. B.10; Video S8, Video S9) that embeds data in polar coordinates (rather than Euclidean coordinates) on a 3-D unit sphere, which is potentially a more natural representation for many high-dimensional datasets (Davidson et al., 2018) and solves the “crowding" problem common to some methods (Ding & Regev, 2019; van der Maaten & Hinton, 2008). Finally, we also apply a modified convolutional version of VAE-SNE (Appendix B.3.2; Figs. B.11, B.12) to visualize natural history images of animal specimen collections (J. F. H. Cuthill et al., 2019; Q. Zhang et al., 2019) by directly embedding the raw pixel data. Our results for these two extensions are described in Appendix B.3.

3.3.1 Comparisons with other dimension reduction algorithms

Current methods for dimensionality reduction generally fall into two classes known as *linear* and *nonlinear* algorithms. Linear algorithms, such as principal components analysis (PCA), compress high-dimensional data by learning linearly weighted combinations (affine transformations) of the original feature set. Typically these algorithms are optimized to preserve the global structure of the data, where local neighborhood relationships are distorted in order to maintain the full coordinate system of the original features as best as possible. On the other hand, nonlinear algorithms (sometimes called manifold learning algorithms) such as t-SNE (van der Maaten & Hinton 2008) and uniform manifold approximation and projection (UMAP; McInnes et al. 2018) typically take the opposite approach of prioritizing relative relationships between data points rather than the global coordinate system. This approach allows local neighborhoods to be preserved while potentially sacrificing information about the larger-scale relationships between data points in the global coordinate space — although, as we demonstrate here, the global distortion imposed by many of these algorithms is actually comparable to that of PCA.

To validate VAE-SNE as a general-purpose method for dimensionality reduction, we quantitatively compare its performance with other dimension reduction algorithms — both linear and nonlinear — using two datasets from different domains (see Methods) describing animal body part dynamics (Berman et al., 2016, 2014a; Pereira et al., 2019) and single-cell RNA-seq expression profiles for hippocampal neurons (La Manno et al., 2018). We benchmark multiple variants of VAE-SNE with different pairwise similarity kernels for preserving local neighborhood information (including kernel functions with learned parameters; Appendix B.2), and we compare these results with those from two high-performance variants of t-SNE (van der Maaten & Hinton, 2008) known as FIt-SNE (Linderman et al., 2017, 2019) and Barnes-Hut-SNE (van der Maaten, 2014), as well as UMAP (McInnes et al., 2018), and two other deep neural network-based dimension reduction methods: scvis (Ding et al., 2018), and ivis (Szubert et al., 2019). We also apply PCA in 2, 5, 10, and 100 dimensions for a linear baseline comparison. We fit each algorithm with a training set and also embed an out-of-sample test set to assess generalization to new data. For both the training and test sets, we then quantitatively assess each algorithm's ability to preserve different types of information about the high-dimensional data when compressing the data to two dimensions, including local, global, fine-scale, and temporal information (Methods). We quantify local information preservation for each algorithm by measuring the preservation of both metric (distance- or radius-based) and topological (nearest neighbors-based) neighborhoods that are

approximately 1% of the total embedding size; we measure global information preservation by calculating the correlation between pairwise distances in high- and low-dimensional space; we assess fine-scale information by measuring neighborhood preservation for multiple neighborhood sizes $< 1\%$ of the total embedding size; and we evaluate temporal information preservation by computing the correlation between high- and low-dimensional temporal derivatives in a timeseries dataset. Overall the qualitative properties of the embeddings produced by each algorithm are strikingly similar within datasets (Fig. 3.2), which likely indicates shared mathematical properties of how the latent distributions are modeled. However, we do find potentially important quantitative differences between these algorithms in terms of information preservation and processing speed. We summarize our overall assessments of each nonlinear dimension reduction algorithm in Tables B.1, B.2, B.3.

Local structure preservation

We find that VAE-SNE compares closely to FIt-SNE (Linderman et al., 2017), Barnes-Hut-SNE (van der Maaten, 2014), and UMAP (McInnes et al., 2018) in preserving local structure for both the training set (Figs. B.1a, B.2a, B.5a) and test set (Figs. B.3a, B.4a), while scvis (Ding et al., 2018) and ivis (Szubert et al., 2019) perform slightly worse. Our results show that VAE-SNE with a t-SNE similarity kernel (van der Maaten & Hinton, 2008) performs the best for preserving local structure, but VAE-SNE with a Gaussian SNE kernel (Hinton & Roweis, 2003) also performs well — similarly to scvis (Ding et al., 2018) and ivis (Szubert et al., 2019). We also find that learning the similarity kernel parameters (for both Gaussian and Student's *t* kernels) as a function of each data point does not improve performance for our local preservation metrics. The top performing algorithms for local structure preservation (VAE-SNE, t-SNE, and UMAP) are closely comparable to 5-dimensional PCA for both metrics we used to assess local neighborhood preservation.

Global structure preservation

We find that VAE-SNE also does well in preserving global structure for both the training set (Figs. B.1a, B.2b, B.5a) and test set (Figs. B.3a, B.4b). VAE-SNE with a Gaussian SNE kernel performs best for this metric, but VAE-SNE with a t-SNE kernel also performs nearly as well. Notably all the neural-network-based methods (VAE-SNE, scvis Ding et al. 2018, ivis Szubert et al. 2019) outperform both t-SNE and UMAP (McInnes et al., 2018) in preserving global structure for both datasets we tested. This is perhaps not surprising given that recent work has shown

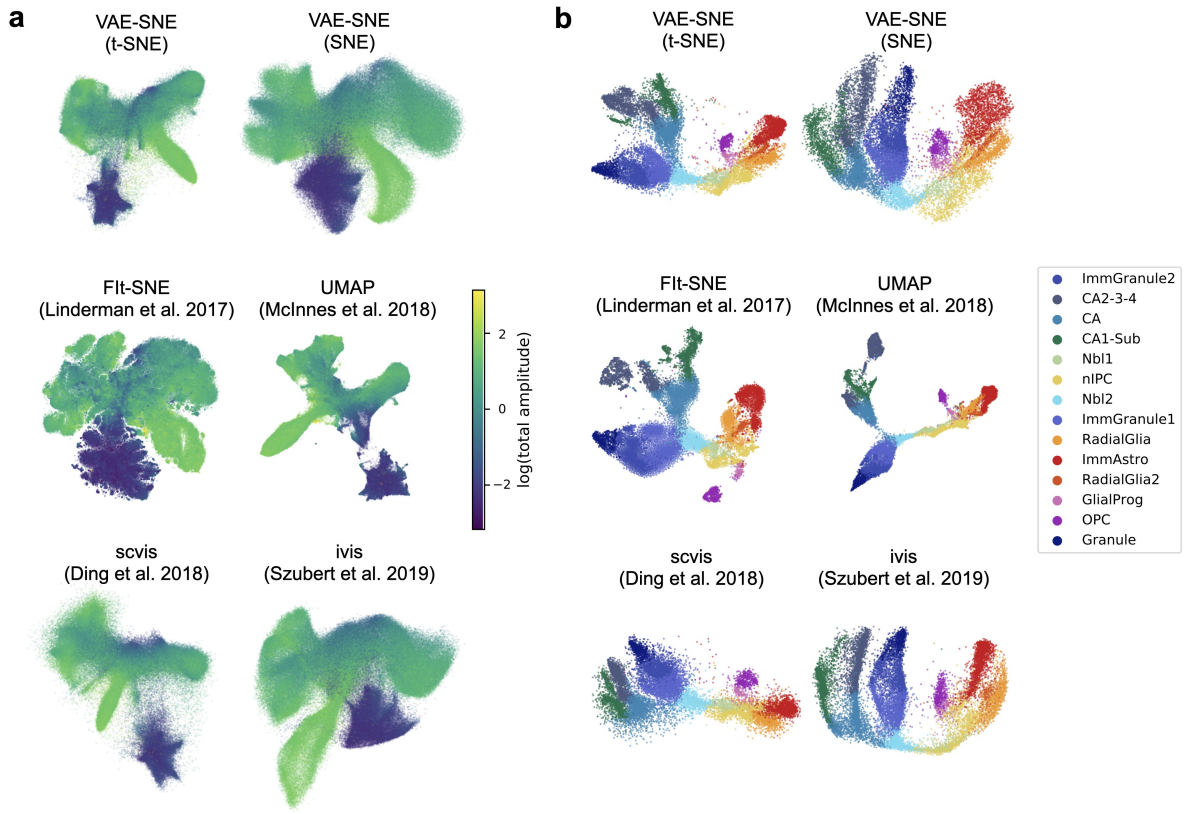

Figure 3.2. Embeddings for body posture dynamics and single-cell RNA-seq data. a, 2-D embeddings of body posture dynamics data from Berman et al. (2016, 2014a); Pereira et al. (2019) for each algorithm we tested. The color of each point indicates the logarithm of the total amplitude (overall movement) of body parts for each observation. **b**, 2-D embeddings of single-cell RNA-seq data of developing hippocampal neurons from La Manno et al. (2018) for each algorithm. The color of each point indicates the cell type for that observation as described by La Manno et al. (2018).

neural network models tend to learn the same axes as PCA (Rolinek et al., 2019). Additionally, these results show that learning the similarity kernel parameters as a function of each data point does improve global structure preservation for VAE-SNE with a t-SNE kernel — likely because it is optimized to be more similar to the Gaussian kernel used to calculate high-dimensional similarities (Appendix B.2). The top performing algorithms for this metric are comparable to 2-dimensional PCA, which demonstrates that nonlinear algorithms are capable of preserving the same global information as PCA while also better preserving local structure. On one hand, the scvis (Ding et al., 2018) algorithm in particular excels at preserving global structure for the single-cell RNA-seq dataset we tested (Fig. B.5a), while, on the other hand, ivis (Szubert et al., 2019) performs much more poorly than the other neural network algorithms for this dataset, and

FIt-SNE (Linderman et al., 2017, 2019) and Barnes-Hut-SNE (van der Maaten, 2014) perform even worse. We also show that UMAP (McInnes et al., 2018) with PCA initialization better preserves global structure than the default Laplacian Eigenmap initialization.

Fine-scale structure preservation

In addition to local and global structure preservation, we evaluate the ability of each algorithm to preserve very fine-scale neighborhood information (Figs. B.1b, B.3b, B.5b). We find that both FIt-SNE (Linderman et al., 2017) and Barnes-Hut-SNE (van der Maaten, 2014) excel at preserving this fine-scale information for the posture dynamics dataset (Figs. B.1b, B.3b) while every other nonlinear algorithm performs relatively poorly for both the training and test set. For the single-cell RNA-seq dataset, this distinction is not nearly as large and the algorithms all perform more similarly (Fig. B.5b), which indicates performance varies depending on the dataset. Performance for the ivis algorithm (Szubert et al., 2019) is especially poor for this metric on the single cell RNA-seq dataset. However, neighborhood membership for neighborhoods between 1% and 10% of the total embedding size are all similarly well-preserved for each algorithm.

Temporal structure preservation

Because one of the datasets we use for benchmarking is a behavioral timeseries, for these data we also assess the temporal structure preservation of each algorithm (Figs. B.3a, B.4c) on the out-of-sample test set (the training set is randomly sampled across multiple timeseries, so temporal information is not preserved). We find that VAE-SNE (particularly the SNE kernel variant), FIt-SNE (Linderman et al., 2017), Barnes-Hut-SNE (van der Maaten, 2014), scvis (Ding et al., 2018), and ivis (Szubert et al., 2019) perform at the same level as 5-dimensional PCA in preserving temporal structure, while UMAP (McInnes et al., 2018) performs relatively poorly in comparison to the other algorithms — even worse than 2-dimensional PCA.

Speed comparisons

In addition to assessing information preservation, we also compare the speed the of each algorithm both when fitting the algorithm to the training set (Figs. B.1c, B.5c) and when embedding an out-of-sample test set (Figs. B.3c, B.5c). We find that training time increases approximately linearly with the size of the dataset for each algorithm. UMAP (McInnes et

al., 2018) has the fastest training time (approximately as fast as PCA), followed by FIt-SNE (Linderman et al., 2017) and Barnes-Hut-SNE (van der Maaten, 2014), and then VAE-SNE. While VAE-SNE is slower for fitting the training set than both UMAP (McInnes et al., 2018) and t-SNE, it is much faster than the other two neural network methods scvis (Ding et al., 2018) and ivis (Szubert et al., 2019). We also demonstrate that VAE-SNE, and the other neural network methods, can quickly embed out-of-sample test data (Figs. B.3c, B.5c). The time needed for embedding new data is much higher for both t-SNE and UMAP, and while the elapsed time for embedding the test set scales linearly with the number of samples for all algorithms, we also find that it increases with the size of the training set for both UMAP (McInnes et al., 2018) and Barnes-Hut-SNE (van der Maaten, 2014) (Fig. B.3c). This is almost certainly because adding new data for these algorithms requires calculating approximate nearest neighbors between the out-of-sample data and the training set, which consequently requires more computation time for larger training sets. Unexpectedly, FIt-SNE (Linderman et al., 2017) does not exhibit this behavior despite using similar nearest neighbor calculations to Barnes-Hut-SNE (van der Maaten, 2014). On the other hand, VAE-SNE and other deep learning algorithms do not suffer from this limitation. Finally, while we do not comprehensively assess memory complexity of different algorithms in this paper, we stopped our speed comparisons at data subsets with 232,000 ($\times$ 1500 dimensions) observations because UMAP began to cause out-of-memory errors for larger subsets — while all of the other algorithms we tested could still successfully run under the same conditions. This helps to illustrate the key advantage of deep learning-based methods, which naturally maintain very low memory complexity by applying optimization using small batches of data.

3.3.2 Using the likelihood to assess out-of-sample data

Because VAE-SNE also calculates a likelihood score for reconstructing the original high-dimensional data, we can use this to assess performance on out-of-sample data, which is an idea originally proposed by Ding et al. (2018). To test this, we calculate the likelihood score for real data from the posture dynamics dataset (Berman et al., 2016, 2014a; Pereira et al., 2019) and randomly-permuted data (randomized across feature columns) from the same dataset. We find that the likelihood score is reliably lower for the randomized data, and the two likelihood distributions are well separated (Fig. B.6a), which shows this metric could potentially be used to detect outliers. We also compare the entropy of the approximate posterior distribution for

each embedded sample as another potential metric for detecting outliers. While we find that the entropy is much higher for the randomized data, the distribution is highly overlapping with the entropy for the real data (Fig. B.6b), which indicates the entropy may not be as useful for evaluating the embedding quality.

3.3.3 Clustering body posture dynamics to reveal stereotyped behavioral organization

To demonstrate its capabilities as a clustering algorithm, we use VAE-SNE to automatically discretize a dynamical timeseries dataset describing the high-dimensional body posture and behavioral repertoire of 59 freely-behaving fruit flies (*D. melanogaster*; Berman et al. 2016, 2014a; Pereira et al. 2019) — a commonly-used model organism for neuroscience, pharmaceutical, and genetics research. To accomplish this, we use the annotated training data from (Pereira et al., 2019) to train a pose estimation model using deep learning-based software (DeepPoseKit; Graving et al. 2019a). We then use this trained model to automatically track the spatial locations of 10 body parts (head, legs, wings, abdomen) directly from video timeseries data and generate time-frequency spectrograms describing body-part dynamics for each observation in the timeseries (Berman et al., 2014a), which naturally incorporates multi-scale temporal information into each data vector. We then apply VAE-SNE to compress the data to a 30-dimensional latent embedding and simultaneously discretize the dynamical posture timeseries into a set of behavioral clusters. We find that, after optimizing the 30-D VAE-SNE model for 5 repeated trials using the full 21.1 million observation dataset and applying sparse watershed assignment to generate cluster labels (Methods; Fig. 3.1d; Todd et al. 2017), VAE-SNE consistently learns a total of 26 low-level behavioral clusters describing distinct, stereotyped body part movements. We also achieve similar (nearly identical) results when clustering in 10-D and 50-D space and when varying the number of components in the Gaussian mixture prior used for clustering — provided that the number of components is large enough (e.g., $K \geq 100$).

To provide a broad overview of the behavioral structure discovered by VAE-SNE, we manually group these low-level clusters into 6 high-level clusters (Figs. 3.3, B.7; Video S1) by examining video clips sampled from each cluster (Video S2–Video S7) and by calculating and visualizing the mean spectrograms for each low-level cluster to quantify the average distribution of body part movements across frequencies for each behavioral class (Figs. B.8d-f, B.9d-i). These high-level clusters include: locomotion (Video S2), anterior grooming (Video S3), posterior grooming (Video

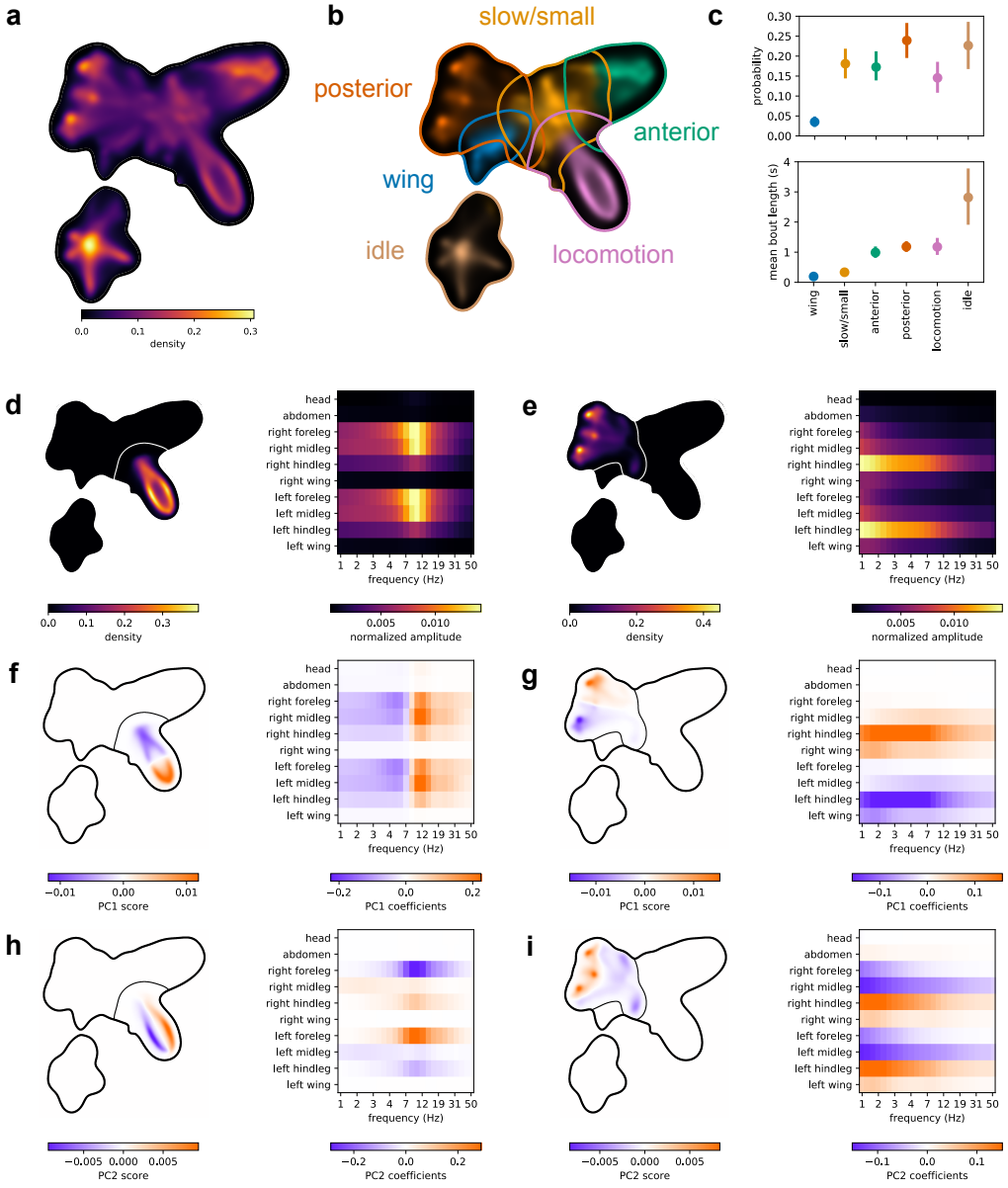

Figure 3.3. Clustering body posture dynamics. **a**, The posterior probability density for the full 21.1 million observation body posture dynamics dataset from Berman et al. (2016, 2014a); Pereira et al. (2019) embedded using a 2-dimensional VAE-SNE model. **b**, The manually-grouped high-level cluster assignments produced using the learned prior from a 30-dimensional VAE-SNE embedding visualized in the 2-D embedding, where contours are the largest 90% probability density contour for each cluster distribution. **c**, Mean and 95% bootstrap intervals of the marginal (stationary) probability and mean bout length for each high-level cluster (n = 59 per cluster). **d**-**i**, Visualizations describing the high-level locomotion (**d**,**f**,**h**; Video S2; Fig. B.8) and posterior grooming (**e**,**g**,**i**; Video S4; Fig. B.9) clusters. **d-e**, The 2-D posterior probability density for each cluster (left) and the mean spectrogram for each cluster (right). **f**-**i**, The principal component scores for the two largest components of the spectrograms assigned to each cluster visualized within the 2-D embedding (left), and the eigenvector coefficients describing the linear contribution of each spectrogram feature (right) for the principal component score.

S4), wing movements (Video S5), small/slow leg movements (Video S6), and idle behavior (Video S7). Many of the low-level clusters (10 clusters in total) describe distinct slow/small leg movements, while there are 3 low-level clusters for locomotion (Fig. B.8), 3 for anterior grooming, 6 for posterior grooming (Fig. B.9), 2 for wing movements, and 2 for idle behavior. Videos and posture timeseries data sampled from each cluster also clearly demonstrate the stereotypy of behaviors within these behavioral classes, which matches well with previous work describing these dynamics (Berman et al., 2016, 2014a; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; Pereira et al., 2019). Additionally, the principal components of the spectrograms from each high-level cluster (Fig. 3.3f-i; Fig. B.7d-i) reveal continuous variation related to asymmetrical body movements and differences in peak movement frequency. We calculate basic statistics describing cluster usage across individuals (Figs. 3.3c, B.8c, B.9c) including the marginal (stationary) probability of behavioral classes and the mean bout length, or the average amount of time a behavior is performed when an individual transitions into that cluster. In particular, the low probability and short bout length for wing movements and short bout length for slow/small leg movements (Fig. 3.3c) indicate these clusters may be transitional or idiosyncratic behaviors (Todd et al., 2017). For the low-level locomotion clusters (Fig. B.8) we also calculate the forward component of the leg movement velocity (in body lengths per second, or $\mathrm{BL} \cdot \mathrm{s}^{-1}$) relative to the egocentric orientation of the animal. We then use the forward velocity to classify each leg in the timeseries as “swing" (forward velocity $> 0\,\mathrm{BL} \cdot \mathrm{s}^{-1}$) or “stance" (forward velocity $\leq 0\,\mathrm{BL} \cdot \mathrm{s}^{-1}$) and find that our low-level locomotion clusters show signatures of distinct locomotory gaits (i.e., tetrapod and tripod gaits; Mendes et al. 2013; Pereira et al. 2019) with different numbers of legs being used for walking, on average, within each cluster. Together these results demonstrate that VAE-SNE is able to automatically decompose the dynamics of known complex behaviors (Video S1).

Due to the many philosophical complexities of objectively evaluating unsupervised cluster representations (reviewed by Jain et al. 1999; Kleinberg 2003; Todd et al. 2017), we forgo any further quantitative assessment of our clustering results and instead leave this for future work. For example, it is unclear how to best select the number of clusters for many different algorithms; how to properly compare algorithms that naturally produce different numbers of clusters and cluster shapes; and what metric(s) should be used to meaningfully evaluate a clustering description as generally “good" or “useful" other than manual, qualitative validation of the results, which we already provide here — though several quantitative descriptors with

varying levels of desirability have been recently proposed for behavioral data (Todd et al., 2017). Comparing unsupervised cluster labels with a priori-defined labels — as is common practice (e.g., Guo et al. 2017; Jiang et al. 2016; Luxem et al. 2020; Xie et al. 2016; Yang et al. 2019) — is also problematic, as human-supervised descriptions may not accurately capture the underlying structure of the data distribution, and this is especially true for datasets where the goal is to potentially discover subtle differences that are undetectable by humans (e.g., Wiltschko et al. 2015). Despite the limitations imposed by these complexities, our results still illustrate multiple useful features of VAE-SNE as a general-purpose method.

Overall, we demonstrate how VAE-SNE can be used as a practical, scalable, and flexible tool for clustering real-world high-dimensional data. In this case, we transform posture data into interpretable behavioral labels that are comparable to those from previous methods (Berman et al., 2016, 2014a; Cande et al., 2018; Klibaite et al., 2017; Klibaite & Shaevitz, 2019; Pereira et al., 2019; Todd et al., 2017). However, in contrast to many of these existing methods, VAE-SNE performs dimension reduction and clustering simultaneously, and unlike most previously-described algorithms for clustering data (e.g., Guo et al. 2017; Jiang et al. 2016; Xie et al. 2016; Yang et al. 2019), our method learns a small set of decipherable classes without the need to carefully tune the number of clusters fitted to the data, which can often be a non-trivial, unintuitive, and computationally-intensive process (Fang & Wang, 2012; Milligan & Cooper, 1985; Pham et al., 2005; Todd et al., 2017). Instead, any arbitrarily large number will give similar results due to the sparse watershed assignment procedure we use to combine overlapping clusters (Methods; Fig. 3.1d; Todd et al. 2017). In contrast to methods that impose strong assumptions about cluster shape, our clustering method has relaxed assumptions and allows for more complex (e.g., non-convex) cluster distributions based on the local structure of the data. Additionally, in comparison to prior methods for unsupervised behavioral analysis, VAE-SNE has the advantage of being able to use more than two dimensions for clustering data, which has been shown to provide higher-quality behavioral labels with many potentially-desirable properties (Todd et al., 2017). Finally, our results further show that there is no need to carefully select a subset of data to use for training (e.g., the importance sampling technique described by Berman et al. 2014a), which can also be a time-consuming process. Instead, VAE-SNE can be readily applied to large datasets that cannot fit into memory while still successfully detecting relatively short-lived and infrequent types of behavior, such as wing movements (Fig. 3.3b-c; Video S5).

3.4 Discussion

Here we introduce VAE-SNE, a deep generative model for simultaneously reducing dimensionality and clustering data. We compare VAE-SNE to existing methods for dimensionality reduction and demonstrate its utility and versatility using real-world examples. Our results establish that VAE-SNE is able to generate robust and interpretable compressed representations for data from different domains and is comparable in performance to other nonlinear methods for dimensionality reduction. In contrast to these existing methods, VAE-SNE has the advantage of being able to automatically cluster similar observations into a small set of classes, which can then be used to summarize large datasets with coarse-grained descriptors or select specific subpopulations of data for more detailed analysis. Our approach can also readily scale to very large datasets by leveraging techniques from deep learning — including, and especially, out-of-core data that cannot fit into memory. However, despite these strengths, VAE-SNE still has important limitations depending on the goals of the user, and there are many ways in which the model could be improved or extended in subsequent iterations. There are also other domains that VAE-SNE could be applied to in the future.

VAE-SNE preserves local relationships while also minimizing global structure distortion. Additionally, while VAE-SNE is not explicitly an autoregressive model, it still preserves a good deal of high-dimensional timeseries information. However, our results also show that VAE-SNE, and most of the other dimension reduction methods we tested, does not accurately preserve fine-scale structure (neighborhoods $<$1% of the total embedding size). For many applications, preserving these details may be unimportant, but this structure has been shown to be useful for detecting infrequent types of data, such as rare cell types (Linderman et al., 2019). Therefore, our results suggest that if researchers wish to preserve this type of information they should use FIt-SNE (Linderman et al., 2017, 2019) or Barnes-Hut-SNE (van der Maaten, 2014) over other algorithms for dimension reduction. We also find that, when initialized with PCA over the default initialization, UMAP (McInnes et al., 2018) preserves global structure slightly better without noticeably affecting local structure preservation, so PCA may be a more advantageous choice for initializing UMAP embeddings.

VAE-SNE optimizes faster than existing deep learning methods for dimensionality reduction, but FIt-SNE (Linderman et al., 2017, 2019), Barnes-Hut-SNE (van der Maaten, 2014), and UMAP (McInnes et al., 2018) are still faster. However, the training time for deep-neural-network

methods like VAE-SNE and ivis (Szubert et al., 2019) can be variable due to the use of early stopping criteria that automatically end training when no improvement in the objective function is detected. These early stopping criteria could be easily adjusted to further shorten (or lengthen) training time. While we did not assess performance during the optimization process, much of the training time for VAE-SNE is spent on minor improvements to the objective function, which indicates adequate results can also be achieved with less training time. Additionally, FIt-SNE (Linderman et al., 2017, 2019), Barnes-Hut-SNE (van der Maaten, 2014), and UMAP (McInnes et al., 2018), are much slower for embedding new data because they calculate nearest neighbors for the new data and further optimize the embedding, which VAE-SNE does not require due to its learned encoder function. For smaller datasets that can fit in memory FIt-SNE (Linderman et al., 2017, 2019), Barnes-Hut-SNE (van der Maaten, 2014), and UMAP (McInnes et al., 2018) are still attractive options for dimensionality reduction, but for datasets that do no fit into memory, VAE-SNE provides some distinct advantages.

VAE-SNE has the ability to detect outliers and assess the embedding quality for out-of-sample data. This provides a straightforward mechanism for identifying new data to include in the training set, which can further improve performance. Most of the other algorithms we tested, or at least the specific software implementations we tested, provide no mechanism for quantitatively assessing embedding quality for each observation — with outliers being simply embedded under the assumption that the data are well supported by the training distribution. This can cause problems for any downstream analysis, especially when using statistical tests to answer scientific questions. Further improvements for outlier detection might include the use of Bayesian inference (Hafner et al., 2018) or other methods for estimating predictive uncertainty (reviewed by Kendall & Gal 2017).

We demonstrate that results produced by VAE-SNE can serve as a highly-interpretable coarse-grained description of tens-of-millions of observations — with several advantages over existing methods for clustering data. Applying VAE-SNE to future research in the behavioral sciences could help to reveal the genetic, environmental, and neural underpinnings of animal behavior (Berman, 2018; Brown & De Bivort, 2018; Datta et al., 2019) — especially when combined with recent advances in behavioral measurement (Graving et al., 2019a; Günel et al., 2019; A. Mathis et al., 2018; Pereira et al., 2019) as well as genetic (Doudna & Charpentier, 2014; Ran et al., 2013), sensory (Stowers et al., 2017), and neural (Bath et al., 2014; Cande et al., 2018) manipulations. The clustering capabilities of VAE-SNE could also be applied to other

types of data, such as single-cell RNA-seq data (Ding et al., 2018; La Manno et al., 2018) and natural history images (J. F. H. Cuthill et al., 2019; Q. Zhang et al., 2019), but we leave this as future work for other researchers and domain experts to explore and validate. VAE-SNE might also be further improved by the use of more complex hierarchical clustering distributions (Razavi et al., 2019; Roberts et al., 2018; Tomczak & Welling, 2017), where additional scales with finer- or coarser-grained descriptions can be selected from the model for post-hoc analysis. Recent work has also shown that iteratively adjusting the parameters of the t-SNE similarity kernel can be used to generate a hierarchy of clusters in the latent embedding (Robinson & Pierce-Hoffman, 2020), which could be potentially applied to VAE-SNE as well.

To demonstrate the flexibility of VAE-SNE as a deep learning model, we introduce a variant for embedding data in polar coordinates on a unit sphere (Appendix B.3.1). We find that VAE-SNE successfully preserves structure in a spherical embedding as well (Fig. B.10; Video S8; Video S9), which may be a more natural way to model some high-dimensional data sets (Davidson et al., 2018) since it avoids the “crowding" problem common to other embedding methods (Ding & Regev, 2019; van der Maaten & Hinton, 2008). While we focus on the Euclidean and cosine distances for calculating local neighborhoods, any differentiable distance function could potentially be substituted to create different embedding geometries, and, while we focus on kernels from the location-scale family of probability distributions (i.e. Gaussian, Student’s *t*), other log probability functions could potentially be used as well.

We also introduce a convolutional version of VAE-SNE for embedding images directly from raw pixel data (Appendix B.3.2). After applying this model to natural history images, we find that it groups perceptually-similar images based on complex sets of image features that correspond with taxonomic groupings (Figs. B.11, B.12). These results indicate that convolutional VAE-SNE may be useful for tasks such as relating distributions of complex animal coloration patterns to ecological, evolutionary, and behavioral function (I. C. Cuthill et al., 2017; J. F. H. Cuthill et al., 2019; Ezray et al., 2019; Wham et al., 2019). Future applications might include applying VAE-SNE to audio data (e.g., Oord et al. 2016; Sainburg et al. 2019).

There are multitude of ways in which VAE-SNE could be further improved or extended. Naturally, future work could apply more recent advances in variational and probabilistic inference like normalizing flows (Kingma et al., 2016; Papamakarios et al., 2017; Rezende & Mohamed, 2015), which allow data to be modeled with a more direct invertible mapping from the latent posterior to the data distribution, while also employing flexible, arbitrarily-complex distributions.

The latent distribution used for VAE-SNE could also be modeled using many other types of representations such as quantized (Van Den Oord et al., 2017) or categorical (Jang et al., 2016; Maddison et al., 2016) distributions. Recent progress in generative adversarial networks (GANs; Goodfellow et al. 2014), may also provide further enhancements for modeling complex feature dependencies within the data distribution (Dieng, Ruiz, et al., 2019; Larsen et al., 2016; Srivastava et al., 2017). Timeseries data could be explicitly modeled using autoregressive deep neural networks (e.g., Oord et al. 2016) for the encoder and decoder similar to M. Johnson et al. (2016); Luxem et al. (2020); Markowitz et al. (2018); Pandarinath et al. (2018); Sussillo et al. (2016); Wiltschko et al. (2015), and the latent distribution can be optimized to accurately predict future observations, which has been shown to be a useful framework for modeling behavior (Berman et al., 2016; Luxem et al., 2020). Additionally, computational efficiency might be further improved by applying recent advances in metric (Sohn, 2016) and contrastive learning (T. Chen et al., 2020), which may reduce or eliminate the need to perform expensive pairwise computations. Recent work on density-preserving versions of t-SNE and UMAP (Narayan et al., 2020) could also be incorporated to further improve the embedding quality.

Explicitly modeling hierarchical structure caused by variance across individual trials and subjects (Pandarinath et al., 2018) and batch effects due to variance in sampling procedures (Ding & Regev, 2019) is also important for improving VAE-SNE in the future. These effects could be accounted for with more complex, hierarchically-parameterized models (Pandarinath et al., 2018; Sussillo et al., 2016), hierarchical latent distributions (Razavi et al., 2019; Roberts et al., 2018; Tomczak & Welling, 2017), and new similarity kernels — such as the conditional t-SNE kernel recently proposed by Kang et al. (2019). The general use of conditional (e.g., Van den Oord et al. 2016) or supervised (e.g., Alemi et al. 2016) labels when optimizing the model could also help to integrate additional prior information about the data distribution into the latent distribution, the latter of which is already a feature of both UMAP (McInnes et al., 2018) and ivis (Szubert et al., 2019).

In summary, VAE-SNE is a general-purpose deep learning model for both dimension reduction and clustering that can be applied to many different types of data and readily scales to large datasets. Together our results illustrate that it is a robust, feature-rich method with multiple distinct advantages that make it an effective tool for analyzing real-world datasets across disciplines.

3.5 Methods

3.5.1 The VAE-SNE model

VAE-SNE is a variational autoencoder (VAE; Appendix B.1.1) with a learned Gaussian mixture prior (Dilokthanakul et al., 2016; Kingma et al., 2014; Tomczak & Welling, 2017) that is optimized using the ELBO objective function (derived in Appendix B.1.2) with an additional local neighborhood regularizer (Ding et al., 2018; Hinton & Roweis, 2003; van der Maaten, 2009; van der Maaten & Hinton, 2008). The likelihood and divergence terms from the ELBO objective can be broadly considered as an information theoretic trade-off between reconstruction accuracy (distortion) and compression (rate) respectively (Alemi et al., 2016, 2017; Chalk et al., 2016), which makes VAEs an attractive solution for dimensionality reduction. However, there are implicit problems with the ELBO objective (reviewed by Alemi et al. 2017; Dieng, Kim, et al. 2019) that may prevent the model from learning a useful latent representation — e.g., a powerful, overparameterized decoder can simply ignore the compressed latent codes but still produce high-quality reconstructions. These issues render VAEs problematic as a general method for reducing dimensionality, as the primary purpose of dimensionality reduction is to create compressed representations that preserve important statistical features of the original data distribution.

Regularizing the ELBO to improve structure preservation

We address the problems outlined above by optimizing VAE-SNE with a regularized version of the ELBO. This modification introduces a pairwise similarity regularizer derived from the (*t*-distributed) stochastic neighbor embedding (SNE/t-SNE) objective (Hinton & Roweis, 2003; van der Maaten, 2009; van der Maaten & Hinton, 2008). This idea of using the SNE objective for regularizing the latent space of VAEs was first proposed by Chien & Hsu (2017), which they called variational manifold probabilistic linear discriminant analysis (vm-PLDA), and later independently proposed by Ding et al. (2018) with their scvis model. However, the idea of applying the SNE objective to autoencoders, and deep neural networks in general, was introduced much earlier by van der Maaten (2009) with parametric t-SNE (pt-SNE), who proposed to use this objective in conjunction with an autoencoder to jointly learn a latent embedding. The pt-SNE model (van der Maaten, 2009) was also recently combined with advances from the Barnes-Hut-SNE algorithm (van der Maaten, 2014) under the name net-SNE (Cho et al., 2018). Additionally, Moody (2017)

developed one of the first publicly-available pieces of software to combine the SNE objective with variational inference (variational t-SNE, or vt-SNE; and topic-SNE) but did not use a deep neural network to amortize inference across a set of shared parameters. Im et al. (2018) also proposed a variational bound on the t-SNE objective to improve optimization.

Here we apply the SNE objective to a VAE in a similar fashion to Ding et al. (2018). That is, we use the SNE objective as a method of better preserving structure in the latent embedding produced by our VAE, which improves the usefulness of the compressed representation (approximate posterior) produced by the ELBO. When combined into a single objective, we call this the stochastic neighbor evidence lower bound, or SNELBO. Generalizing from Ding et al. (2018), given a high-dimensional data matrix $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and model parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$, the SNELBO objective is written as:

$$\underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\arg\min} -\mathrm{SNELBO}(\mathbf{X}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \underset{\boldsymbol{\theta},\boldsymbol{\phi}}{\arg\min} -\frac{1}{N}\sum_i \mathrm{ELBO}_i(\mathbf{x}_i, \boldsymbol{\theta}, \boldsymbol{\phi}) - \alpha\,\mathrm{SNE}_i(\mathbf{X}, \boldsymbol{\phi}) \tag{3.1a}$$

$$\mathrm{ELBO}_i(\mathbf{x}_i, \boldsymbol{\theta}, \boldsymbol{\phi}) = \gamma\,\mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)}[\underbrace{\log p_{\boldsymbol{\theta}}(\mathbf{x}_i|\mathbf{z}_i)}_{\text{distortion}}] - \beta\,\underbrace{\mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)\|p_{\boldsymbol{\theta}}(\mathbf{z})]}_{\text{rate}} \tag{3.1b}$$

$$\mathrm{SNE}_i(\mathbf{X}, \boldsymbol{\phi}) = \mathbb{E}_{\substack{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) \\ \mathbf{z}_j \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_j)}} \left[\sum_j \mathrm{SNE}_{j|i}(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\phi})\right] \tag{3.1c}$$

$$= \mathbb{E}_{\substack{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) \\ \mathbf{z}_j \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_j)}} \underbrace{\left[\sum_j \mathbb{KL}[p(\mathbf{x}_j|\mathbf{x}_i)\|q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)]\right]}_{\text{pairwise similarity}} \tag{3.1d}$$

for $i, j = 1, \ldots, N$ and $i \neq j$, where N is the number of observations in the $N \times M$ matrix $\mathbf{X} \in \mathbb{R}^M$. Thus vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ are the ith and jth row in $\mathbf{X}$, while $\mathbf{z}_i$ and $\mathbf{z}_j$ are Monte Carlo samples from the approximate low-dimensional posterior $\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)$ and $\mathbf{z}_j \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_j)$ respectively (Eq. B.3c) — sampled using the reparameterization trick from Kingma & Welling (2013), or $\mathbf{z}_i = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$, where ϵ is an auxillary noise variable $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $\odot$ is the element-wise product (see Appendix B.1.3 for further discussion).

The objective function (Eq. 3.1a) consists of three terms, which can be interpreted as follows: (**1**) the expected log likelihood of the decoder distribution (Eq. 3.1b; distortion) minimizes distortion between the observed ground truth $\mathbf{x}_i$ and reconstruction, or maximizes accuracy, and preserves global structure in the embedding; (**2**) the divergence between the approximate posterior and the prior distribution (Eq. 3.1b; rate) constrains the global coordinate space of the

embedding and restricts the rate of information (relative to the prior) that can be transmitted through the compressed space; and (**3**) the expected divergence between pairwise similarities (Eq. 3.1d) in high-dimensional space $p(\mathbf{x}_j|\mathbf{x}_i)$ and those in low-dimensional space $q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)$ acts as a regularizer to preserve local neighbor relationships between data points. Further details of this stochastic neighbor regularizer are derived in Appendix B.2.

The Lagrange multipliers γ, β, and α are used to weight the distortion, rate, and pairwise similarity terms respectively, which we include as hyperparameters for the model. These multipliers can be adjusted to produce different forms of the objective for optimizing the model — e.g., increasing or decreasing the rate with the β multiplier (Higgins et al., 2017) — but in practice we set $\gamma = \beta = 1$, while α is set (following Ding et al. 2018) to the dimensionality of the data $\alpha = M$ to match the distortion term, which scales with the size of the input, or $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \sum_{m=1}^{M} \log p_{\boldsymbol{\theta}}(x_m|\mathbf{z})$.

Learning a Gaussian mixture prior

For optimizing the VAE-SNE objective (Eq. 3.1a), we use a learned, or empirical, Gaussian mixture prior for $p_{\boldsymbol{\theta}}(\mathbf{z})$ which allows for an arbitrarily complex distribution (similar to Dilokthanakul et al. 2016; Kingma et al. 2014; Tomczak & Welling 2017). Using a more complex distribution allows for a tighter bound on objective, and, after optimization, approaches the true posterior distribution as the complexity of the distribution is increased (Cremer et al., 2017; Dilokthanakul et al., 2016; Kingma et al., 2014; Tomczak & Welling, 2017). The Gaussian mixture distribution is written as the weighted mixture of K Gaussian components:

$$p_{\boldsymbol{\theta}}(\mathbf{z}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_k, \mathbf{I}). \tag{3.2}$$

The mean $\boldsymbol{\mu}_k \in \mathbf{M}$ and mixture weight $\omega_k \in \boldsymbol{\omega}$ of each component are learned as model parameters $\{\mathbf{M}, \boldsymbol{\omega}\} \in \boldsymbol{\theta}$ subject to a softmax normalization constraint $\sum_{k=1}^{K} \omega_k = 1$. We also regularize the prior distribution by minimizing the divergence between the mixture distribution used to weight each component and a maximum-entropy mixture distribution, or:

$$\arg\min_{\boldsymbol{\omega}} \sum_{k=1}^{K} \omega_k \log \omega_k + \omega_k \log K. \tag{3.3}$$

This prevents the prior from degenerating to a small number of modes (a problem described in more detail by Dilokthanakul et al. 2016; Kingma et al. 2014) by increasing the entropy of the mixture distribution. A higher entropy mixture distribution forces to model to utilize more of the components within the distribution, which increases the number of clusters and, consequently, the level of detail of the final clustering description (Still & Bialek, 2004). An analogous maximum entropy regularizer was also recently applied to solve the long-standing mode collapse problem common to generative adversarial networks (GANs; Dieng, Ruiz, et al. 2019).

The covariance for each component distribution could be learned as free parameters, but we find that using a simpler identity covariance matrix $\mathbf{I}$ allows for a sufficiently expressive prior distribution without adding additional complexity — and is less prone to cluster degeneracy during optimization. Using a highly-flexible (i.e., $K \gg 1$) learned distribution as the prior for the latent space allows for better structure preservation, as non-convex structures are not distorted by the use of an overly simple prior. Also note that the special case of $K = 1$ mixture component is equivalent to the standard VAE prior (Kingma & Welling, 2013), or $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$, which is the prior used by Ding et al. (2018).

Calculating the rate loss term The parameters for the Gaussian mixture prior $\{\mathbf{M}, \boldsymbol{\omega}\} \in \boldsymbol{\theta}$ are then learned from the data via the rate term in the VAE-SNE objective (Eq. 3.1b). For the special case of $K = 1$ we compute the Kullback-Leibler divergence analytically; however, because there is no analytical solution for a Gaussian mixture distribution with $K > 1$, we instead approximate this term numerically using Monte Carlo integration. In this case we use the expected log-density ratio for calculating the rate (Appendix B.1.2), which is written as:

$$\mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) \| p_{\boldsymbol{\theta}}(\mathbf{z})] = \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)}{p_{\boldsymbol{\theta}}(\mathbf{z})} \, d\mathbf{z} \tag{3.4a}$$

$$= \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)} \left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)}{p_{\boldsymbol{\theta}}(\mathbf{z}_i)} \right] \tag{3.4b}$$

$$= \mathbb{E}_{\mathbf{z}_i \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)} \left[\log q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i) - \log p_{\boldsymbol{\theta}}(\mathbf{z}_i) \right]. \tag{3.4c}$$

Clustering data with the Gaussian mixture prior After optimizing the parameters for the prior, we can then use the learned Gaussian mixture to assign embedded data to discrete clusters. In other words, we wish to calculate the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})$, where $\mathbf{y}$ is a vector of class labels, or $\mathbf{y} = \{y_1, y_2, \ldots, y_K\}$. However, the Gaussian mixture prior can contain

highly-overlapping component distributions, which can cause undesirable side-effects. On one hand, this renders the parameterized mode for each overlapping component an unreliable descriptor of the surrounding local density, as each component is then simply a degenerate sub-mode within a non-Gaussian density cluster rather than a distinct subpopulation within the distribution delineated by the structure of the data. On the other hand, a Gaussian mixture distribution can have any arbitrary arrangement of weighted components, which makes the task of directly calculating the true local density mode for each embedded point both analytically and numerically intractable. Therefore, to circumvent these problems, we apply the sparse watershed assignment procedure described by Todd et al. (2017) to find the true local maximum for each component in the distribution — rather than for every embedded observation — through numerical optimization, which requires only a nominal amount of additional computation. We can then merge overlapping components and assign embedded data to a mode that more accurately reflects the underlying (potentially non-Gaussian) region of local density.

Because this sparse watershed procedure produces clusters with an arbitrary number of weighted components, calculating the full posterior probability $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})$ for each data point is computationally complex. So for the sake of simplicity, we perform hard label assignment. In other words, we calculate the mode of the cluster distribution for each value of $\mathbf{z}$, or:

$$l_i = \arg\max_l p_{\boldsymbol{\theta}}(y_l|\mathbf{z}_i), \tag{3.5}$$

for $l = 1, \ldots, K$, where l_i is the assigned label for the latent vector $\mathbf{z}_i$. This hard label assignment procedure is performed in 3 steps: (**1**) latent vectors are initially assigned to the nearest (highest local density) component in the Gaussian mixture prior; (**2**) the Gaussian mixture distribution is further optimized to combine overlapping mixture components using sparse watershed assignment (Todd et al., 2017); and (**3**) the initial cluster assignments are then recursively updated using the learned hierarchy of overlapping components to ensure each latent vector is assigned to the mode that best represents the underlying density of the local neighborhood for that observation. To accomplish these steps, the expected value of the approximate posterior for each data point is initially assigned to a single mode in the Gaussian mixture distribution by calculating the weighted mixture component with the maximum likelihood (minimum distortion), which is written as:

$$k_i = \arg\max_k \omega_k \mathcal{N}(\mathbb{E}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i)]|\boldsymbol{\mu}_k, \mathbf{I}), \tag{3.6}$$

where k_i is the initial cluster assignment for the ith data point $\mathbf{x}_i$. We then combine degenerate (highly-overlapping) modes from the distribution by applying the sparse watershed procedure described by Todd et al. (2017). Using this procedure, the initial cluster assignments are further combined by optimizing the mean of each component to ascend to its local maximum within the Gaussian mixture prior, which we write as a minimization of the negative log-likelihood, or:

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} -\frac{1}{K}\sum_{k=1}^{K} \log \sum_{l=1}^{K} \omega_l \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{\mu}_l, \mathbf{I}), \tag{3.7}$$

where $\boldsymbol{\mu}_k^* \in \mathbf{M}^*$ is the optimized mean of each component. We optimize this objective numerically with the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1×10^{-3} until the objective (Eq. 3.7) stops improving for 100 training steps. We then merge cluster assignments based on whether the mode for the initial cluster assignment k_i has moved within the basin of attraction for another mixture component in the distribution (after optimizing Eq. 3.7), or:

$$l_i = \arg\max_l \omega_l \mathcal{N}(\boldsymbol{\mu}_{k_i}^*|\boldsymbol{\mu}_l, \mathbf{I}) \tag{3.8}$$

where l_i is the sparse watershed label assignment for the ith data point $\mathbf{x}_i$, which was assigned to the k_ith mode of the distribution $\boldsymbol{\mu}_{k_i}$ in the initial cluster assignment step (Eq. 3.6). We then repeat this assignment procedure K times to ensure all label assignments to degenerate modes are reassigned to the mode with the highest local density:

$$l_i \coloneqq \arg\max_l \omega_l \mathcal{N}(\boldsymbol{\mu}_{l_i}^*|\boldsymbol{\mu}_l, \mathbf{I}) \text{ for } k = 1, \ldots, K. \tag{3.9}$$

Note that, for data assigned to non-degenerate modes in the initial step, typically the cluster assignment remains unchanged, where $l_i = k_i$.

3.5.2 Comparing dimensionality reduction algorithms

We compared VAE-SNE to other dimensionality reduction algorithms including PCA (scikit-learn v0.23.0; Pedregosa et al. 2011), t-SNE (van der Maaten & Hinton, 2008), UMAP (v0.4.0; McInnes et al. 2018), scvis (Ding et al., 2018), and ivis (v1.7.2; Szubert et al. 2019). Our main comparisons involve compressing data to two dimensions for visualization purposes, but VAE-SNE (and other

algorithms) can be used for dimensionality reduction more generally.

openTSNE and t-SNE variants

For t-SNE we used the openTSNE (v0.4.0) implementation from Poličar et al. (2019), which includes improvements from Linderman et al. (2017, 2019); van der Maaten (2014) to maximize speed and scalability, as well as methods for embedding out-of-sample data described by Poličar et al. (2019) (see also Berman et al. 2014a; Kobak & Berens 2019). We tested two versions of openTSNE using both the Barnes-Hut approximation (Barnes-Hut-SNE) from van der Maaten (2014) and the Fourier interpolation approximation (FIt-SNE) from Linderman et al. (2017, 2019). However, FIt-SNE, the fastest version of openTSNE, is practically limited to very low dimensional embeddings (i.e., 1-D or 2-D) due to the Fourier interpolation algorithm used for approximating the gradient during optimization, and therefore cannot be used for more general-purpose dimensionality reduction (Linderman et al., 2017, 2019).

scvis as a special case of VAE-SNE

We found the original implementation of scvis (Ding et al., 2018) difficult to use for our comparisons without extensive modification, as it relies on outdated software dependencies and is limited to specific data file formats for using the code. However, scvis (Ding et al., 2018) can be considered a special case of VAE-SNE with specific hyperparameter settings, so instead we used VAE-SNE with hyperparameters matched to those described by Ding et al. (2018) for making comparisons. In particular, we used the network architecture for the encoder and decoder networks described by Ding et al. (2018), along with ELU activations (Clevert et al., 2015). We also use the asymmetric similarity kernel for the high-dimensional similarities (Eq. B.8a), and we set $K = 1$ for the number of components in the prior distribution (Eq. 3.2). For benchmarking the processing speed of scvis (Ding et al., 2018), we disabled our added parallel computations (Section 3.5.5) to match the speed of the original implementation from Ding et al. (2018), and we calculated training time based on the original recommendation from Ding et al. (2018) for training with batch size of 512 for 100 epochs.

Setting hyperparameters for comparisons

For each algorithm we used Euclidean distances for calculating pairwise similarities (the default for all of the algorithms tested) along with the default settings for all other hyperparameters with some exceptions. For t-SNE, we set `n_jobs=-1` to enable parallel processing. For UMAP, we also compare PCA initialization for the low-dimensional embedding (vs. the default Laplacian Eigenmap initialization), which is not a default option but improves global structure preservation. For ivis (Szubert et al., 2019), we used the default model and followed recommendations from Szubert et al. (2019) to adjust the early stopping criteria for different dataset sizes.

The hyperparameters for different methods could, of course, be adjusted ad infinitum to produce different types of embeddings and could bias performance for different datasets in many ways; however, the comparisons we make in this paper are not meant to be exhaustive, only informative in terms of validating VAE-SNE as a comparable method. In the end, researchers will have to decide for themselves which algorithm is most useful for their specific application. It is also worth considering that, for some of the algorithms tested, adjusting the hyperparameters can dramatically alter computational and memory requirements — for example, increasing the perplexity hyperparamater for FIt-SNE (Linderman et al., 2017) and Barnes-Hut-SNE (van der Maaten, 2014) or the `n_neighbors` hyperparameter for UMAP, increases number of nearest neighbors that are computed and, consequently, the size of the nearest neighbors graph used to optimize the embedding. Our decision to use default settings is also especially reasonable for the t-SNE variants we tested given that the openTSNE package (Poličar et al., 2019) uses hyperparameter suggestions from Kobak & Berens (2019), which have been empirically shown to work well across many datasets.

VAE-SNE hyperparameters

We tested multiple variants of VAE-SNE in our comparisons, but across these variants we use similar hyperparameters for training. For the encoder and decoder networks we use 4 densely-connected layers each with 256 units (with biases). For each layer we apply the nonlinear SELU activation function and use the appropriate random initialization for the weights described by Klambauer et al. (2017). We train each VAE-SNE model for a maximum of 100 epochs with an initial batch size of 512 using the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001. For the perplexity hyperparameter, we calculate this as a function of the batch size used

during training, which we call the *perplexity ratio*, such that $\mathrm{P} = b\varrho$ where P is the perplexity, b is the batch size, and ϱ is the perplexity ratio. To improve global structure preservation, we begin training with $\varrho = 0.1$ and then anneal to $\varrho = 0.01$ by exponentially decaying ϱ after each training batch (similar to the perplexity annealing technique described by Kobak & Berens 2019). After the perplexity ratio is fully annealed to the target value, we then perform early stopping if pairwise similarity loss stops improving by at least 0.001 per epoch with a patience of 5 epochs (lack of progress is ignored for 5 epochs before stopping training). While it is common practice to decrease the learning rate after training stagnates to further improve performance, we instead increase the batch size, which has been shown to provide similar improvements (Smith et al., 2017). Therefore after training stagnates and early stopping is initiated for the initial batch size of 512, we increase the batch size to 1024 and continue training until early stopping is initiated again using the same criteria. For the Gaussian mixture prior we set the number of components to $K = 100$, but we found that any arbitrarily large number of components produced similar (nearly identical) results.

We tested 4 variants of VAE-SNE with different similarity kernels. We tested VAE-SNE using a t-SNE similarity kernel with (**1**) constant kernel parameters ($\nu = \tau = 1$) as well as (**2**) learned kernel parameters (van der Maaten, 2009). We also tested VAE-SNE variants using a SNE kernel with (**3**) constant ($\eta = 1$) and (**4**) learned parameters as well. Otherwise the hyperparameters for each variant were kept constant, as described above.

Local structure preservation

After embedding the data with each algorithm we assessed local structure preservation with two measures of preservation that define local neighborhoods in different ways. For both of these metrics we targeted neighborhoods that correspond to $\sim 1\%$ of the total embedding size.

metric-based neighborhoods First, we used a metric-based measure of local neighborhood preservation, where neighborhoods are defined based on distance (a fixed radius) to a cluster center. Following Becht et al. (2019) we applied the k-means clustering algorithm (with $k = 100$ clusters; using scikit-learn v0.23; Pedregosa et al. 2011) to the high-dimensional data and the low-dimensional embedding for each method, which effectively divides the data into small Voronoi regions. We then calculated the normalized mutual information (reviewed by Vinh et al. 2010; see also McDaid et al. 2011) between the high-dimensional and low-dimensional cluster assignments

(using scikit-learn v0.23; Pedregosa et al. 2011). This provides a symmetric and permutation invariant measure of how well local neighborhood memberships from the high-dimensional space are preserved by each embedding method — with similarity ranging from 0 (no overlap, or random) to 1 (perfect overlap). We performed 5 replicates of this for each trial.

topological neighborhoods Second, we assessed local neighborhood preservation topologically by calculating the exact nearest neighbors for 1000 randomly selected data points and then defining the local neighborhood for each point as k nearest neighbors, where k is selected such that $\frac{k}{N} \approx 0.01$, and N is the total embedding size. We then computed the proportion of the neighbors that are assigned to the correct local neighborhood in low-dimensional embedding, which ranges from 0 (no neighbors preserved) to 1 (all neighbors preserved). We performed 5 replicates of this for each trial.

Global structure preservation

To assess global structure preservation we follow Becht et al. (2019) by calculating the Pearson correlation between pairwise squared Euclidean distances for 10,000 points in the high-dimensional space and the low-dimensional embedding for each method (for a total of 49.995 million distances). As distances have a lower bound of zero and tend to follow a log-normal (or Gamma) distribution, we first log transformed the distances in order to homogenize the variance and better match the assumptions of Pearson's correlation score. The Pearson correlation then provides a measure of the global structure preservation ranging from -1 (anti-correlated) to 1 (correlated). We performed 5 replicates of this for each trial.

Fine-scale structure preservation

Because our metrics for local structure preservation only account for a single scale but not the fine-scale structure within local neighborhoods, we also assessed topological structure preservation for smaller neighborhood sizes. As before, we calculated the exact nearest neighbors for 1000 randomly selected data points. We then computed the proportion of points assigned to the correct neighborhood across 14 dyadically ($\log_2$) spaced neighborhood sizes ranging from $k = 2^1$ to $k = 2^{14}$. Neighborhood sizes were then normalized as a proportion of the total embedding size, or $\frac{k}{N}$. We performed 5 replicates of this for each trial and neighborhood size.

Temporal structure preservation

Because the largest dataset we use is also timeseries data, we assess temporal structure preservation for the test set by calculating Euclidean distances between sequential time points in high-dimensions and low-dimensions for each method. We then calculate the Pearson correlation coefficient of the log transformed distances (same as for assessing global structure preservation) for 50 randomly selected 10 minute subsets (60,000 observations) within the full timeseries. This then provides a measure of how well temporal derivatives are preserved in the low-dimensional embedding ranging from -1 (anti-correlated) to 1 (correlated).

Hierarchical bootstrap for statistical comparisons

To compare each information preservation metric statistically we performed hierarchical bootstrapping (see Saravanan et al. 2019 for a recent review). Every trial for each dimension reduction method has multiple observations per metric, which creates hierarchical dependencies in the data. To account for this, we use seaborn v0.10.1 (Waskom et al., 2020) to calculate and plot hierarchical bootstrap estimates of the mean for each information preservation metric — resampling (with replacement) both within trials and across trials (n=1000 bootstrap samples). We then plot the 95% intervals of the bootstrap distribution to compare the performance of each dimension reduction method statistically. Rather than attempting to make decisions regarding the statistical “significance" of these bootstrap distributions based on an arbitrary threshold, we instead simply treat them as a measure of the uncertainty (variance) in effect size for each information preservation metric. The computational experiments from which the information preservation metrics are derived could be run ad infinitum to achieve statistical significance, which is effectively a measure of statistical resolution based on the number of observations, but this is not necessarily informative in practice.

3.5.3 Datasets

Animal body posture dynamics

The largest dataset we used for comparisons is a behavioral dataset from Berman et al. (2016, 2014a); Pereira et al. (2019) consisting of $\sim$1-h video recordings (at 100Hz) for 59 freely-behaving individual fruit flies (*Drosophila melanogaster*) for a total of $\sim$21.1 million observations (downloaded from: http://arks.princeton.edu/ark:/88435/dsp01pz50gz79z). We tracked the full

body posture of each individual with DeepPoseKit v0.3.6 (Graving et al., 2019a) using the procedures described by Graving et al. (2019a) to train a deep convolutional pose estimation model using the keypoint annotations from Pereira et al. (2019) as training data. For each video this produced a multivariate timeseries of the Euclidean coordinates describing 32 body part positions in the video — including the head, neck, eyes, thorax, abdomen, wings, and 24 leg joints. We then rotationally and translationally aligned the posture data at each timepoint to the major body axis (neck-thorax vector) and calculated the sine and cosine of the keypoint angles for the 30 body parts not used for alignment. This resulted in a $30 \times 2 = 60$ dimensional posture timeseries. To transform the spatial posture data into a dynamical spatio-temporal representation, we then applied a normalized Morlet wavelet transform from Berman et al. (2014a) using the behavelet Python package v0.0.1 (Graving, 2019) to generate a multi-scale time-frequency spectrogram of the body posture dynamics for each time point. Following Berman et al. (2014a); Pereira et al. (2019), we used 25 dyadically ($\log_2$) spaced frequencies ranging from 1Hz to 50Hz (the Nyquist frequency of the signal), which expanded the dimensionality of the timeseries from $30 \times 2 = 60$ to $30 \times 2 \times 25 = 1500$.

Dimension reduction comparisons To generate a training set for benchmarking the different algorithms, we uniformly randomly sampled a subset of data from the body posture dynamics timeseries for 58 of 59 individuals while excluding one randomly selected individual to use as a test set. We tested 4 training set sizes: $58 \times 500 = 29,000$; $58 \times 1000 = 58,000$; $58 \times 2000 = 116,000$; $58 \times 4000 = 232,000$, above which we encountered out-of-memory errors when running UMAP (McInnes et al., 2018) on larger subsets of data. Each test set contains $\sim 360,000$ sequential observations. We then applied each dimension reduction method to the training set and subsequently embedded the test set. For training VAE-SNE we used the cross-entropy loss as a log likelihood function, as it matches well with the normalized time-frequency data, but we also found that other likelihood functions work similarly well.

Behavioral clustering To simplify the dataset for performing our clustering analysis, we used the sine and cosine of the keypoint angles for the 6 legs (the distal tips of each leg), 2 wings, head, and abdomen for a total of 10 body parts and a $10 \times 2 = 20$ dimensional posture timeseries. As before we applied the time-frequency transform which expands the dimensionality of the timeseries from $10 \times 2 = 20$ to $10 \times 2 \times 25 = 500$. We then applied VAE-SNE with a t-SNE

kernel (Appendix B.2; $\nu = \tau = 1$) to compress the spectrogram data to 30 dimensions. We used the cross-entropy between normalized time-frequency vectors, or $\mathbb{H}[\mathbf{x}_i, \mathbf{x}_j] = -\sum \mathbf{x}_i \log \mathbf{x}_j$, as our metric for calculating high-dimensional similarities (Appendix B.2), as this provides a more natural measure of divergence between the normalized spectrograms than Euclidean distance. The cross-entropy is closely related (up to a constant) to the Kullback-Leibler divergence — the metric originally used by Berman et al. (2014a) — but is slightly faster to calculate, which reduces training time. When visualizing the spectrograms we integrate (sum) across the wavelet coefficients for the sine and cosine for each body part in the spectrogram.

Single-cell RNA-seq

To test the application of VAE-SNE to single-cell RNA-seq data, we used data from La Manno et al. (2018) which consists of 18,213 observations describing the development and cell fate of hippocampal neurons. We preprocessed these data using the velocyto.py (v0.17.17) package from La Manno et al. (2018). We compressed the raw expression values to 500 dimensions using PCA before applying subsequent dimension reduction algorithms. We applied each dimension reduction algorithm to the full dataset and then re-embedded the training set in place of a test set in order to evaluate the speed for embedding new data. We report information preservation metrics only for the training set, as no test set was used due to the relatively small size of the dataset. For training VAE-SNE on this dataset we use a Student-t likelihood function, but found other likelihood functions work similarly well.

Natural history images

We also applied a convolutional variant of VAE-SNE to natural history images, and to test this we used two datasets: a set of 59,244 shell images from Q. Zhang et al. (2019) and a set of 2,468 butterfly images from J. F. H. Cuthill et al. (2019). All images were preprocessed by applying local adaptive thresholding to detect and remove the background. Images were then zero-padded to create a 1:1 aspect ratio and resized to a resolution of 192×192. We trained convolutional VAE-SNE using the same hyperparameters as the dimension reduction experiments, but using batches of only 256 images.

3.5.4 Computing hardware

All performance comparisons were conducted on a high-end consumer-grade workstation equipped with an Intel Core-i9-7900X CPU (10 cores, 20 threads @ 3.30GHz), 32GB of DDR4 RAM, a 4TB NVMe solid state drive, and a NVIDIA GeForce GTX 1080 Ti GPU (11 GB GDDR5X VRAM).

3.5.5 Parallelizing pairwise computations to improve performance

To improve performance of pairwise computations over Ding et al. (2018), we reimplemented the underlying algorithms for training VAE-SNE. The largest performance bottleneck for VAE-SNE is the recursive binary search algorithm for computing high-dimensional pairwise similarities (Appendix B.2). However, the computations for this algorithm are embarrassingly parallel, so we reimplemented it to run recursion loops in parallel across multiple CPU threads. This was accomplished by JIT-compiling the code using the numba library (Lam et al., 2015), which resulted in massive speed improvements. We also reimplemented all pairwise distance calculations on the GPU using PyTorch (Paszke et al., 2019), which further improved performance.

3.5.6 Code availability

The code for VAE-SNE is freely available at https://github.com/jgraving/vaesne under a permissive open-source license. The library is written primarily using PyTorch v1.5.0 (Paszke et al., 2019) and includes a scikit-learn-style API (Buitinck et al., 2013) for fitting the model (`model.fit()`) and predicting on new data (`model.predict()`).

Discussion

Here I have discussed computer vision and deep learning-based methods for the study of animal behavior. I highlighted existing methods and introduced new advances in these methodological areas that helped to solve limitations of existing approaches. Some of the work presented here has already been widely adopted by the animal behavior community and has aided other researchers to make further advances in the field of behavioral quantification. However, this field of quantitative behavioral research is still in its infancy and the techniques presented in this thesis are only scratching the surface of the full set of approaches that are available for studying animal behavior. In the future, it is likely that even more general-purpose methods will be developed and applied to a range of experimental tasks. Considering the different use cases and applications that are highlighted in this thesis, deep learning and computational modeling will almost certainly become (and arguably already is) an important and powerful general-purpose tool for understanding animal behavior.

The techniques presented in this thesis for measuring behavior have already evolved since their publication. For example, some of the methods have been made obsolete, or the general ideas have been built on and improved in subsequent work by other researchers, which only highlights the fast-moving nature of the field. The conventional 2D barcode tracking algorithms used in Chapter 1 are now out-of-date, and newer, more robust deep learning methods have been shown to be much more reliable for this task (Boenisch et al., 2018; Hu et al., 2019; Sixt et al., 2018; Wild et al., 2018). Instead of using conventional computer vision to localize and decode barcodes, deep learning algorithms can be trained with artificially augmented data that better match real-world scenarios where conventional methods often fail (Sixt et al., 2018). This approach includes simple techniques like using conventional methods to generate training data for deep learning algorithms, which can then be used to detect, localize, and decode barcodes (Hu et al., 2019), as well as algorithms that generate completely artificial data in a way that matches the real-world image statistics of the task (Sixt et al., 2018). There have also been

recent advances in using deep learning for markerless recognition of individual birds (Ferreira et al., 2019), where RFID tags are used in combination with inexpensive camera hardware to automatically generate training data for individual identification.

Other researchers working to advance individual tracking and pose estimation methods have begun to iterate on and apply the ideas proposed in Chapter 2. For example, newer methods have begun to integrate temporal information when training pose estimation models (Liu et al., 2020; Wu et al., 2020) and have extended pose estimation methods to directly estimate posture and track pairs and groups of individuals directly from experimental videos without any of the preprocessing described in Chapter 2 (Pereira et al., 2020). However, many limitations still remain for these methods and there is a great deal of work left to be done. For example, generalizing to new experimental scenarios with limited training data (A. Mathis, Biasi, et al., 2020) and re-identifying individuals after visual occlusions both still remain exceptionally complex problems, especially in cases where the lighting or other environmental factors have changed dramatically (A. Mathis, Biasi, et al., 2020; Romero-Ferrero et al., 2019). The task of multiple pose estimation and pose tracking in 3D, especially in difficult field scenarios, still remains a challenging open problem as well. The extension of these algorithms to real-time and field-based experiments also poses issues; however, there has been recent progress on both of these fronts (Kane et al., 2020; Zuffi et al., 2019).

In general, the tools and methods presented in this thesis already provide ample opportunities to answer scientific questions, but the complexity of these data necessitates new tools to extract meaningful and interpretable information. In Chapter 3, I introduced general-purpose methods for interpreting and analyzing these behavioral data, however, this is only a first step and one of many possible ways to apply data-driven modeling for addressing important research topics. I discussed some of these challenges in Chapters 2 and 3, and this topic has also been discussed extensively elsewhere (Berman, 2018; Brown & De Bivort, 2018; Datta et al., 2019). In particular, the methods I present in Chapter 3 can only model relatively short timescales and do not take into account longer timescales or individual differences in behavior. Future work will have to reckon with the conceptual challenges of modeling detailed behavioral data in a way that allows researchers to answer important scientific questions while also balancing model complexity with interpretability. Several approaches have been proposed for modeling behavior in recent years (Berman et al., 2016, 2014a; Costa et al., 2019; M. Johnson et al., 2016; Markowitz et al., 2018; Wiltschko et al., 2015), but all have their limitations. In the end, the complexities of modeling

behavioral data enters the realm of the philosophical. However the central role of the scientist is to undertake these decisions for how best to model and interpret their data when making inferences about the natural world.

Appendices

Appendix A

Appendix to “DeepPoseKit: a software toolkit for fast and robust animal pose estimation using deep learning"

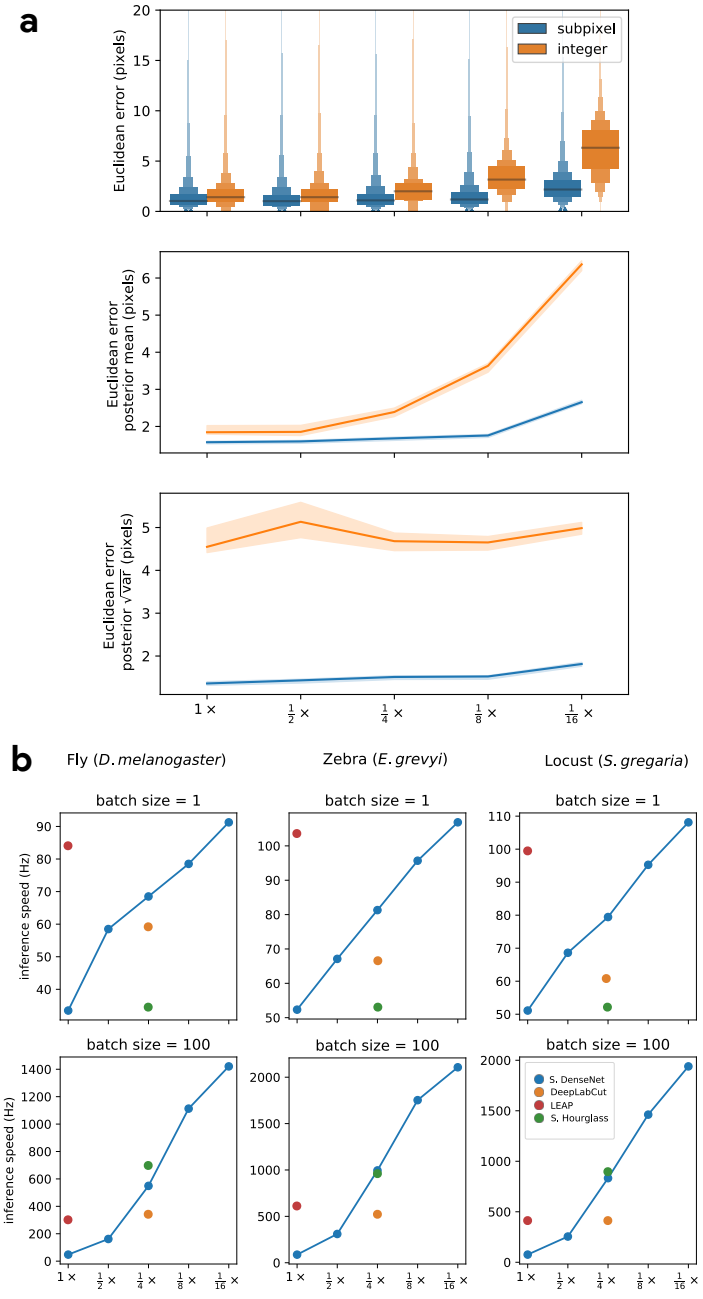

Figure A.1. Our subpixel maxima algorithm increases speed without decreasing accuracy. Prediction accuracy on the fly dataset is maintained across downsampling configurations (**a**). Letter-value plots (**a-top**) show the raw error distributions for each configuration. Visualizations of the credible intervals (99% highest-density region) of the posterior distributions for the mean and variance (**a-bottom**) illustrate statistical differences between the error distributions, where using subpixel maxima decreases both the mean and variance of the error distribution. Inference speed is fast and can be run in real-time on single images (batch size = 1) at ∼30-110Hz or offline (batch size = 100) upwards of 1000Hz (**b**). Plots show the inference speeds for our Stacked DenseNet model across downsampling configurations as well as the other models we tested for each of our datasets.

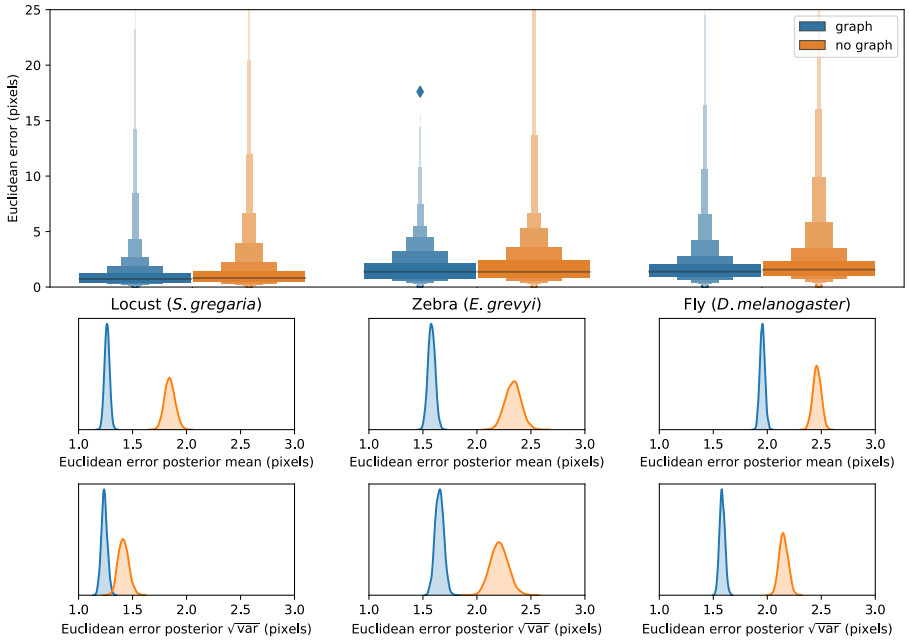

Figure A.2. Predicting the multi-scale geometry of the posture graph reduces error. Letter-value plots (**top**) show the raw error distributions for each experiment. Visualizations of the posterior distributions for the mean and variance (**bottom**) show statistical differences between the error distributions. Predicting the posture graph decreases both the mean and variance of the error distribution.

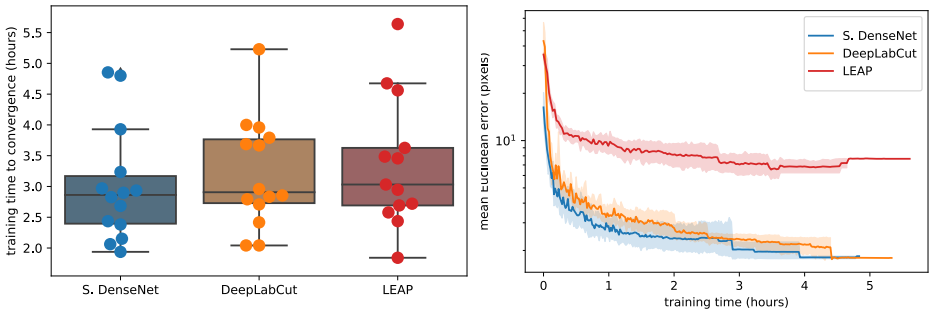

Figure A.3. Training time required for our Stacked DenseNet model, the DeepLabCut model (A. Mathis et al., 2018), and the LEAP model (Pereira et al., 2019) (n=15 per model) using our zebra dataset. Boxplots and swarm plots (**left**) show the total training time to convergence (<0.001 improvement in validation loss for 50 epochs). Line plots (**right**) illustrate the Euclidean error of the validation set during training, where error bars show bootstrapped (n=1000) 99% confidence intervals of the mean. Fully training models to convergence requires only a few hours of optimization (**left**) with reasonable accuracy reached after only 1 hour (**right**) for our Stacked DenseNet model.

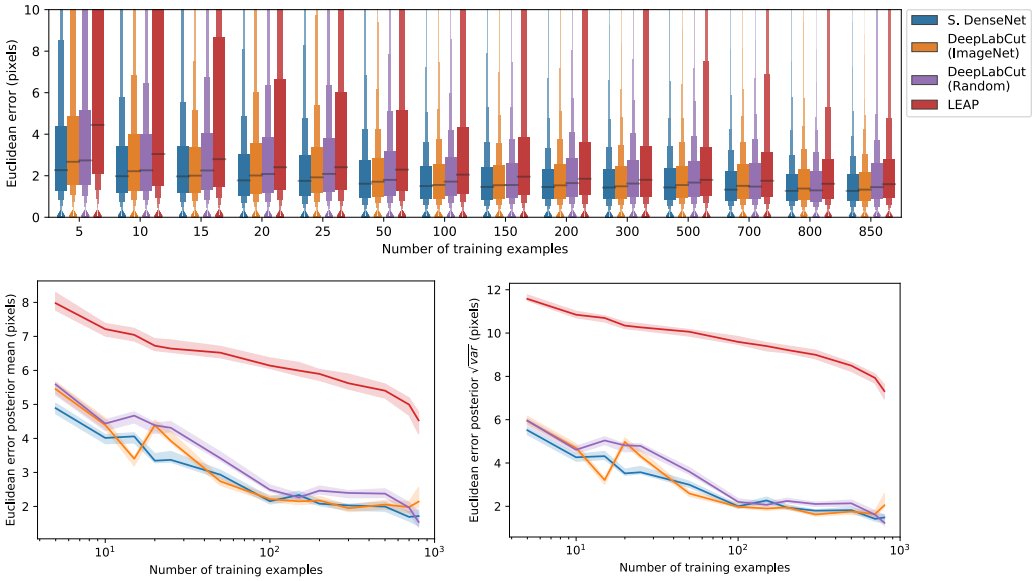

Figure A.4. A comparison of prediction accuracy with different numbers of training examples from our zebra dataset. The error distributions shown as letter-value plots (**top**) illustrate the Euclidean error for the remainder of the dataset not used for training—with a total of 900 labeled examples in the dataset. Line plots (**bottom**) show posterior credible intervals (99% highest-density region) for the mean and variance of the error distributions. We tested our Stacked DenseNet model; the DeepLabCut model (A. Mathis et al., 2018) with transfer learning—i.e., with weights pretrained on ImageNet (Deng et al., 2009); the same model without transfer learning—i.e., with randomly-initialized weights; and the LEAP model (Pereira et al., 2019). Our Stacked DenseNet model achieves high accuracy using few training examples without the use the transfer learning. Using pretrained weights does slightly decrease overall prediction error for the DeepLabCut model (A. Mathis et al., 2018), but the effect size is relatively small.

A.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 (Badrinarayanan et al., 2015; Long et al., 2015).

Once a training set is generated (Appendix A.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 A.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* (Kiefer et al., 1952; Robbins & Monro, 1951). 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 & 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).

A.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 (Insafutdinov et al., 2016; A. Mathis et al., 2018; Pratt, 1993)— 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 2.1).

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. A. Mathis et al. (2018) recently added a Python-based GUI in an updated version of their software—including active learning and image sampling methods (see Nath et al. 2019a). Our framework also includes a Python-based GUI for annotating data with similar features to A. Mathis et al. (2018) and Pereira et al. (2019).

A.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 (Insafutdinov et al., 2016; A. Mathis et al., 2018; Newell et al., 2016; 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)— directional 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., $p(\mathrm{foot}|\mathrm{head})$). 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).

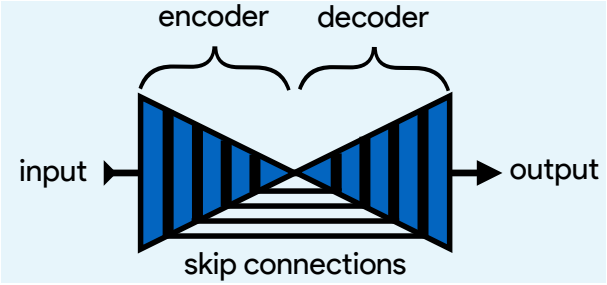

Figure A.5. An illustration of the basic encoder-decoder design. The encoder converts the input images into spatial features, and the decoder transforms spatial features to the desired output.

A.4 Encoder-decoder models

A popular type of F-CNN (Appendix A.3) for solving posture regression problems is known as an *encoder-decoder* model (Figure A.5), 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 G. Huang et al.'s (2017) *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 & 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 (G. Huang et al., 2017). Recent work has also shown that DenseNet's elaborate residual connections 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.

A.5 The state of the art for individual pose estimation

Many of the current state-of-the-art models for individual posture estimation are based on the design from Newell et al. (2016) (e.g., Ke et al. 2018, Y. Chen et al. 2017; also see benchmark results from Andriluka et al. 2014), but employ various modifications that increase complexity to improve performance. Newell et al. (2016) employ what they call a *Stacked Hourglass* network (Appendix A.3 Figure A.6), which consists of a series of multi-scale encoder-decoder *hourglass* modules connected together in a feed-forward configuration (Figure 2.3). 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

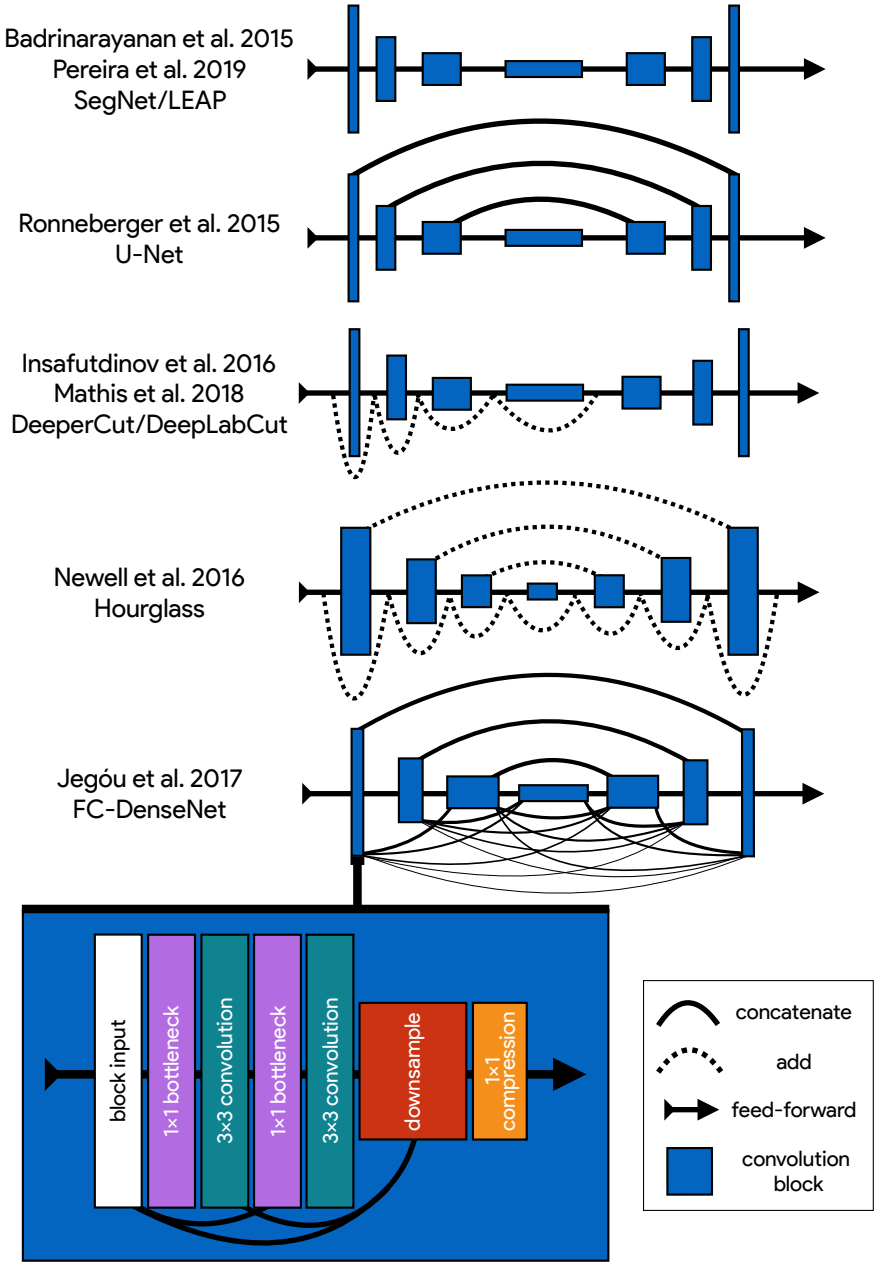

Figure A.6. An illustration showing the progression of encoder-decoder architectures from the literature—ordered by performance from top to bottom (see Appendix A.3 Box A.4 for further details). Most advances in performance have come from adding connections between layers in the network, culminating 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 G. Huang et al. 2017) to increase efficiency (Appendix A.7)

intermediate supervision (Figure 2.3) 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 A.3 Box A.4 and Appendix A.3 Figure A.6), but to the best of our knowledge, the model introduced by Jégou et al. (2017) has not been previously applied to pose estimation.

A.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 A.3 Figure A.6; Appendix A.3 Box A.4). 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 the LEAP model (Pereira et al., 2019) unsuitable in many cases. While their approach is simple and effective for a multitude of experimental setups, the LEAP model (Pereira et al., 2019) is also implicitly limited in the same ways as Badrinarayanan et al.'s SegNet model (see Appendix A.3 Box A.4 for details). The LEAP model cannot make predictions using multiple spatial scales and is not robust to data variance such as rotations (Pereira et al., 2019).

A.7 Linear model fitting 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 & 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.

A.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 hourglass networks with intermediate supervision (Appendix A.4) applied at the output of each network. We also include hyperparameters for the bottleneck and compression layers described by G. Huang et al. (2017) 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 G. Huang et al. 2017 and Appendix A.3 Figure A.6 for details).

A.9 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 the LEAP model (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 this model in the original configuration described by Pereira et al. (2019). The LEAP model could be modified to output smaller confidence maps

and increase inference speed, but because there is no obvious "best" way to alter the model to achieve this, we forgo any modification. Additionally, applying our subpixel maxima algorithm at high resolution reduces inference speed compared to integer-based maxima, so this would bias our speed comparisons.

A.10 Our implementation of the DeepLabCut model

Because the DeepLabCut model from A. Mathis et al. (2018) was not implemented in Keras (a requirement for our pose estimation framework), we re-implemented it. Implementing this model directly in our framework is important to ensure model training and data augmentation are identical when making comparisons between models. As a consequence, our version of this model does not exactly match the description in the paper but is identical except for the output. Rather than using the location refinement maps described by Insafutdinov et al. (2016) and post-processing confidence maps on the CPU, our version of the DeepLabCut model (A. Mathis et al., 2018) has an additional transposed convolutional layer to upsample the output to $\frac{1}{4}\times$ resolution and uses our subpixel maxima algorithm.

To demonstrate that our implementation of the DeepLabCut model matches the performance described by A. Mathis et al. (2018), we compared prediction accuracy between the two frameworks using the odor-trail mouse dataset provided by A. Mathis et al. (2018) (downloaded from `https://github.com/AlexEMG/DeepLabCut/`). This dataset consists of 116 images of a freely-moving individual mouse labeled with four keypoints describing the location of the snout, ears, and the base of the tail. See A. Mathis et al. (2018) for further details on this dataset. We trained both models using 95% training and 5% validation data and applied data augmentations for both frameworks using the data augmentation procedure described by Nath et al. (2019a). We tried to match these data augmentations as best as possible in DeepPoseKit; however, rather than cropping images as described by Nath et al. (2019a), we randomly translated the images independently along the horizontal and vertical axis by drawing from a uniform distribution in the range [-100%, +100%]—where percentages are relative to the size of each axis. Translating the images in this way should serve the same purpose as cropping them.

We trained the original DeepLabCut model (A. Mathis et al., 2018) using the default settings and recommendations from Nath et al. (2019a) for 1 million training iterations. See A. Mathis et al. (2018); Nath et al. (2019a) for further details on the data augmentation and training routine

for the original implementation of the DeepLabCut model (A. Mathis et al., 2018). For our re-implementation of the DeepLabCut model (A. Mathis et al., 2018) we trained the model with the same batch size and optimization scheme described in the "Model training" section. We then calculated the the prediction accuracy on the full data set. We repeated this procedure five times for each model and fit a Bayesian linear model to a randomly selected subset of the evaluation data to compare the results statistically (see Appendix A.6).

These results demonstrate that our re-implementation of and modification to the DeepLabCut model (A. Mathis et al., 2018) have little effect on prediction accuracy (Appendix A.7 Figure A.7). We also provide qualitative comparisons of these results in Appendix A.7 Figure A.7-Figure supplement A.8 and Appendix A.7 Figure A.7-video A.10. For these qualitative comparisons, we also added an additional rotational augmentation (drawing from a uniform distribution in the range [-180°, +180°)) when training our implementation of the DeepLabCut model (A. Mathis et al., 2018) as we noticed this improved generalization to the video for situations where the mouse rotated its body axis. To the best of our knowledge, rotational augmentations are not currently available when using the software from A. Mathis et al. (2018); Nath et al. (2019a), which demonstrates the flexibility of the data augmentation pipeline (Jung, 2018) for DeepPoseKit. The inference speed for the odor-trail mouse dataset using our implementation of the DeepLabCut model (A. Mathis et al., 2018) is $\sim$49Hz with a batch size of 64 (offline speeds) and $\sim$35Hz with a batch size of 1 (real-time speeds) at full resolution 640$\times$480, which matches well with results from A. Mathis & Warren (2018) of $\sim$47Hz and $\sim$32Hz respectively. This suggests our modifications did not affect the speed of the model and that our speed comparisons are also reasonable. Because the training routine could be changed for any underlying model—including the new models we present in this paper—this factor is not relevant when making comparisons as long as training is identical for all models being compared, which we ensure when performing our comparisons.

A.11 Depthwise-separable convolutions for memory-limited applications

In an effort to maximize model efficiency, we also experimented with replacing 3$\times$3 convolutions in our model implementations with 3$\times$3 depthwise-separable convolutions —first introduced by Chollet (2017) and now commonly used in fast, efficient “mobile” CNNs (e.g., Sandler et

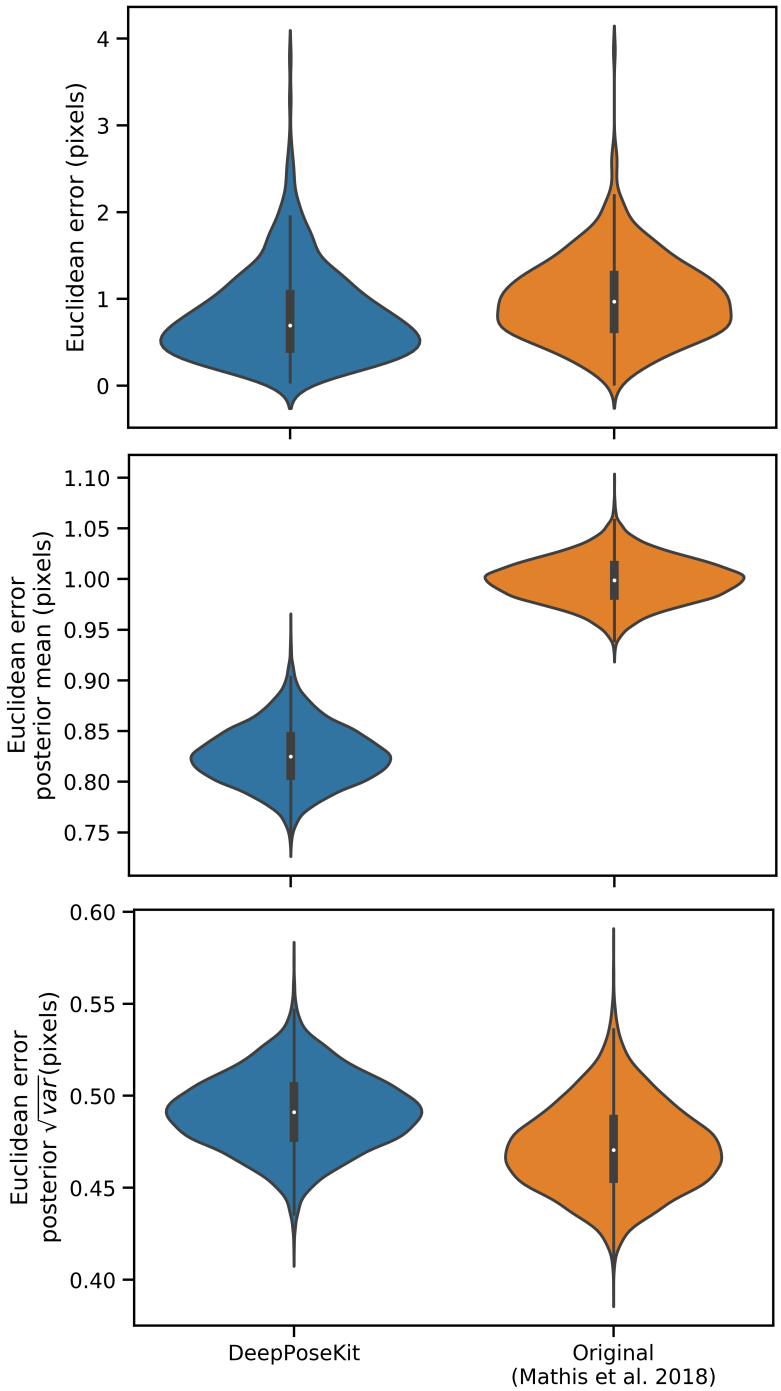

Figure A.7. Prediction errors for the odor-trail mouse dataset from A. Mathis et al. (2018) using the original implementation of the DeepLabCut model (A. Mathis et al., 2018; Nath et al., 2019a) and our modified version of this model implemented in DeepPoseKit. Mean prediction error is slightly lower for the DeepPoseKit implementation, but there is no discernible difference in variance. These results indicate that the models achieve nearly identical prediction accuracy despite modification. We also provide qualitative comparisons of these results in Appendix A.7 Figure A.7-Figure supplements A.8 and A.9, and Appendix A.7 Figure A.7-video A.10.

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

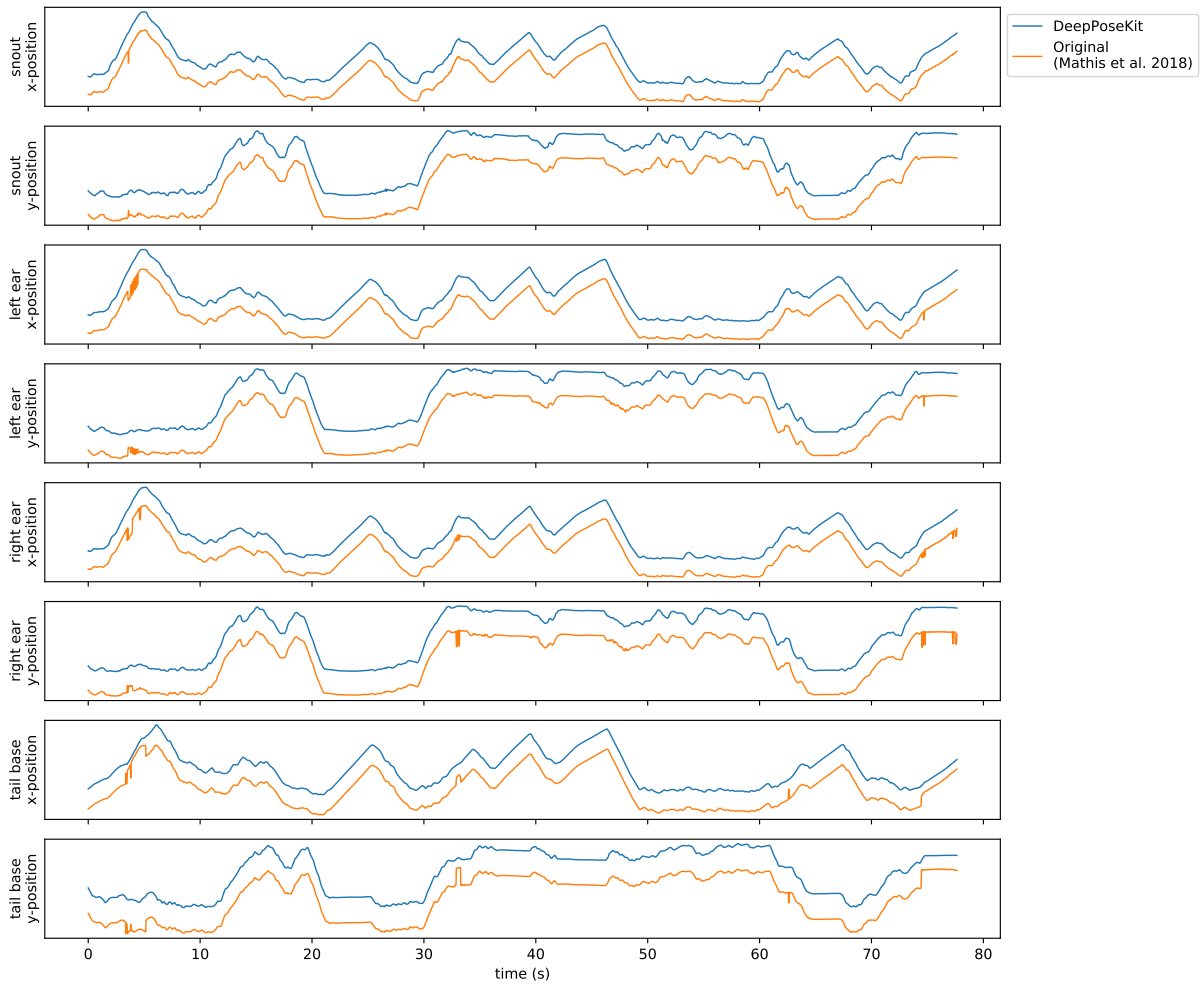

Figure A.8. Plots of the predicted output for Appendix A.7 Figure A.7-video A.10 comparing our implementation of the DeepLabCut model (A. Mathis et al., 2018) in DeepPoseKit vs. the original implementation from A. Mathis et al. (2018); Nath et al. (2019a). Note the many fast jumps in position for the original verison from A. Mathis et al. (2018), which indicates prediction errors.

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.

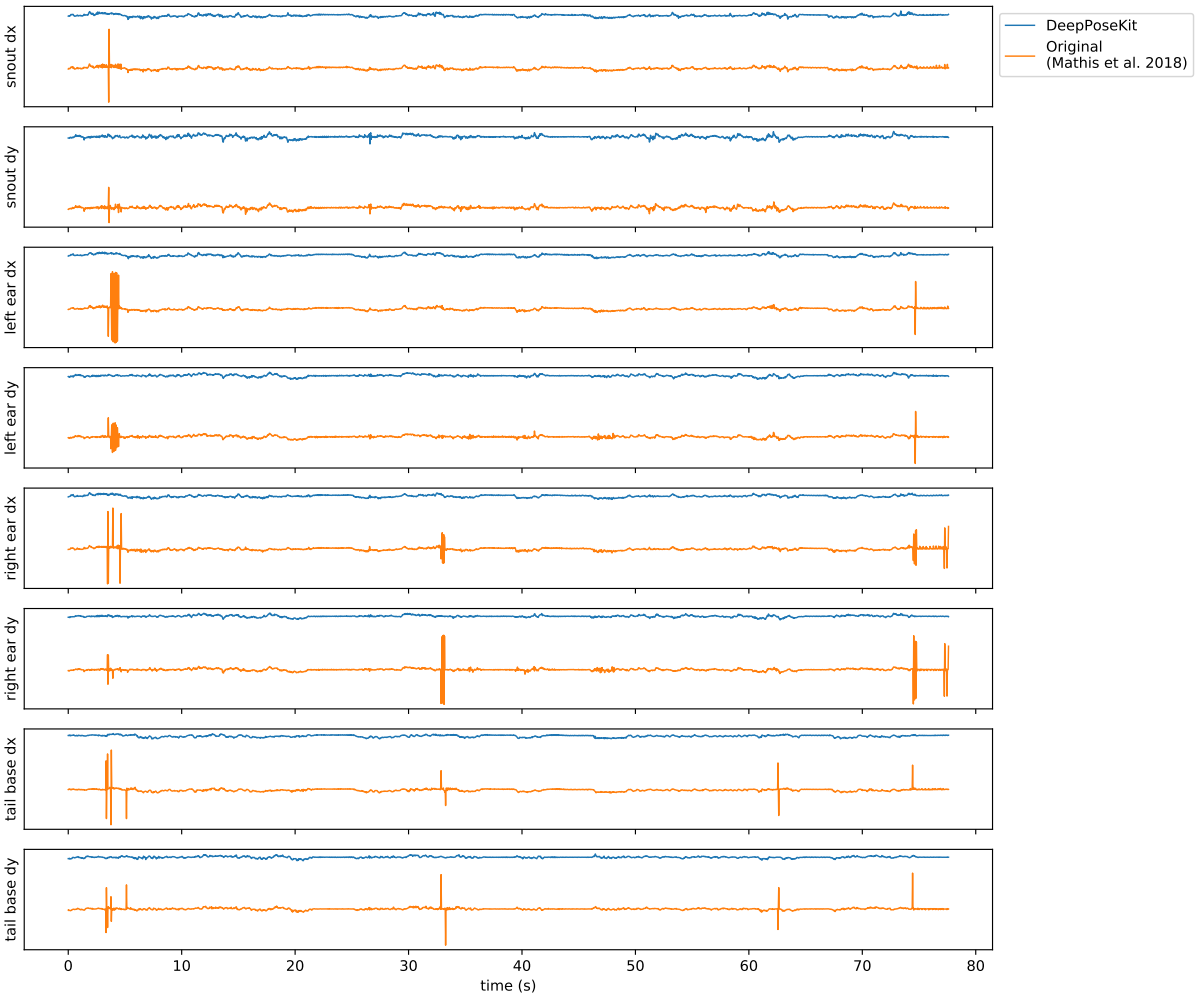

Figure A.9. Plots of the temporal derivatives of the predicted output for Appendix A.7 Figure A.7-video A.10 comparing our implementation of the DeepLabCut model (A. Mathis et al., 2018) in DeepPoseKit vs. the original implementation from A. Mathis et al. (2018); Nath et al. (2019a). Note the many fast jumps in position for the original verison from A. Mathis et al. (2018), which indicates prediction errors

Figure A.10. A video comparison of the tracking output of our implementation of the DeepLabCut model (A. Mathis et al., 2018) in DeepPoseKit vs. the original implementation from A. Mathis et al. (2018); Nath et al. (2019a). `https://youtu.be/YFmO5C0hUw4`

Appendix B

Appendix to “VAE-SNE: a deep generative model for simultaneous dimensionality reduction and clustering"

Table B.1. Ranked information preservation metric performance for nonlinear dimension reduction algorithms. Rankings for each nonlinear dimension reduction algorithm in terms of general performance for local, global, fine-scale, and temporal structure preservation (lower is better).

name	citation	local	global	fine-scale	temporal
VAE-SNE (t-SNE)	this paper	1	1	2	2
VAE-SNE (SNE)	this paper	2	1	2	1
FIt-SNE	Linderman et al. (2017)	1	2	1	2
Barnes-Hut-SNE	van der Maaten (2014)	1	2	1	2
UMAP (LE init)	McInnes et al. (2018)	1	3	2	3
UMAP (PCA init)	McInnes et al. (2018)	1	2	2	3
scvis	Ding et al. (2018)	2	1	2	2
ivis	Szubert et al. (2019)	2	1	3	1

Table B.2. Ranked processing speed performance for nonlinear dimension reduction algorithms. Rankings for each nonlinear dimension reduction algorithm in terms of general performance for training time and test time (lower is better), as well as whether or not test time increases as a function of training set size.

name	citation	train time	test time	test time $\propto$ train size
VAE-SNE	this paper	4	1	no
FIt-SNE	Linderman et al. (2017)	2	4	no
Barnes-Hut-SNE	van der Maaten (2014)	3	5	yes
UMAP	McInnes et al. (2018)	1	3	yes
scvis	Ding et al. (2018)	6	1	no
ivis	Szubert et al. (2019)	5	2	no

Table B.3. Additional features for nonlinear dimension reduction algorithms. A summary of potentially useful additional features for each nonlinear dimension reduction algorithm including batch training for applying dimension reduction to large out-of-core datasets, non-Euclidean embeddings for different types of compressed representations, whether the algorithm is tractable in higher dimensions ($>$2), and whether the algorithm learns a distribution of clusters within the data.

name	citation	batch training	non-Euclidean	$>$2 dims.	clustering
VAE-SNE	this paper	yes	yes	yes	yes
FIt-SNE	Linderman et al. (2017)	no	no	no	no
Barnes-Hut-SNE	van der Maaten (2014)	no	no	yes	no
UMAP	McInnes et al. (2018)	no	yes	yes	no
scvis	Ding et al. (2018)	yes	no	yes	no
ivis	Szubert et al. (2019)	yes	no	yes	no

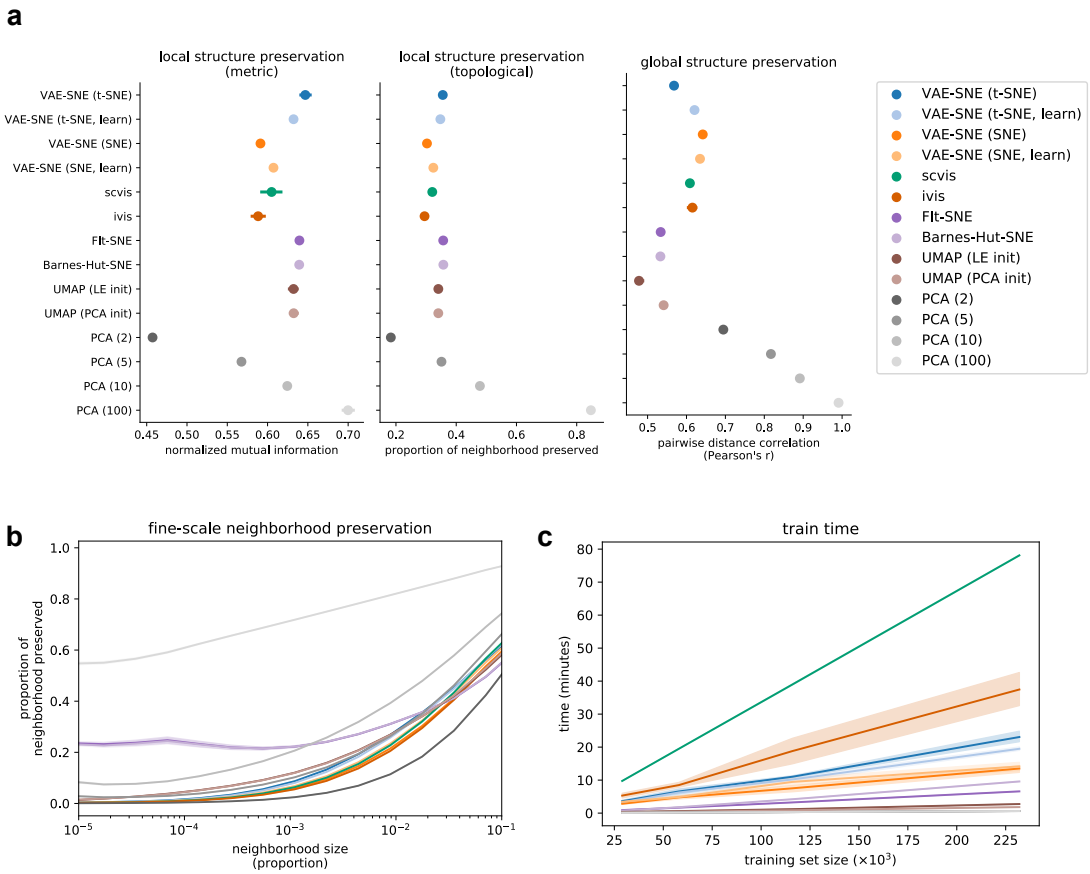

Figure B.1. Dimension reduction performance for the posture dynamics training set. Plots show performance comparisons for the posture dynamics dataset (Berman et al., 2016, 2014a; Pereira et al., 2019) using the training set. **a**, Mean and 95% interval of the bootstrap distribution for local and global structure preservation. Results are pooled across all training set sizes (for each metric n = 4 training set sizes $\times$ 5 trials $\times$ 5 replicates = 100 per algorithm). **b**, Mean and 95% interval of the bootstrap distribution for fine-scale structure preservation across multiple neighbor sizes (as a proportion of the total embedding size). Results are from the largest training set size only (n = 14 neighborhood sizes $\times$ 5 trials $\times$ 5 replicates = 350 per algorithm). **c**, Training time for fitting each algorithm across different training set sizes (n = 4 training set sizes $\times$ 5 trials = 20 per algorithm).

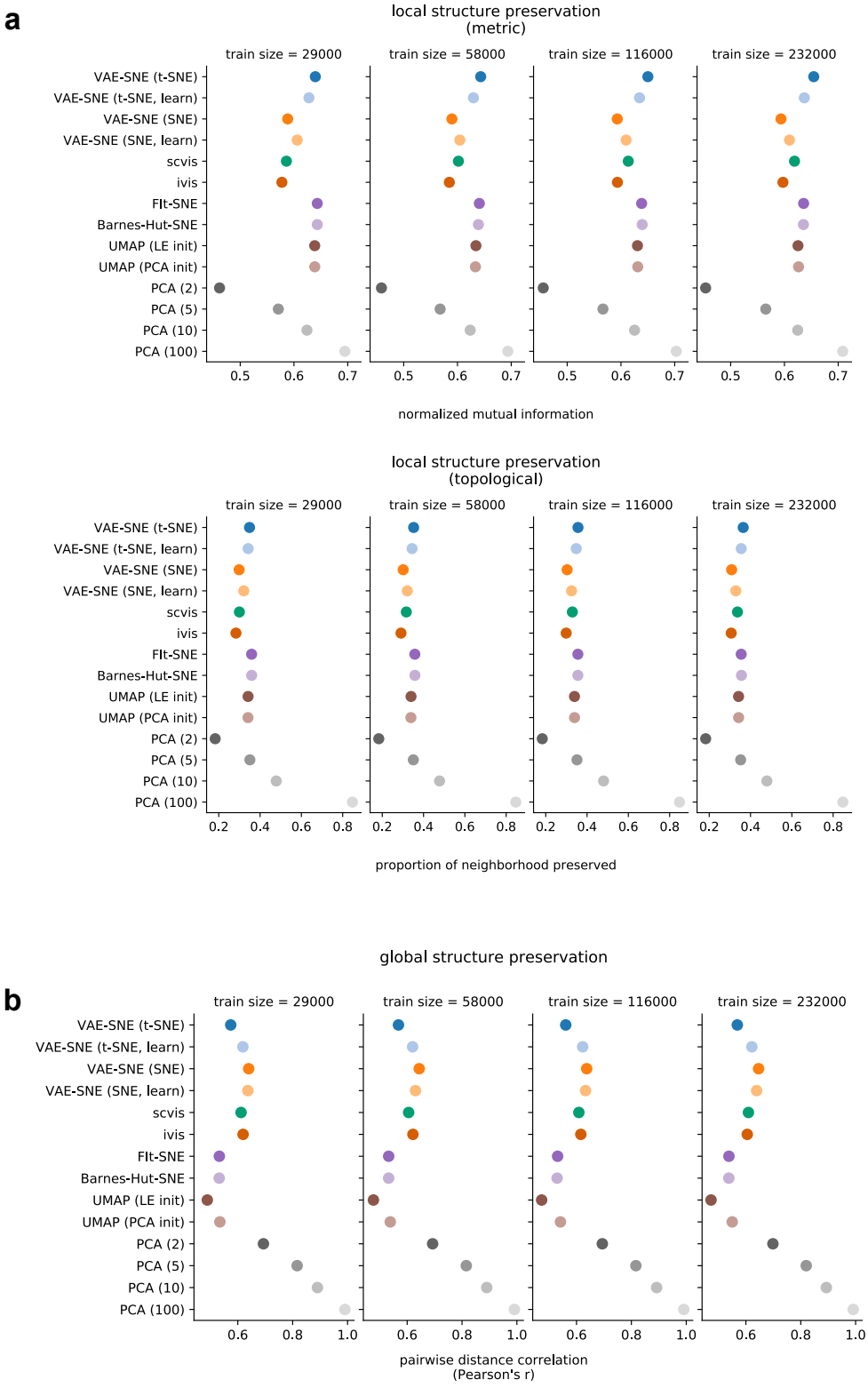

Figure B.2. Dimension reduction performance for the posture dynamics training set across training set sizes. Plots show performance comparisons for the posture dynamics dataset (Berman et al., 2016, 2014a; Pereira et al., 2019) using training sets of different sizes. **a-b**, Mean and 95% interval of the bootstrap distribution for local (**a**) and global (**b**) structure preservation. (for each metric n = 5 trials $\times$ 5 replicates = 25 per training set size per algorithm)

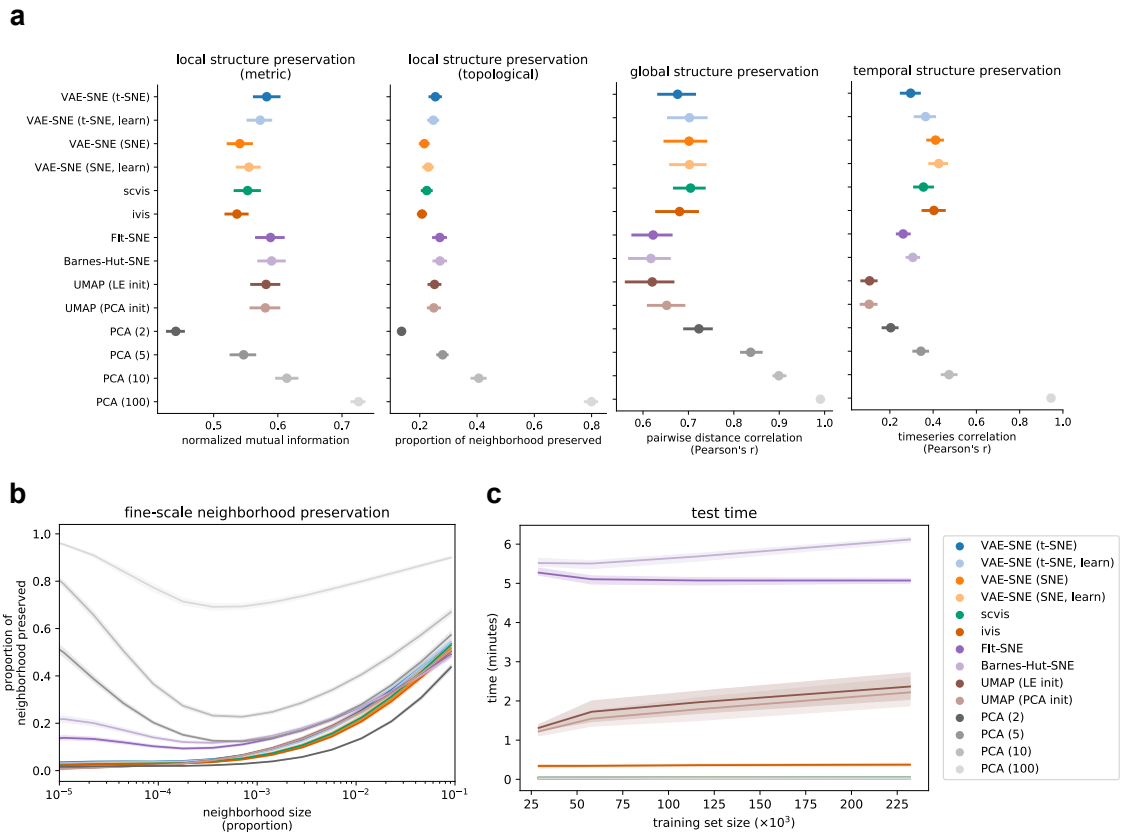

Figure B.3. Dimension reduction performance for the posture dynamics test set. Plots show performance comparisons for the posture dynamics dataset (Berman et al., 2016, 2014a; Pereira et al., 2019) using the test set. **a**, Mean and 95% interval of the bootstrap distribution for local, global, and temporal structure preservation. Results are pooled across all training set sizes (for local and global structure n = 4 training set sizes $\times$ 5 trials $\times$ 5 replicates = 100 per algorithm; for temporal structure n = 4 training set sizes $\times$ 5 trials $\times$ 50 subsamples = 1000 per algorithm). **b**, Mean and 95% interval of the bootstrap distribution for fine-scale structure preservation across multiple neighbor sizes (as a proportion of the total embedding size). Results are from the largest training set size only (n = 14 neighborhood sizes $\times$ 5 trials $\times$ 5 replicates = 350 per algorithm). **c**, Elapsed time for embedding the test set with each algorithm across different training set sizes (n = 4 training set sizes $\times$ 5 trials = 20 per algorithm).

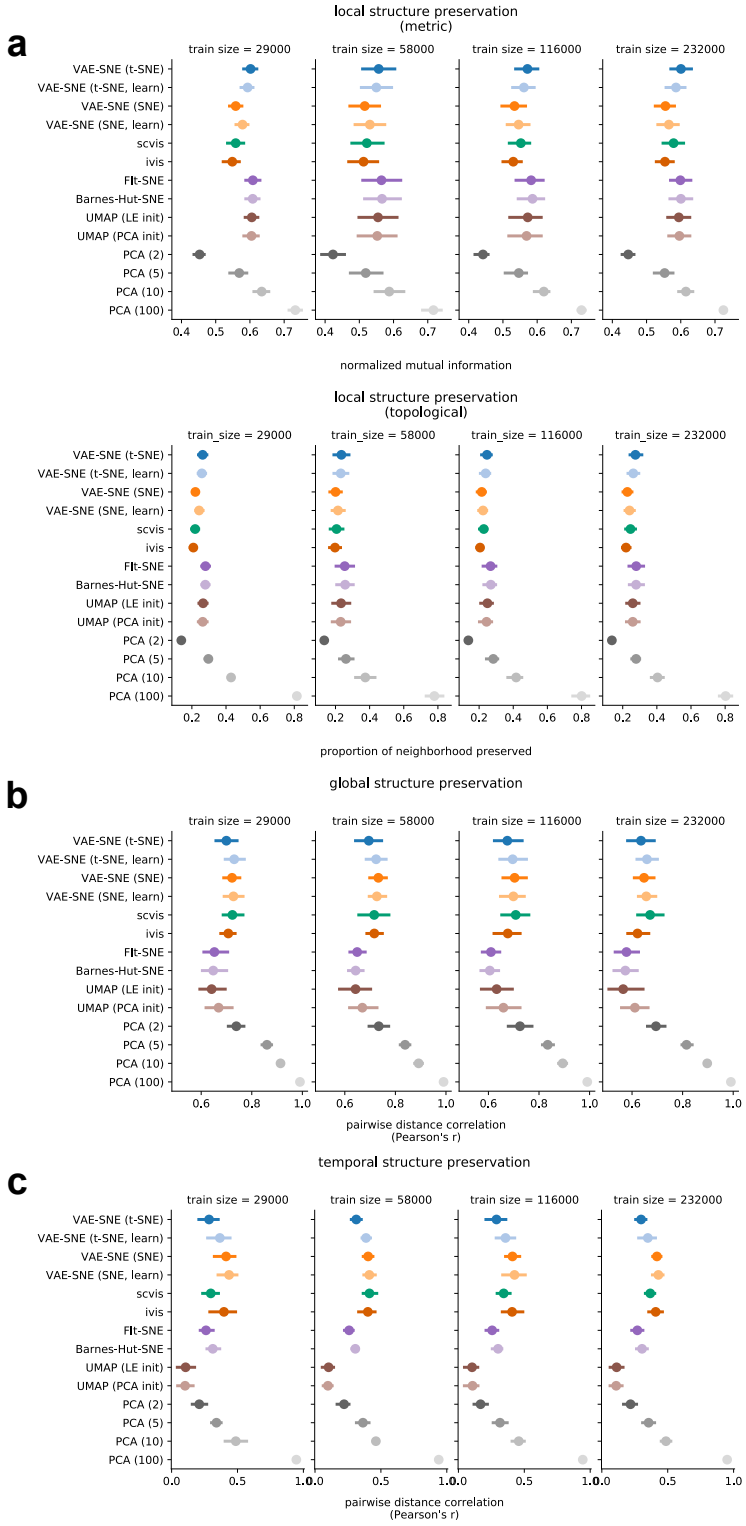

Figure B.4. Dimension reduction performance for the posture dynamics test set across training set sizes. Plots show performance comparisons for the posture dynamics dataset (Berman et al., 2016, 2014a; Pereira et al., 2019) using training sets of different sizes. **a-c**, Mean and 95% interval of the bootstrap distribution for local (**a**), global (**b**), and temporal (**c**) structure preservation (for local and global structure n = 5 trials $\times$ 5 replicates = 25 per algorithm for each training set size; for temporal structure n = 5 trials $\times$ 50 subsamples = 250 per algorithm for each training set size).

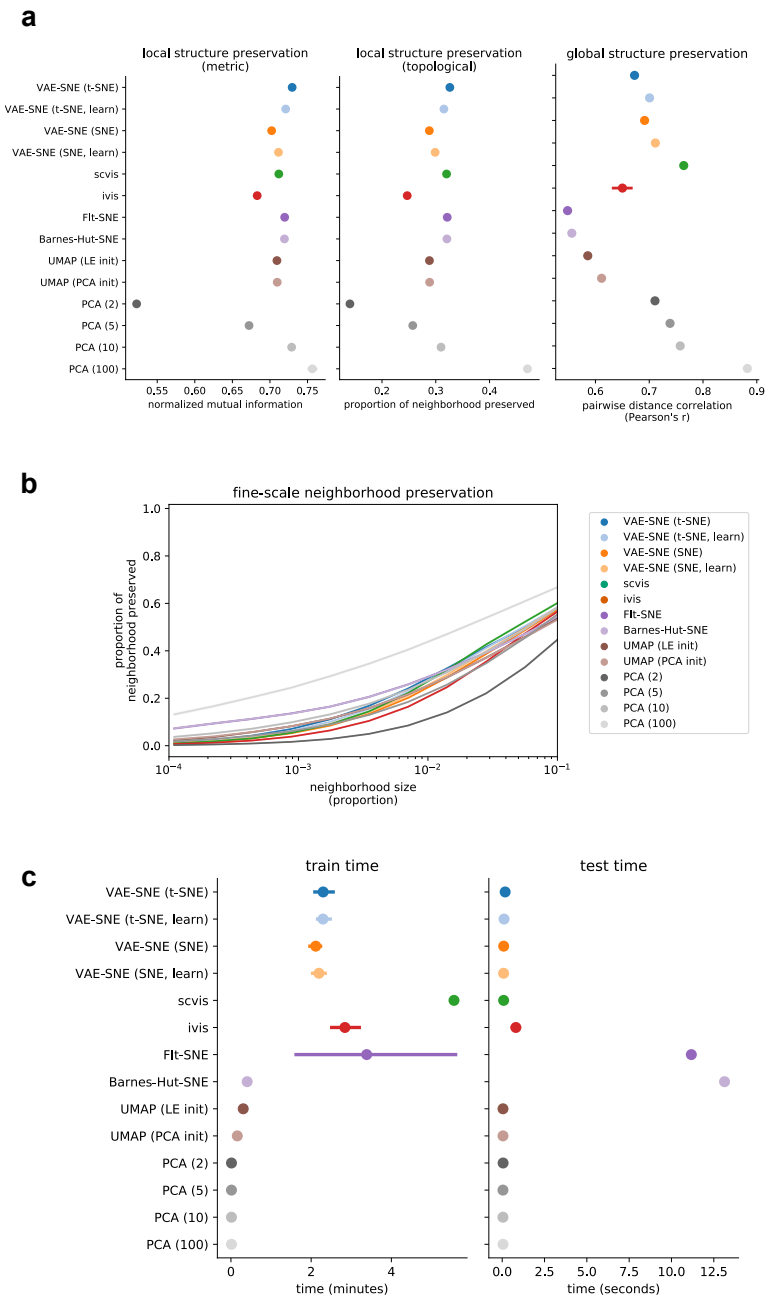

Figure B.5. Dimension reduction performance for the single-cell RNA-seq dataset. Plots show performance comparisons for the single-cell RNA-seq dataset from La Manno et al. (2018) using the entire dataset. **a**, Mean and 95% interval of the bootstrap distribution for local and global structure preservation (for each metric n = 5 trials $\times$ 5 replicates = 25 per algorithm). **b**, Mean and 95% interval of the bootstrap distribution for fine-scale structure preservation across multiple neighbor sizes (as a proportion of the total embedding size; n = 14 neighborhood sizes $\times$ 5 trials $\times$ 5 replicates = 350 per algorithm). **c**, Elapsed time for embedding the training set and re-embedding the training set as a “test" set with each algorithm (for each metric n = 5 trials $\times$ 5 replicates = 25 per algorithm).

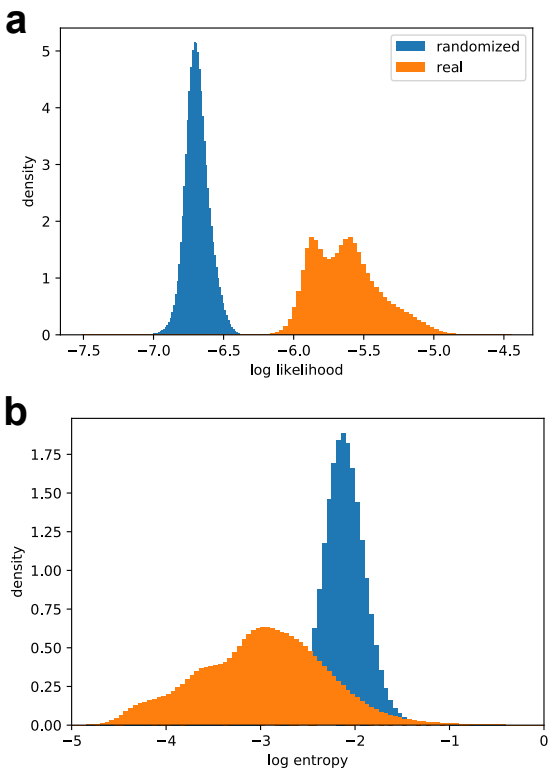

Figure B.6. Likelihood and entropy distributions. **a**, Histograms of the log likelihood scores from the decoder (Eq. 3.1b; distortion) for real and randomized data (n = 232,000 for each distribution). **b**, Histograms of the log entropy from the approximate posterior (Eq. B.3d) for real and randomized data (n = 232,000 for each distribution).

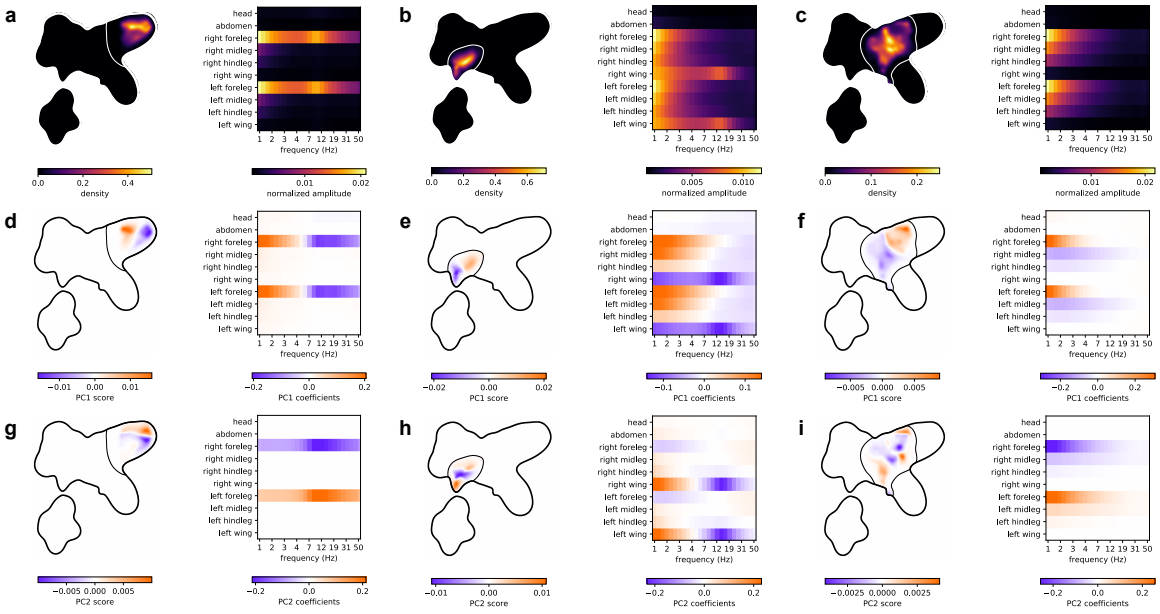

Figure B.7. High-level behavioral clusters. Visualizations describing the manually-grouped high-level clusters for anterior grooming (**a**,**e**,**g**), wing movements (**b**,**e**,**h**) and small/slow leg movements (**c**,**f**,**i**). **a-c**, The 2-D posterior probability density for each cluster (left), where contours are the largest 90% probability density contour for each cluster distribution, and the mean spectrogram for each cluster (right). **d**-**i**, The principal component scores of the spectrograms assigned to each cluster visualized within the 2-D embedding (left) and the eigenvector coefficients describing the linear contribution of each spectrogram feature (right) for the principal component score.

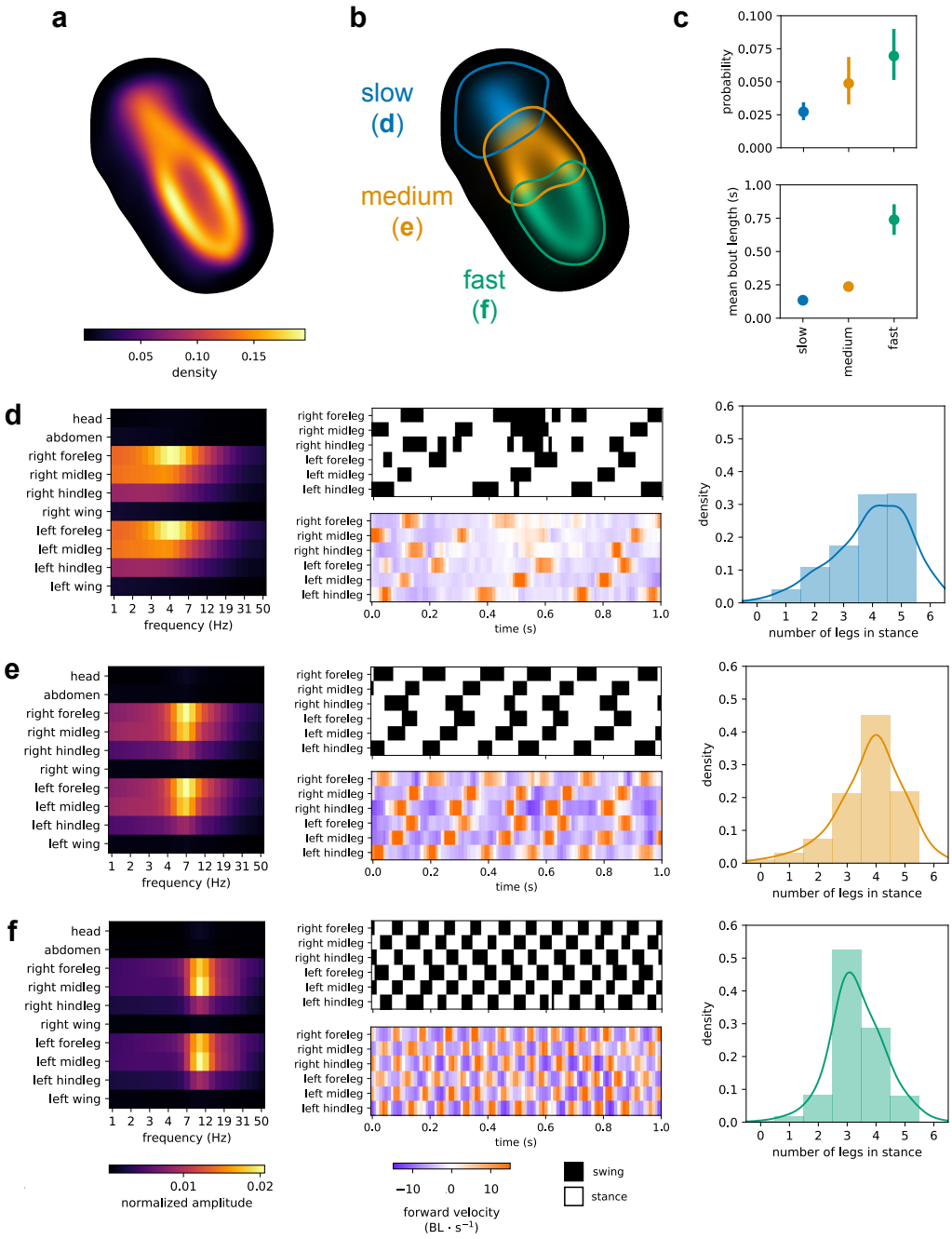

Figure B.8. Low-level locomotion clusters. Visualizations describing the low-level clusters within the high-level locomotion cluster. **a-b**, The 2-D posterior probability density for the high-level cluster (**a**) and for each low-level cluster (**b**), where letters for each cluster label correspond to panels **d-f**. Contours are the largest 90% probability density contour for each cluster distribution. **c**, Mean and 95% bootstrap intervals of the marginal (stationary) probability and mean bout length for each low-level cluster (n = 59 per cluster). **d-f**, The mean spectrogram (left), example time segments (middle) showing forward velocity of each leg measured in body lengths (BL) per second and swing (forward velocity $> 0\,\mathrm{BL} \cdot \mathrm{s}^{-1}$) or stance (forward velocity $\leq 0\,\mathrm{BL} \cdot \mathrm{s}^{-1}$) classification, and histograms (right) showing the number of legs classified as stance in each timestep assigned to each cluster (n = 0.57 million for slow, **d**; n = 1.03 million for medium, **e**; and n = 1.47 million for fast, **f**) — where the label for each panel in **d-f** corresponds to a cluster label in panel **b**. Example videos for these low-level clusters are shown in Video S2.

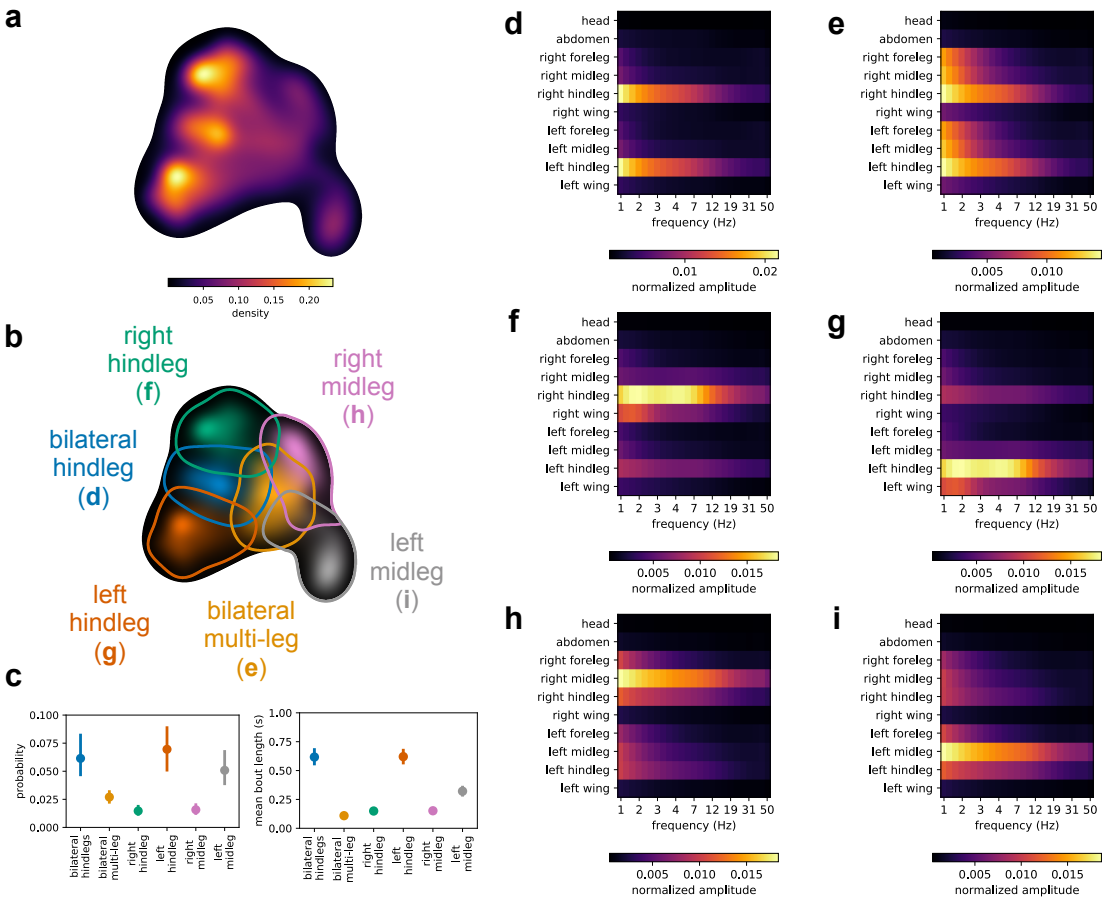

Figure B.9. Low-level posterior grooming clusters. Visualizations describing the low-level clusters within the high-level posterior grooming cluster. **a-b**, The 2-D posterior probability density for the high-level cluster (**a**) and for each low-level cluster (**b**), where letters for each cluster label correspond to panels **d-i**. Contours are the largest 90% probability density contour for each cluster distribution. **c**, Mean and 95% bootstrap intervals of the marginal (stationary) probability and mean bout length for each low-level cluster (n = 59 per cluster). **d-i**, The mean spectrogram for each cluster — where the label for each panel in **d-i** corresponds to a cluster label in panel **b**. Example videos for these low-level clusters are shown in Video S4.

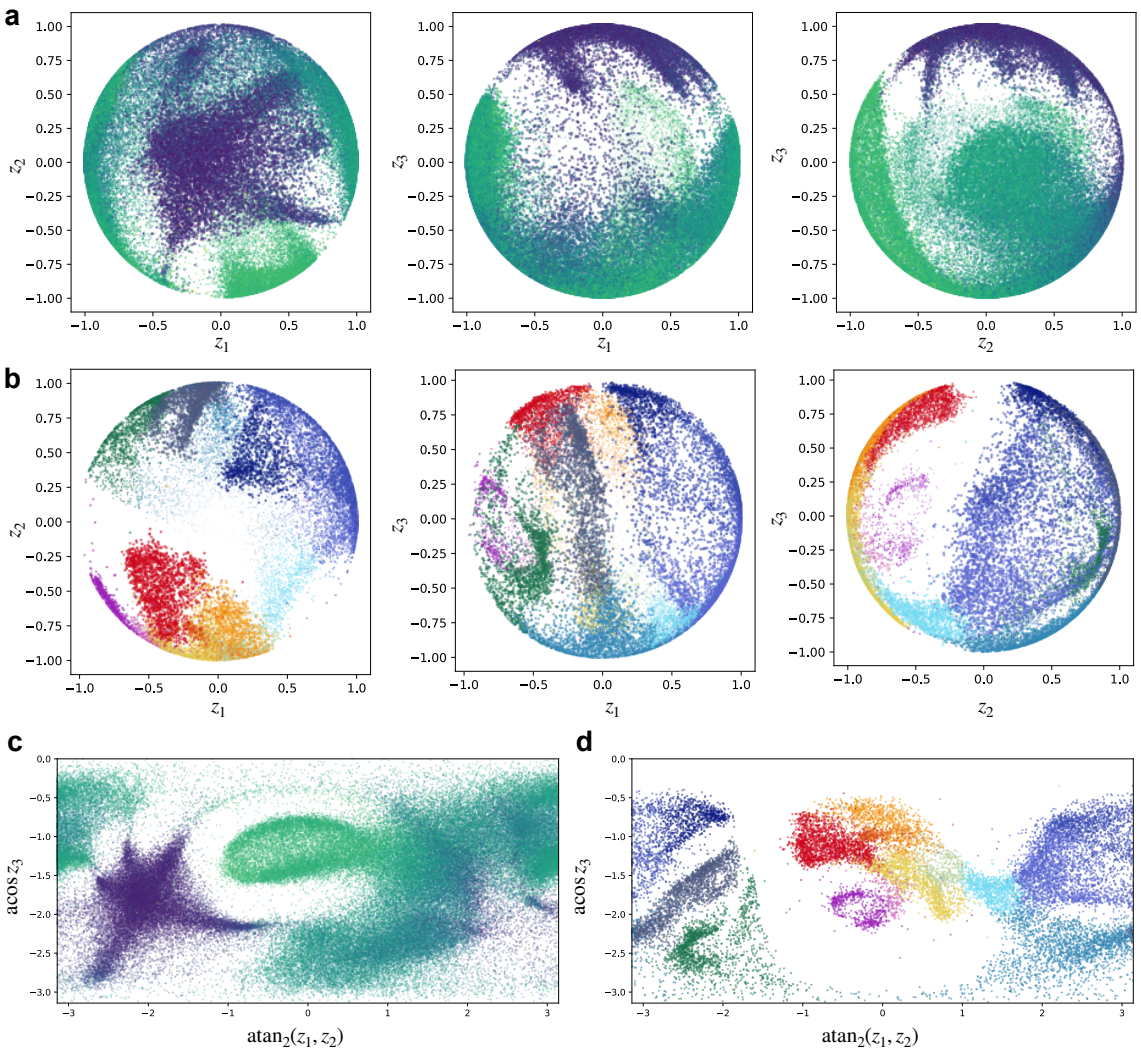

Figure B.10. Spherical embeddings with von Mises-Fisher VAE-SNE. **a-b**, Spherical embeddings using VAE-SNE with a von Mises-Fisher similarity kernel (Appendix B.3.1) of the posture dynamics dataset (**a**; Video S8) from Berman et al. (2016, 2014a); Pereira et al. (2019) and the single-cell RNA-seq dataset (**b**; Video S9) from La Manno et al. (2018). **c-d**, Stereographic (planar) projections of the spherical embeddings from **a-b**. Colors for **a-d** are the same as in Fig. 3.2 (total amplitude and cell type).

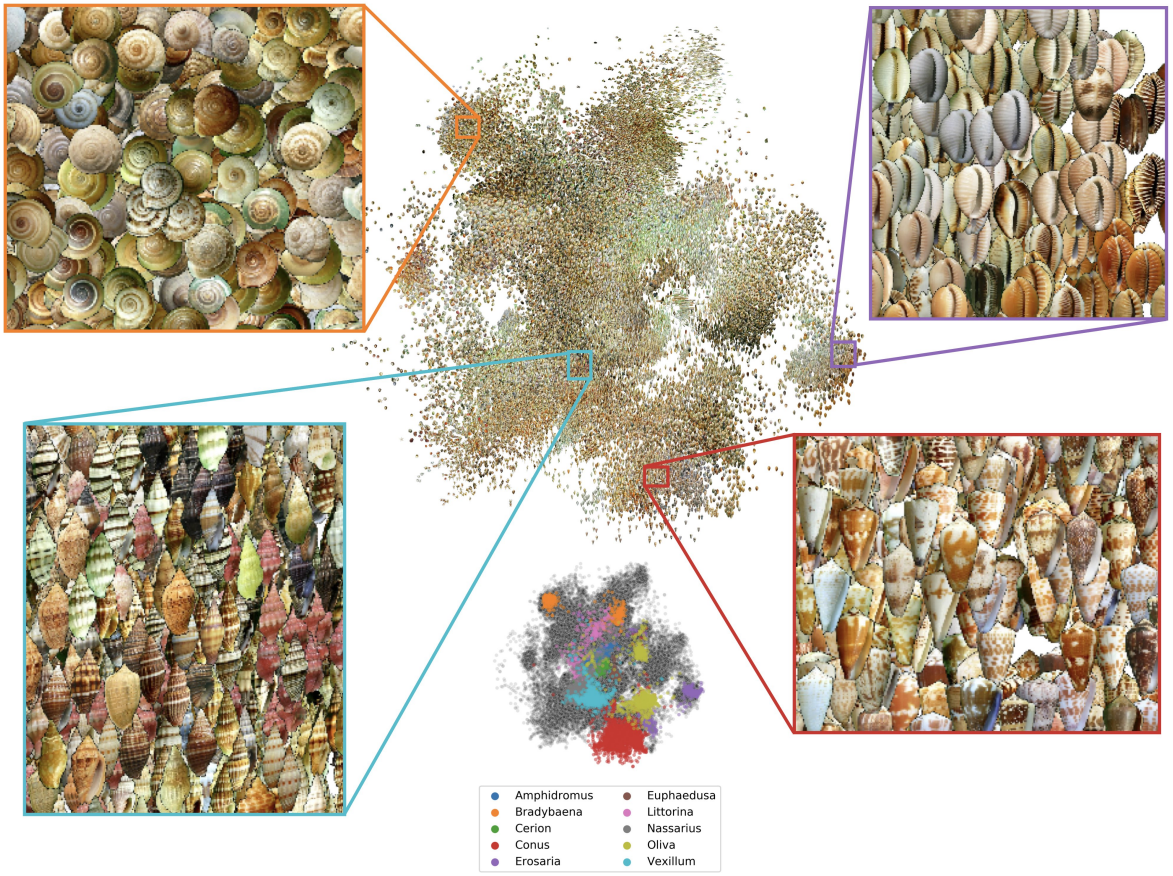

Figure B.11. Embedding shell images. Shell images from Q. Zhang et al. (2019) embedded in two dimensions using convolutional VAE-SNE. Insets illustrate example regions of perceptually similar images from the taxonomic genera *Bradybaena* (land snails; top-left), *Erosaria* (cowries; top-right), *Vexillum* (sea snails; bottom-left), and *Conus* (cone snails; bottom-right). Scatter plot (bottom-center) shows the 10 most common genera in the dataset.

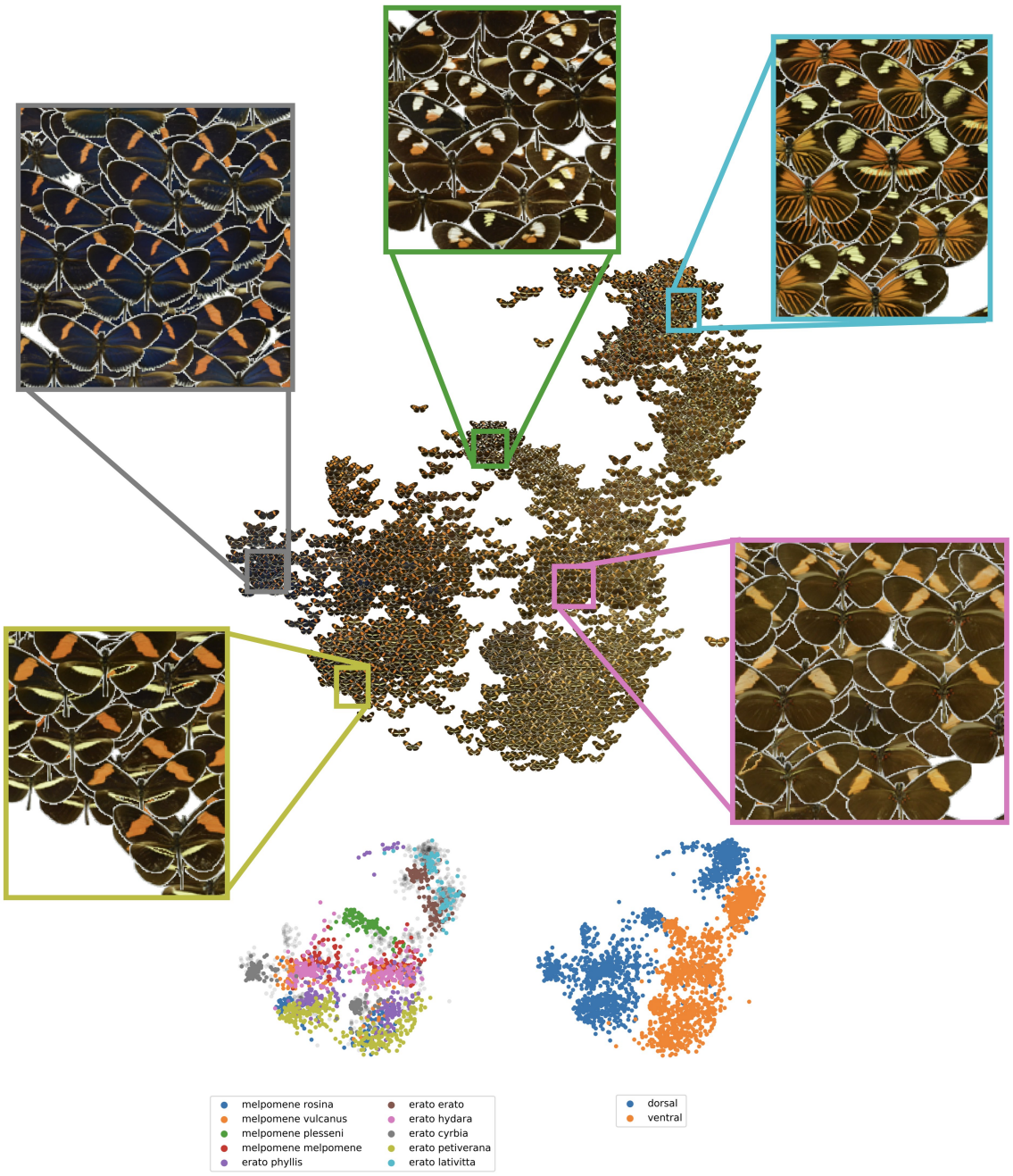

Figure B.12. Embedding butterfly images. Butterfly (*Heliconius spp.*) images from J. F. H. Cuthill et al. (2019) embedded in two dimensions using convolutional VAE-SNE. Insets show example regions of perceptually similar subspecies (top). Scatter plots (bottom) show labels for the 10 most common subspecies in the dataset (bottom-left) and the image viewpoint relative to the specimen's dorso-ventral body axis (bottom-right).

Figure Video S1. Video segments labeled with VAE-SNE. Randomly selected video segments ($1/2\times$ speed) labeled with VAE-SNE illustrating the temporal dynamics of movements through the behavioral space and transitions between high-level clusters within the distribution. **a**, https://youtu.be/JlbSdKzvLfk; **b**, https://youtu.be/uWScG_UuzRQ; **c**, https://youtu.be/T8e_JSoCwMA

Figure Video S2. Samples from the locomotion cluster. Randomly sampled videos ($1/3\times$ speed) from the locomotion cluster showing: **a**, slow walking (https://youtu.be/hB3JIRF2JGQ); **b**, medium walking (https://youtu.be/kNHGJypOGhs); and **c**, fast walking (https://youtu.be/A2sLtgYhHGc). Red lines show the posture tracking data for all 32 keypoints.

Figure Video S3. Samples from the anterior grooming cluster. Randomly sampled videos ($1/3\times$ speed) from one of the anterior grooming clusters (https://youtu.be/0MT3lb2bJro). Red lines show the posture tracking data for all 32 keypoints.

Figure Video S4. Samples from the posterior grooming cluster. Randomly sampled videos ($1/3\times$ speed) from the posterior grooming cluster showing: **a**, bilateral hindleg grooming (https://youtu.be/O_Tyf4pEQMo); **b**, right hindleg grooming (https://youtu.be/VTIwZp6d6b4); **b**, left midleg grooming (https://youtu.be/0vJvAINbfjw). Red lines show the posture tracking data for all 32 keypoints.

Figure Video S5. Samples from the wing movements cluster. Randomly sampled videos ($1/3\times$ speed) from the wing movements cluster showing: **a**, wing extensions (https://youtu.be/lE31SeJ7ehY); and **b**, wing flicks (https://youtu.be/nsgnFbrk090). Red lines show the posture tracking data for all 32 keypoints.

Figure Video S6. Samples from the small/slow leg movements cluster. Randomly sampled videos ($1/3\times$ speed) from the small/slow leg movement cluster showing: **a**, small leg movements (https://youtu.be/ARkH1uvPBnQ); **b**, slow leg movements (https://youtu.be/hwL7ovNjbBQ); **c**, small left midleg movements (https://youtu.be/o8vxtgwzx9Q) Red lines show the posture tracking data for all 32 keypoints.

Figure Video S7. Samples from the idle cluster. Randomly sampled videos ($1/3\times$ speed) from the idle cluster (https://youtu.be/0wbdqmuCe_g). Red lines show the posture tracking data for all 32 keypoints.

Figure Video S8. Spherical embedding of the posture dynamics dataset. Rotating view of the posture dynamics dataset (https://youtu.be/QcDUlQUOvdo) from Berman et al. (2016, 2014a); Pereira et al. (2019) embedded on a 3-D sphere using von Mises-Fisher VAE-SNE. Colors are the same as in Fig. 3.2 (total amplitude).

Figure Video S9. Spherical embedding of the single-cell RNA-seq dataset. Rotating view of the single-cell RNA-seq dataset (https://youtu.be/jyIWB6-qye0) from La Manno et al. (2018) embedded on a 3-D sphere using von Mises-Fisher VAE-SNE. Colors are the same as in Fig. 3.2 (cell type).

B.1 Variational autoencoders and the evidence lower bound

B.1.1 VAEs as approximate Bayesian inference

As is common to most dimensionality reduction algorithms, we seek to model a high-dimensional data distribution $p(\mathbf{x})$ using a low dimensional latent distribution $p(\mathbf{z})$. Variational autoencoders (VAEs) are one such model that combines both modeling and inference by defining a joint distribution between a latent variable $\mathbf{z}$ and observed samples $\mathbf{x}$. We can accomplish this using a generative model that maps samples from the low-dimensional latent distribution to the high-dimensional data distribution using a set of shared parameters $\boldsymbol{\theta}$, which can take the form of a deep neural network model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathrm{DNN}_{\boldsymbol{\theta}}(\mathbf{z})$ with some prior over latent distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$. We then wish to find the model parameters $\boldsymbol{\theta}$ that maximize the joint likelihood, which can be written as:

$$\arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}). \tag{B.1}$$

Although, to compute the low-dimensional distribution for the data, we then need to derive the latent posterior for the model $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. This can be derived from the likelihood using Bayes' rule:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}. \tag{B.2}$$

However, computing the integral in Eq. B.2 $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z}$ is not tractable in practice. Therefore, we require a way to approximate this latent posterior distribution, which is the exact problem for which VAEs provide a tractable solution.

Like other VAE models (Burda et al., 2015; Dieng, Kim, et al., 2019; Dilokthanakul et al., 2016; Ding et al., 2018; Kingma et al., 2014; Kingma & Welling, 2013), VAE-SNE performs dimensionality reduction by nonlinearly mapping observed high-dimensional data vectors $\mathbf{x}$ to a low-dimensional embedding $\mathbf{z}$ using a deep neural network (DNN) as an encoder function $\mathrm{DNN}_{\boldsymbol{\phi}} : \mathbf{x} \rightarrow \mathbf{z}$ (Eq. B.3e) with the goal of learning an approximate posterior over the latent distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ (Eq. B.3d), where the parameters of the approximate posterior are learned as a function of the data (Eq. B.3e) and the encoder parameters $\boldsymbol{\phi}$ are then shared across observed samples — known as *amortization*. The model then maps latent vectors sampled from the low-dimensional embedding (Eq. B.3c) to reconstruct the original high-dimensional space $\mathrm{DNN}_{\boldsymbol{\theta}} : \mathbf{z} \rightarrow \tilde{\mathbf{x}}$ (Eq. B.3a) using a generative decoder function we defined earlier (rewritten in Eq. B.3b). More precisely:

$$\tilde{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) \tag{B.3a}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{L}\left(\mathbf{x}|\mathrm{DNN}_{\boldsymbol{\theta}}(\mathbf{z})\right) \tag{B.3b}$$

$$\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \tag{B.3c}$$

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)) \tag{B.3d}$$

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2) = \mathrm{DNN}_{\boldsymbol{\phi}}(\mathbf{x}) \tag{B.3e}$$

where $\mathcal{L}(\mathbf{x}|\cdot)$ is a user-selected likelihood function parameterized by the decoder function $\mathrm{DNN}_{\boldsymbol{\theta}}(\mathbf{z})$, and $\mathcal{N}(\cdot|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ is a multivariate Gaussian whose parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$ are a specified by the encoder function $\mathrm{DNN}_{\boldsymbol{\phi}}(\mathbf{x})$.

B.1.2 Deriving the evidence lower bound

After defining the generative model, we then wish to optimize the parameters of the encoder $\boldsymbol{\phi}$ and decoder $\boldsymbol{\theta}$ — given a set of observed samples from a data distribution $\mathbf{x} \sim p(\mathbf{x})$ — so that the approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ matches closely with the true latent posterior from the generative decoder, or $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. In other words, we wish to minimize the divergence between the two distributions, or:

$$\arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] = \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\, d\mathbf{z}. \tag{B.4}$$

However, as we have already established, computing the true posterior is intractable, so researchers have derived a lower bound known as the evidence lower bound, or ELBO (Kingma & Welling, 2013), to approximate this objective. The ELBO can be derived directly from Eq.B.4 (Adams, 2020), which is written as:

$$\begin{aligned}
\mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} && \text{(B.5a)}\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) + \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} - \log p_{\boldsymbol{\theta}}(\mathbf{x}) && \text{(B.5b)}\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) + \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} - \int \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) p_{\boldsymbol{\theta}}(\mathbf{x})\, d\mathbf{z} && \text{(B.5c)}\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) + \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) p_{\boldsymbol{\theta}}(\mathbf{x})}\, d\mathbf{z} && \text{(B.5d)}\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] && \text{(B.5e)}\\
&= \log p_{\boldsymbol{\theta}}(\mathbf{x}) - \mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi}). && \text{(B.5f)}
\end{aligned}$$

Because the Kullback-Leibler divergence is strictly non-negative, the ELBO is then a lower bound on the log marginal likelihood. However, The ELBO can also be derived by applying Jensen's inequality, as is more common in the literature (Kingma & Welling, 2013), to directly calculate a lower bound on the log marginal likelihood, or:

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \log \int p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\, d\mathbf{z} && \text{(B.6a)}\\
&= \log \int p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\, d\mathbf{z} && \text{(B.6b)}\\
&= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] && \text{(B.6c)}\\
&\geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] = \mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi}). && \text{(B.6d)}
\end{aligned}$$

To learn the latent distribution given the model and the data, the ELBO is then maximized to optimize the model parameters. Here we write this as a minimization of the negative ELBO, which can be further decomposed into separate terms for the log-likelihood and the divergence

between the approximate posterior and the prior over the latent distribution, or:

$$\begin{aligned}
\arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} -\mathrm{ELBO}(\boldsymbol{\theta},\boldsymbol{\phi}) &= \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}\right] && \text{(B.7a)}\\
&= \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}\right] && \text{(B.7b)}\\
&= \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z})}\right] && \text{(B.7c)}\\
&= \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \underbrace{\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]}_{\text{likelihood}} + \underbrace{\mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z})]}_{\text{divergence}}. && \text{(B.7d)}
\end{aligned}$$

The derivation of the ELBO has also been discussed at length elsewhere (e.g., Alemi et al. 2016, 2017; Burda et al. 2015; Dilokthanakul et al. 2016; Ding et al. 2018; Kingma et al. 2014; Kingma & Welling 2013; also see Kingma & Welling 2019 for a comprehensive introduction).

B.1.3 Importance-weighted ELBO

While we use only a single Monte Carlo sample from the approximate posterior per training batch, we also include a hyperparameter for multiple samples per training batch using the importance-weighted ELBO from Burda et al. (2015), which modifies how the expectation in Eq. B.7c is calculated to produce a tighter bound on the loss by implicitly increasing the complexity of the posterior (Cremer et al., 2017). However, we did not see any obvious performance improvements when using the importance-weighted objective, and increasing the number of Monte Carlo samples per batch also increases training time. The general utility of calculating a tighter bound is also unclear (Rainforth et al., 2018) but this may be related to the generalization ability of the model. We leave further exploration of this hyperparameter for future work.

B.2 Stochastic neighbor regularization

By modeling local neighborhoods as probability distributions, known as pairwise similarity kernels, and then minimizing the divergence between the neighborhood distributions in high- and low-dimensional space, we preserve more local structure within the low-dimensional embedding than a standard VAE (Ding et al., 2018). To compute these pairwise similarities, we largely follow Hinton & Roweis (2003) and van der Maaten & Hinton (2008) by modeling local neighborhoods as the probability of transitioning from a landmark point to its nearby neighbors when performing

a random walk initialized from the landmark.

High-dimensional transition probabilities To accomplish this, pairwise transition probabilities in high-dimensional space $t(\mathbf{x}_j|\mathbf{x}_i)$ are modeled by applying a Gaussian kernel to convert the pairwise distances between data points $d(\mathbf{x}_i, \mathbf{x}_j)$ into conditional probabilities — with self transitions set to $t(\mathbf{x}_i|\mathbf{x}_i) = 0$. While Ding et al. (2018) use these asymmetric conditional probabilities $t(\mathbf{x}_j|\mathbf{x}_i)$ directly for the high-dimensional similarities, van der Maaten & Hinton (2008) show that symmetrizing the pairwise similarities so that $p(\mathbf{x}_j|\mathbf{x}_i) = p(\mathbf{x}_i|\mathbf{x}_j)$ reduces susceptibility to outliers, which can become ill-determined in the low-dimensional embedding with an asymmetric kernel. Therefore, we use the symmetrized conditional probabilities, which are computed as:

$$p(\mathbf{x}_j|\mathbf{x}_i) = \frac{t(\mathbf{x}_j|\mathbf{x}_i) + t(\mathbf{x}_i|\mathbf{x}_j)}{\sum_n t(\mathbf{x}_n|\mathbf{x}_i) + t(\mathbf{x}_i|\mathbf{x}_n)} \tag{B.8a}$$

$$t(\mathbf{x}_j|\mathbf{x}_i) = \frac{\mathcal{N}(\mathbf{x}_j|\mathbf{x}_i, \varsigma_i^2)}{\sum_n \mathcal{N}(\mathbf{x}_n|\mathbf{x}_i, \varsigma_i^2)} = \frac{\exp\left(-d(\mathbf{x}_i, \mathbf{x}_j)^2/2\varsigma_i^2\right)}{\sum_n \exp\left(-d(\mathbf{x}_i, \mathbf{x}_n)^2/2\varsigma_i^2\right)}, \tag{B.8b}$$

for $n = 1, \ldots, N$ and $n \neq i$, where $d(\cdot, \cdot)$ is a user-selected distance metric, such as the Euclidean distance. The landmark data point $\mathbf{x}_i$ can then be considered the mean, and ς_i^2 is the variance of the Gaussian kernel describing the local neighborhood around $\mathbf{x}_i$ — thereby assigning more probability mass to nearby neighbors. The variance ς_i^2 is selected for each data point via binary search such that $2^{\mathrm{H}_i} \approx \mathrm{P}$, where P is the desired perplexity (a user-defined hyperparameter), 2^{H_i} is the perplexity of the kernel for the ith data point, which approximately corresponds to the number of nearest neighbors considered by the kernel, and H_i is the Shannon entropy in bits, or:

$$\mathrm{H}_i = \sum_j t(\mathbf{x}_j|\mathbf{x}_i) \log_2 t(\mathbf{x}_j|\mathbf{x}_i). \tag{B.9}$$

Low-dimensional transition probabilities The low-dimensional similarities $q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)$ are then calculated according to Hinton & Roweis (2003) and van der Maaten & Hinton (2008) using a kernel function $w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)$ to convert pairwise distances into conditional probabilities:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = \frac{w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)}{\sum_n w_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{z}_i)}. \tag{B.10}$$

As in high-dimensional space, self transitions are set to $q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{z}_i) = 0$. Here we test two kernel functions for preserving Euclidean similarities.

t-SNE kernel First is the heavy-tailed Student's *t*-distributed kernel used for the t-SNE algorithm (van der Maaten & Hinton, 2008) with the log probability function written as:

$$\log w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = \log \mathcal{T}(\mathbf{z}_j|\mathbf{z}_i, \nu_i, \tau_i) = -\left(\frac{\nu_i + 1}{2}\right) \log\left(1 + \frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{\tau_i \nu_i}\right) - Z_i \tag{B.11a}$$

$$Z_i = \log \tau_i + \frac{\log(\nu_i \pi)}{2} + \log \Gamma\left(\frac{\nu_i}{2}\right) + \log \Gamma\left(\frac{\nu_i + 1}{2}\right), \tag{B.11b}$$

where τ_i is the scale, ν_i is the degrees of freedom, which varies the heavy-tails of the kernel, and $\Gamma(\cdot)$ is the gamma function. We write this as a log probability to more clearly show the relationship with the similarity loss term derived later in this section (Eq. B.14c). The Student's *t*-distribution is used primarily to alleviate the "crowding problem" (van der Maaten & Hinton, 2008) that can occur with other nonlinear embedding algorithms, including the original SNE algorithm (Hinton & Roweis, 2003), where points are too densely packed in the low-dimensional space and moderately distant points are "crushed" together as an artifact of the embedding algorithm.

SNE kernel Secondly, we test a Gaussian kernel — the kernel used for the original SNE algorithm (Hinton & Roweis, 2003; van der Maaten & Hinton, 2008) — with the log probability function:

$$\log w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = \log \mathcal{N}(\mathbf{z}_j|\mathbf{z}_i, \eta_i^2) = \frac{-\|\mathbf{z}_i - \mathbf{z}_j\|^2}{2\eta_i^2} + Z_i \tag{B.12a}$$

$$Z_i = \log \eta_i + \log \sqrt{2\pi}, \tag{B.12b}$$

where η_i^2 is the variance.

Setting the kernel parameters The kernel parameters for the low-dimensional similarities are typically set to a constant value, such as $\tau_i = \nu_i = \eta_i = 1$ (van der Maaten & Hinton, 2008), or are scaled linearly with the dimensionality of the latent embedding (van der Maaten, 2009), but we also test similarity kernels where these parameters are learned for each data point, parameterized by the encoder $\mathrm{DNN}_{\boldsymbol{\phi}}(\mathbf{x})$ — an idea proposed by van der Maaten (2009). When the kernel parameters are constant across all data points, the log normalization terms (Eqs. B.11b, B.12b) used for calculating the log probabilities can be omitted as an additive constant that has no effect on the calculations after normalization. However, this term is potentially important for optimization when learning these parameters as a function of each data point, so we include it in our calculations.

Reinterpreting the similarity loss term To maximize numerical stability when optimizing the similarity term, we substitute the cross-entropy between the high-dimensional and low-dimensional similarities $\mathrm{H}[p(\mathbf{x}_j|\mathbf{x}_i), q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)]$, which is proportional to the Kullback-Leibler divergence and, after dropping the expectation, can be derived as follows:

$$\sum_j \mathrm{SNE}_{j|i}(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\phi}) = \sum_j \mathbb{KL}[p(\mathbf{x}_j|\mathbf{x}_i) \| q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)] \tag{B.13a}$$

$$= \sum_j p(\mathbf{x}_j|\mathbf{x}_i) \log \frac{p(\mathbf{x}_j|\mathbf{x}_i)}{q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)} \tag{B.13b}$$

$$= \underbrace{\sum_j p(\mathbf{x}_j|\mathbf{x}_i) \log p(\mathbf{x}_j|\mathbf{x}_i)}_{-\text{entropy}} \underbrace{- \sum_j p(\mathbf{x}_j|\mathbf{x}_i) \log q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)}_{\text{cross entropy}} \tag{B.13c}$$

$$= \text{constant} - \sum_j p(\mathbf{x}_j|\mathbf{x}_i) \log q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) \tag{B.13d}$$

$$\propto - \sum_j p(\mathbf{x}_j|\mathbf{x}_i) \log q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = \sum_j \mathrm{H}[p(\mathbf{x}_j|\mathbf{x}_i), q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)]. \tag{B.13e}$$

Consequently, the Kullback-Leibler divergence for the similarity term can be reinterpreted as the cross-entropy between the pairwise similarities up to an additive constant (the negative entropy of the high-dimensional similarities), which can be omitted for the purposes of optimization. To further improve numerical stability for this computation, the cross-entropy is decomposed into attractive and repulsive forces using the unnormalized similarities (following Ding et al. 2018;

Kobak & Berens 2019), which is written as:

$$-\sum_j p(\mathbf{x}_j|\mathbf{x}_i)\log q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = -\sum_j p(\mathbf{x}_j|\mathbf{x}_i)\log\frac{w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)}{\sum_n w_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{z}_i)} \tag{B.14a}$$

$$= -\sum_j p(\mathbf{x}_j|\mathbf{x}_i)\log w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) + \sum_j p(\mathbf{x}_j|\mathbf{x}_i)\log\sum_j w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) \tag{B.14b}$$

$$= \underbrace{-\sum_j p(\mathbf{x}_j|\mathbf{x}_i)\log w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)}_{\text{attract}} + \underbrace{\log\sum_j w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i)}_{\text{repel}}. \tag{B.14c}$$

This may also help to clarify why we wrote the low-dimensional kernels as log-probability functions in Eqs. B.11a, B.12a.

B.3 Extensions of VAE-SNE

B.3.1 Spherical embeddings with a von Mises-Fisher kernel

In addition to embeddings with Euclidean geometry, we introduce a version of VAE-SNE that uses polar geometry and embeds high-dimensional data on the surface of a 3D unit sphere. We calculate the high-dimensional similarities according to Appendix B.2, but we alter the calculations for the transition probabilities by using the cosine similarity for the high-dimensional pairwise metric. After normalization, this is equivalent to using a (hyper)spherical von Mises-Fisher distribution as the similarity kernel, or:

$$t(\mathbf{x}_j|\mathbf{x}_i) = \frac{\mathcal{F}(\mathbf{x}_j|\mathbf{x}_i,\kappa_i)}{\sum_n \mathcal{F}(\mathbf{x}_n|\mathbf{x}_i,\kappa_i)} = \frac{\exp\left(\hat{\mathbf{x}}_i\cdot\hat{\mathbf{x}}_j^{\mathrm{T}}\kappa_i\right)}{\sum_n \exp\left(\hat{\mathbf{x}}_i\cdot\hat{\mathbf{x}}_n^{\mathrm{T}}\kappa_i\right)}, \tag{B.15}$$

where $\hat{\mathbf{x}}_i = \mathbf{x}_i/\|\mathbf{x}_i\|^2$ and κ_i is the concentration parameter (the inverse variance $\kappa_i = \varsigma_i^{-2}$), which is selected using binary search to match the perplexity to a desired value (see Appendix B.2 for details). We then calculate the low-dimensional similarities using a 3D von Mises-Fisher

kernel to create a spherical embedding:

$$\log w_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{z}_i) = \log \mathcal{F}(\mathbf{z}_j|\mathbf{z}_i, \rho_i) = \hat{\mathbf{z}}_i \cdot \hat{\mathbf{z}}_j^{\mathrm{T}} \rho_i + Z_i \tag{B.16a}$$

$$Z_i = \log \rho_i - \log \sinh \rho_i - \log 4\pi \tag{B.16b}$$

where $\hat{\mathbf{z}}_i = \mathbf{z}_i / \|\mathbf{z}_i\|^2$ and ρ_i is the concentration parameter (inverse variance). The log normalization term (Eq. B.16b) can be omitted when ρ_i is set to a constant, but we include it for the purposes of optimizing ρ_i as a function of each data point.

The idea of using spherical embeddings for dimensionality reduction has been explored previously with the von Mises-Fisher stochastic neighbor embedding (VMF-SNE) algorithm (Wang & Wang, 2016) as well as more recent work by Ding & Regev (2019) who apply this type of embedding to visualize single-cell RNA-seq data. The UMAP algorithm (McInnes et al., 2018) has a similar option to embed data in polar coordinates, as well as other non-Euclidean spaces. VAEs with (hyper)spherical latent variables have also been explored extensively in the machine learning literature (Davidson et al. 2018; reviewed by Ding & Regev 2019). This type of spherical representation can be useful for data analysis, as high-dimensional vectors are often more accurately represented in polar coordinates. Similar to a heavy-tailed Student's *t* similarity kernel (van der Maaten & Hinton, 2008), a spherical von Mises-Fisher similarity kernel can also prevent "crowding" of the data toward the center of the latent coordinate system (Davidson et al., 2018; Ding & Regev, 2019), which is undesirable for visualizing data (van der Maaten & Hinton, 2008). To test this extension, we use von Mises-Fisher VAE-SNE to embed the posture dynamics dataset from Berman et al. (2016, 2014a); Pereira et al. (2019) as well as the single-cell RNA-seq dataset from La Manno et al. (2018) and visualize the embeddings across the three dimensions of the unit sphere (Fig. B.10; Video S8; Video S9). We find that the results are qualitatively similar to 2-D Euclidean embeddings of the same data (Fig. 3.2), but are instead embedded across a 3-D sphere. Despite not using a heavy-tailed similarity kernel (van der Maaten & Hinton, 2008) these spherical embeddings naturally do not exhibit any crowding problems (Davidson et al., 2018; Ding & Regev, 2019), which may make this a useful visualization tool for some scenarios.

B.3.2 Convolutional VAE-SNE for image data

We introduce a convolutional version of VAE-SNE for embedding image data from raw pixels. This version of VAE-SNE is modified by first applying a 2-D convolutional neural network $\mathrm{CNN}_{\boldsymbol{\phi}}$ — a SqueezeNet v1.1 (Iandola et al., 2016) pretrained on ImageNet (Deng et al., 2009) — to each image and then calculating the pairwise similarity using spatially-pooled feature maps from the $\mathrm{CNN}_{\boldsymbol{\phi}}$ output. The high-dimensional transition probabilities (Appendix B.2) are then calculated using a Gaussian kernel:

$$t(\mathbf{x}_j|\mathbf{x}_i) = \frac{\exp\left(-d(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j)^2/2\varsigma_i^2\right)}{\sum_n \exp\left(-d(\hat{\mathbf{v}}_i, \hat{\mathbf{v}}_n)^2/2\varsigma_i^2\right)}, \tag{B.17}$$

where $\hat{\mathbf{v}}_i$ is a vector of spatially-pooled feature maps from the $\mathrm{CNN}_{\boldsymbol{\phi}}$ output, or $\hat{\mathbf{v}}_i = \mathrm{CNN}_{\boldsymbol{\phi}}(\mathbf{x}_i)$. The approximate posterior is then calculated as a nonlinear function of the pooled feature maps $\mathrm{DNN}_{\boldsymbol{\phi}} : \hat{\mathbf{v}}_i \rightarrow \mathbf{z}_i$, which is written as $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}_i) = \mathcal{N}(\mathbf{z}|\mathrm{DNN}_{\boldsymbol{\phi}}(\hat{\mathbf{v}}_i))$. For the decoder we use a feed-forward network $\mathrm{DNN}_{\boldsymbol{\theta}} : \mathbf{z}_i \rightarrow \tilde{\mathbf{v}}_i$ as before, where $\tilde{\mathbf{v}}_i$ is a reconstruction of the $\mathrm{CNN}_{\boldsymbol{\phi}}$ output $\hat{\mathbf{v}}_i$. We then apply mean squared error between the pooled feature maps and the reconstruction as the likelihood function for the distortion loss (Eq. 3.1b). A convolutional decoder could also be used to fully reconstruct the raw image pixels, but we found simply reconstructing the pooled feature maps to be effective for visualizing the distribution of images in two dimensions. To demonstrate the utility of convolutional VAE-SNE, we embed natural history image datasets of both shells (Q. Zhang et al., 2019) and (*Heliconius spp.*) butterflies (J. F. H. Cuthill et al., 2019). We then visualize these embeddings to qualitatively assess performance of this VAE-SNE variant (Figs. B.11, B.12). We find that perceptually similar images are grouped together in the embedding based on complex sets of image features — rather than simple heuristics like color — and these groupings correspond to taxonomic relationships within the dataset, which were not explicitly included as part of the training set. This variant of VAE-SNE is functionally similar to using the perceptual distance (J. Johnson et al., 2016; Wham et al., 2019) as a similarity metric and likelihood function except that the model can be trained end-to-end with small batches of images directly using raw pixels instead of first preprocessing images to produce feature activations. These results demonstrate that VAE-SNE can be used to analyze very large image datasets, by loading images in small batches, and can also be extended to images with variable resolution, by integrating across feature map outputs from the CNN to remove the spatial

dimension — both of which are typically not possible with other dimension reduction algorithms.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Adams, R. P. (2020, June 17). *The elbo without jensen, kullback, or leibler.* `https://lips.cs.princeton.edu/the-elbo-without-jensen-or-kl/`.

Adelman, J., Moyers, S., Farine, D., & Hawley, D. (2015). Feeder use predicts both acquisition and transmission of a contagious pathogen in a north american songbird. *Proceedings of the Royal Society B*, *282*, 20151429.

Ahrens, M. B., Li, J. M., Orger, M. B., Robson, D. N., Schier, A. F., Engert, F., & Portugues, R. (2012). Brain-wide neuronal dynamics during motor adaptation in zebrafish. *Nature*, *485*(7399), 471.

Ahrens, M. B., Orger, M. B., Robson, D. N., Li, J. M., & Keller, P. J. (2013). Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, *10*(5), 413.

Akhund-Zade, J., Ho, S., O'Leary, C., & de Bivort, B. (2019). The effect of environmental enrichment on behavioral variability depends on genotype, behavior, and type of enrichment. *Journal of Experimental Biology*. Retrieved from `https://jeb.biologists.org/content/early/2019/08/14/jeb.202234` doi: 10.1242/jeb.202234

Alarcón-Nieto, G., Graving, J. M., Klarevas-Irby, J. A., Maldonado-Chaparro, A. A., Mueller, I., & Farine, D. R. (2018). An automated barcode tracking system for behavioural studies in birds. *Methods in Ecology and Evolution*, *9*(6), 1536–1547.

Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., & Murphy, K. (2017). Fixing a broken elbo. *arXiv preprint arXiv:1711.00464*.

Alisch, T., Crall, J. D., Kao, A. B., Zucker, D., & de Bivort, B. L. (2018). Maple (modular automated platform for large-scale experiments), a robot for integrated organism-handling and phenotyping. *eLife*, *7*, e37166.

Anderson, D. J., & Perona, P. (2014). Toward a science of computational ethology. *Neuron*, *84*(1), 18–31.

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., & Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5167–5176).

Andriluka, M., Pishchulin, L., Gehler, P., & Schiele, B. (2014, June). 2d human pose estimation: New benchmark and state of the art analysis. In *Ieee conference on computer vision and pattern recognition (cvpr).*

Aplin, L., Farine, D., Morand-Ferron, J., Cockburn, A., Thornton, A., & Sheldon, B. (2015). Experimentally induced innovations lead to persistent culture via conformity in wild birds. *Nature*, *518*, 538–541.

Aplin, L. M., Farine, D. R., Mann, R. P., & Sheldon, B. C. (2014). Individual-level personality influences social foraging and collective behaviour in wild birds. *Proceedings of the Royal Society B: Biological Sciences*, *281*(1789), 20141016.

Ayinde, B. O., & Zurada, J. M. (2018). Building efficient convnets using redundant feature pruning. *arXiv preprint arXiv:1802.07653*.

Ayroles, J. F., Buchanan, S. M., O'Leary, C., Skutt-Kakaria, K., Grenier, J. K., Clark, A. G., ... de Bivort, B. L. (2015). Behavioral idiosyncrasy reveals genetic control of phenotypic variability. *Proceedings of the National Academy of Sciences*, *112*(21), 6706–6711.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, *abs/1511.00561*. Retrieved from `http://arxiv.org/abs/1511.00561`

Bala, P. C., Eisenreich, B. R., Yoo, S. B. M., Hayden, B. Y., Park, H. S., & Zimmermann, J. (2020). Openmonkeystudio: automated markerless pose estimation in freely moving macaques. *bioRxiv*.

Bath, D. E., Stowers, J. R., Hörmann, D., Poehlmann, A., Dickson, B. J., & Straw, A. D. (2014). Flymad: rapid thermogenetic control of neuronal activity in freely walking drosophila. *Nature methods*, *11*(7), 756.

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W., Ng, L. G., ... Newell, E. W. (2019). Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, *37*(1), 38.

Berger-Wolf, T., Rubenstein, D., Stewart, C., Holmberg, J., Parham, J., & Crall, J. (2015). Ibeis: Image-based ecological information system: From pixels to science and conservation. In *Bloomberg data for good exchange conference.* New York, NY, USA.

Berman, G. J. (2018). Measuring behavior across scales. *BMC biology*, *16*(1), 23.

Berman, G. J., Bialek, W., & Shaevitz, J. W. (2016). Predictability and hierarchy in drosophila behavior. *Proceedings of the National Academy of Sciences*, *113*(42), 11943–11948.

Berman, G. J., Choi, D. M., Bialek, W., & Shaevitz, J. W. (2014a). Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, *11*(99), 20140672.

Berman, G. J., Choi, D. M., Bialek, W., & Shaevitz, J. W. (2014b). Mapping the structure of drosophilid behavior. *bioRxiv*, 002873.

Bierbach, D., Laskowski, K. L., & Wolf, M. (2017). Behavioural individuality in clonal fish arises despite near-identical rearing conditions. *Nature communications*, *8*, 15361.

Billings, J. C., Medda, A., Shakil, S., Shen, X., Kashyap, A., Chen, S., ... others (2017). Instantaneous brain dynamics mapped to a continuous state space. *Neuroimage*, *162*, 344–352.

Boarman, W., Beigel, M., Goodlett, G., & Sazaki, M. (1998). A passive integrated transponder system for tracking animal movements. *Wildlife Society Bulletin*, *26*, 886–891.

Boenisch, F., Rosemann, B., Wild, B., Dormagen, D., Wario, F., & Landgraf, T. (2018). Tracking all members of a honey bee colony over their lifetime using learned models of correspondence. *Frontiers in Robotics and AI*, *5*, 35. Retrieved from `https://www.frontiersin.org/article/10.3389/frobt.2018.00035` doi: 10.3389/frobt.2018.00035

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in ecology & evolution*, *24*(3), 127–135.

Bonter, D., & Bridge, E. (2011). Applications of radio frequency identification (rfid) in ornithological research: a review. *Journal of Field Ornithology*, *82*, 1–10.

Bonter, D., Zuckerberg, B., Sedgwick, C., & Hochachka, W. (2013). Daily foraging patterns in free-living birds: exploring the predation-starvation trade-off. *Proceedings of the Royal Society B*, *280*, 20123087.

Boogert, N., Farine, D., & Spencer, K. (2014). Developmental stress predicts social network position. *Biology Letters*, *10*, 20140561.

Broderick, A., & Godley, B. (1999). Effect of tagging marine turtles on nesting behaviour and reproductive success. *Animal Behaviour*, *58*, 587–591.

Brown, A. E., & De Bivort, B. (2018). Ethology as a physical science. *Nature Physics*, 1.

Brown, A. E., Yemini, E. I., Grundy, L. J., Jucikas, T., & Schafer, W. R. (2013). A dictionary of behavioral motifs reveals clusters of genes affecting caenorhabditis elegans locomotion. *Proceedings of the National Academy of Sciences*, *110*(2), 791–796.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *Ecml pkdd workshop: Languages for data mining and machine learning* (pp. 108–122).

Burda, Y., Grosse, R., & Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 160–172).

Cande, J., Namiki, S., Qiu, J., Korff, W., Card, G. M., Shaevitz, J. W., ... Berman, G. J. (2018). Optogenetic dissection of descending behavioral control in drosophila. *Elife*, *7*, e34275.

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7291–7299).

Carpenter, B., Lee, D., Brubaker, M. A., Riddell, A., Gelman, A., Goodrich, B., ... Li, P. (2017). Stan: A Probabilistic Programming Language. *J. Stat. Softw.*.

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

Chalk, M., Marre, O., & Tkacik, G. (2016). Relevant sparse codes with variational information bottleneck. In *Advances in neural information processing systems* (pp. 1957–1965).

Chambers, C., Kong, G., Wei, K., & Kording, K. (2019). Pose estimates from online videos show that side-by-side walkers synchronize movement under naturalistic conditions. *PloS one*, *14*(6), e0217861.

Chaudhuri, K., & Dasgupta, S. (2010). Rates of convergence for the cluster tree. In *Advances in neural information processing systems* (pp. 343–351).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Chen, Y., Shen, C., Wei, X.-S., Liu, L., & Yang, J. (2017). Adversarial posenet: A structure-aware convolutional network for human pose estimation. In *Proceedings of the ieee international conference on computer vision* (pp. 1212–1221).

Chien, J.-T., & Hsu, C.-W. (2017). Variational manifold learning for speaker recognition. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 4935–4939).

Cho, H., Berger, B., & Peng, J. (2018). Generalizable and scalable visualization of single-cell data using neural networks. *Cell systems*, *7*(2), 185–191.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1251–1258).

Chollet, F., et al. (2015). *Keras.* `https://github.com/fchollet/keras`. GitHub.

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Costa, A. C., Ahamed, T., & Stephens, G. J. (2019). Adaptive, locally linear models of complex dynamics. *Proceedings of the National Academy of Sciences*, *116*(5), 1501–1510. Retrieved from `https://www.pnas.org/content/116/5/1501` doi: 10.1073/pnas.1813476116

Crall, J. D., Gravish, N., Mountcastle, A. M., & Combes, S. A. (2015). Beetag: a low-cost, image-based tracking system for the study of animal behavior and locomotion. *PloS one*, *10*(9), e0136487.

Cremer, C., Morris, Q., & Duvenaud, D. (2017). Reinterpreting importance-weighted autoencoders. *arXiv preprint arXiv:1704.02916*.

Cuthill, I. C., Allen, W. L., Arbuckle, K., Caspers, B., Chaplin, G., Hauber, M. E., ... others (2017). The biology of color. *Science*, *357*(6350), eaan0221.

Cuthill, J. F. H., Guttenberg, N., Ledger, S., Crowther, R., & Huertas, B. (2019). Deep learning on butterfly phenotypes tests evolution's oldest mathematical model. *Science advances*, *5*(8), eaaw4967.

Datta, S. R., Anderson, D. J., Branson, K., Perona, P., & Leifer, A. (2019). Computational neuroethology: a call to action. *Neuron*, *104*(1), 11–24.

David, M., Auclair, Y., & Cezilly, F. (2011). Personality predicts social dominance in female zebra finches, taeniopygia guttata, in a feeding context. *Animal Behaviour*, *81*, 219–224.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., & Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

Dell, A. I., Bender, J. A., Branson, K., Couzin, I. D., de Polavieja, G. G., Noldus, L. P., ... others (2014). Automated image-based tracking and its application in ecology. *Trends in ecology & evolution*, *29*(7), 417–428.

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

Dieng, A. B., Kim, Y., Rush, A. M., & Blei, D. M. (2019). Avoiding latent variable collapse with generative skip models. In *The 22nd international conference on artificial intelligence and statistics* (pp. 2397–2405).

Dieng, A. B., Ruiz, F. J., Blei, D. M., & Titsias, M. K. (2019). Prescribed generative adversarial networks. *arXiv preprint arXiv:1910.04302*.

Dilokthanakul, N., Mediano, P. A., Garnelo, M., Lee, M. C., Salimbeni, H., Arulkumaran, K., & Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*.

Ding, J., Condon, A., & Shah, S. P. (2018). Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature Communications*, *9*(1), 2002. Retrieved from https://doi.org/10.1038/s41467-018-04368-5 doi: 10.1038/s41467-018-04368-5

Ding, J., & Regev, A. (2019). Deep generative model embedding of single-cell rna-seq profiles on hyperspheres and hyperbolic spaces. *BioRxiv*, 853457.

Donahue, J., & Simonyan, K. (2019). Large scale adversarial representation learning. In *Advances in neural information processing systems* (pp. 10542–10552).

Doudna, J. A., & Charpentier, E. (2014). The new frontier of genome engineering with crispr-cas9. *Science*, *346*(6213), 1258096.

Duane, S., Kennedy, A. D., Pendleton, B. J., & Roweth, D. (1987). Hybrid Monte Carlo. *Phys. Lett. B*, *195*(2), 216–222.

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

Ebbesen, C. L., & Froemke, R. C. (2020). Automatic tracking of mouse social posture dynamics by 3d videography, deep learning and gpu-accelerated robust optimization. *bioRxiv*. Retrieved from https://www.biorxiv.org/content/early/2020/05/25/2020.05.21.109629 doi: 10.1101/2020.05.21.109629

Ezray, B. D., Wham, D. C., Hill, C. E., & Hines, H. M. (2019). Unsupervised machine learning reveals mimicry complexes in bumblebees occur along a perceptual continuum. *Proceedings of the Royal Society B*, *286*(1910), 20191501.

Fang, Y., & Wang, J. (2012). Selection of the number of clusters via the bootstrap method. *Computational Statistics & Data Analysis*, *56*(3), 468–477.

Farine, D., Aplin, L., Garroway, C., Mann, R., & Sheldon, B. (2014). Collective decision making and social interaction rules in mixed-species flocks of songbirds. *Animal Behaviour*, *95*, 173–182.

Farine, D., Spencer, K., & Boogert, N. (2015). Early-life stress triggers juvenile zebra finches to switch social learning strategies. *Current Biology*, *25*, 2184–2188.

Farine, D., & Strandburg-Peshkin, A. (2015). Estimating uncertainty and reliability of social network data using bayesian inference. *Royal Society Open Science*, *2*, 150367.

Ferreira, A. C., Silva, L. R., Renna, F., Brandl, H. B., Renoult, J. P., Farine, D. R., ... Doutrelant, C. (2019). Deep learning-based methods for individual recognition in small birds. *bioRxiv*, 862557.

Figurnov, M., Mohamed, S., & Mnih, A. (2018). Implicit reparameterization gradients. In *Advances in neural information processing systems* (pp. 441–452).

Firth, J., Sheldon, B., & Farine, D. (2016). Pathways of information transmission among wild songbirds follow experimentally imposed changes in social foraging structure. *Biology Letters*, *12*, 20160144.

Firth, J., Voelkl, B., Farine, D., & Sheldon, B. (2015). Experimental evidence that social relationships determine individual foraging behavior. *Current Biology*, *25*, 3138–3143.

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

Fogel, S., Averbuch-Elor, H., Cohen-Or, D., & Goldberger, J. (2019). Clustering-driven deep embedding with pairwise constraints. *IEEE computer graphics and applications*, *39*(4), 16–27.

Fontaine, E., Lentink, D., Kranenbarg, S., Müller, U. K., van Leeuwen, J. L., Barr, A. H., & Burdick, J. W. (2008). Automated visual tracking for studying the ontogeny of zebrafish swimming. *Journal of Experimental Biology*, *211*(8), 1305–1316.

Francisco, F. A., Nührenberg, P., & Jordan, A. (2020). High-resolution, non-invasive animal tracking and reconstruction of local environment in aquatic ecosystems. *Movement ecology*, *8*(1), 1–12.

Francisco, F. A., Nührenberg, P., & Jordan, A. L. (2019). A low-cost, open-source framework for tracking and behavioural analysis of animals in aquatic ecosystems. *bioRxiv*. Retrieved from `https://www.biorxiv.org/content/early/2019/03/09/571232` doi: 10.1101/571232

Ghosh, P., Sajjadi, M. S., Vergari, A., Black, M., & Schölkopf, B. (2019). From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*.

Gomez-Marin, A., Paton, J. J., Kampff, A. R., Costa, R. M., & Mainen, Z. F. (2014). Big behavioral data: psychology, ethology and the foundations of neuroscience. *Nature neuroscience*, *17*(11), 1455.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Graving, J. M. (2017, October). *pinpoint: behavioral tracking using 2D barcode tags v0.0.1-alpha.* Retrieved from `https://doi.org/10.5281/zenodo.1008970` doi: 10.5281/zenodo.1008970

Graving, J. M. (2019, August). *behavelet: a wavelet transform for mapping behavior.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.3376742` doi: 10.5281/zenodo.3376742

Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., & Couzin, I. D. (2019a). Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife*, *8*, e47994.

Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., & Couzin, I. D. (2019b, August). *Example datasets for deepposekit.* Retrieved from `https://doi.org/10.5281/zenodo.3366908` doi: 10.5281/zenodo.3366908

Graving, J. M., & Couzin, I. D. (2020). Vae-sne: a deep generative model for simultaneous dimensionality reduction and clustering. *BioRxiv*.

Griffith, S., Holleley, C., Mariette, M., Pryke, S., & Svedin, N. (2010). Low level of extrapair parentage in wild zebra finches. *Animal Behaviour*, *79*, 261–264.

Grün, D., Lyubimova, A., Kester, L., Wiebrands, K., Basak, O., Sasaki, N., ... van Oudenaarden, A. (2015). Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, *525*(7568), 251.

Guizar-Sicairos, M., Thurman, S. T., & Fienup, J. R. (2008). Efficient subpixel image registration algorithms. *Optics letters*, *33*(2), 156–158.

Günel, S., Rhodin, H., Morales, D., Campagnolo, J., Ramdya, P., & Fua, P. (2019). Deepfly3d: A deep learning-based approach for 3d limb and appendage tracking in tethered, adult drosophila. *bioRxiv*, 640375.

Guo, X., Gao, L., Liu, X., & Yin, J. (2017). Improved deep embedded clustering with local structure preservation. In *Ijcai* (pp. 1753–1759).

Hafner, D., Tran, D., Lillicrap, T., Irpan, A., & Davidson, J. (2018). Reliable uncertainty estimates in deep neural networks using noise contrastive priors.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., ... Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, *2*(5), 6.

Hinton, G. E., & Roweis, S. T. (2003). Stochastic neighbor embedding. In *Advances in neural information processing systems* (pp. 857–864).

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

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

Hu, D., DeTone, D., & Malisiewicz, T. (2019). Deep charuco: Dark charuco marker pose estimation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 8436–8444).

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4700–4708).

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... others (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7310–7311).

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Ihle, M., Kempenaers, B., & Forstmeier, W. (2015). Fitness benefits of mate choice for compatibility in a socially monogamous species. *PLoS Biology*, *13*, 1002248.

Im, D. J., Verma, N., & Branson, K. (2018). Stochastic neighbor embedding under f-divergences. *arXiv preprint arXiv:1811.01247*.

Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. (2016). Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European conference on computer vision* (pp. 34–50).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).

Iqbal, U., Milan, A., & Gall, J. (2017). Posetrack: Joint multi-person pose estimation and tracking. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2011–2020).

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264–323.

Jang, E., Gu, S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jaques, M., Burke, M., & Hospedales, T. (2019). Physics-as-inverse-graphics: Joint unsupervised learning of objects and physics from video. *arXiv preprint arXiv:1905.11169*.

Javer, A., Currie, M., Lee, C. W., Hokanson, J., Li, K., Martineau, C. N., ... others (2018). An open-source platform for analyzing and sharing worm-behavior data. *Nature methods*, *15*(9), 645.

Jégou, S., Drozdzal, M., Vázquez, D., Romero, A., & Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CoRR*, *abs/1611.09326*. Retrieved from `http://arxiv.org/abs/1611.09326`

Jiang, Z., Zheng, Y., Tan, H., Tang, B., & Zhou, H. (2016). Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*.

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (pp. 694–711).

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., & Datta, S. R. (2016). Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems* (pp. 2946–2954).

Jolles, J., Fleetwood-Wilson, A., Nakayama, S., Stumpe, M., Johnstone, R., & Manica, A. (2015). The role of social attraction and its link with boldness in the collective movements of three-spined sticklebacks. *Animal Behaviour*, *99*, 147–153.

Jolles, J. W., Boogert, N. J., Sridhar, V. H., Couzin, I. D., & Manica, A. (2017). Consistent individual differences drive collective behavior and group functioning of schooling fish. *Current Biology*, *27*(18), 2862–2868.

Jun, J. J., Steinmetz, N. A., Siegle, J. H., Denman, D. J., Bauza, M., Barbarits, B., ... others (2017). Fully integrated silicon probes for high-density recording of neural activity. *Nature*, *551*(7679), 232.

Jung, A. (2018). *imgaug.* `https://github.com/aleju/imgaug`. GitHub.

Kabra, M., Robie, A., Rivera-Alba, M., Branson, S., & Branson, K. (2013). Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature Methods*, *10*, 64– 87.

Kain, J., Stokes, C., Gaudry, Q., Song, X., Foley, J., Wilson, R., & De Bivort, B. (2013). Leg-tracking and automated behavioural classification in drosophila. *Nature communications*, *4*, 1910.

Kain, J. S., Stokes, C., & de Bivort, B. L. (2012). Phototactic personality in fruit flies and its suppression by serotonin and white. *Proceedings of the National Academy of Sciences*, *109*(48), 19834–19839.

Kanazawa, A., Zhang, J. Y., Felsen, P., & Malik, J. (2019). Learning 3d human dynamics from video. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5614–5623).

Kane, G. A., Lopes, G., Saunders, J. L., Mathis, A., & Mathis, M. W. (2020). Real-time, low-latency closed-loop feedback using markerless posture tracking. *bioRxiv*.

Kang, B., García, D. G., Lijffijt, J., Santos-Rodríguez, R., & De Bie, T. (2019). Conditional t-sne: Complementary t-sne embeddings through factoring out prior information. *arXiv preprint arXiv:1905.10086*.

Karashchuk, P., Rupp, K. L., Dickinson, E. S., Sanders, E., Azim, E., Brunton, B. W., & Tuthill, J. C. (2020). Anipose: a toolkit for robust markerless 3d pose estimation. *bioRxiv*.

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

Ke, L., Chang, M.-C., Qi, H., & Lyu, S. (2018, September). Multi-scale structure-aware network for human pose estimation. In *The european conference on computer vision (eccv).*

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

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

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems* (pp. 3581–3589).

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems* (pp. 4743–4751).

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in neural information processing systems* (pp. 971–980).

Kleinberg, J. M. (2003). An impossibility theorem for clustering. In *Advances in neural information processing systems* (pp. 463–470).

Klibaite, U., Berman, G. J., Cande, J., Stern, D. L., & Shaevitz, J. W. (2017). An unsupervised method for quantifying the behavior of paired animals. *Physical biology*, *14*(1), 015006.

Klibaite, U., & Shaevitz, J. W. (2019). Interacting fruit flies synchronize behavior. *bioRxiv*, 545483.

Kobak, D., & Berens, P. (2019). The art of using t-sne for single-cell transcriptomics. *Nature communications*, *10*(1), 1–14.

Kobak, D., & Linderman, G. C. (2019). Umap does not preserve global structure any better than t-sne when using the same initialization. *bioRxiv*.

Krakauer, J. W., Ghazanfar, A. A., Gomez-Marin, A., MacIver, M. A., & Poeppel, D. (2017). Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, *93*(3), 480–490.

Kriengwatana, B., Spierings, M., & Cate, C. (2016). Auditory discrimination learning in zebra finches: effects of sex, early life conditions and stimulus characteristics. *Animal Behaviour*, *116*, 99–112.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* (Vol. 26). Springer.

Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in neural information processing systems* (pp. 2539–2547).

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., & Kingma, D. (2019). Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*.

König, B., Lindholm, A., Lopes, P., Dobay, A., Steinert, S., & Buschmann, F.-U. (2015). A system for automatic recording of social behavior in a free-living wild house mouse population. *Animal Biotelemetry*, *3*, 39.

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the llvm compiler infrastructure in hpc* (pp. 1–6).

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., ... others (2018). Rna velocity of single cells. *Nature*, *560*(7719), 494–498.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning* (pp. 1558–1566).

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

LeCun, Y., Cortes, C., & Burges, C. (2010). Mnist handwritten digit database.

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. In *Advances in neural information processing systems* (pp. 6391–6401).

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2017). Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*.

Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S., & Kluger, Y. (2019). Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, *16*(3), 243–245.

Liu, X., Yu, S.-y., Flierman, N., Loyola, S., Kamermans, M., Hoogland, T. M., & De Zeeuw, C. I. (2020). Optiflex: video-based animal pose estimation using deep learning enhanced by optical flow. *BioRxiv*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Luxem, K., Fuhrmann, F., Kürsch, J., Remy, S., & Bauer, P. (2020). Identifying behavioral structure from deep variational embeddings of animal motion. *bioRxiv*.

Maddison, C. J., Mnih, A., & Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Maldonado-Chaparro, A. A., Alarcón-Nieto, G., Klarevas-Irby, J. A., & Farine, D. R. (2018). Experimental disturbances reveal group-level costs of social instability. *Proceedings of the Royal Society B*, *285*(1891), 20181577.

Mariette, M., & Griffith, S. (2012). Nest visit synchrony is high and correlates with reproductive success in the wild zebra finch taeniopygia guttata. *Journal of Avian Biology*, *43*, 131–140.

Mariette, M., Pariser, E., Gilby, A., Magrath, M., Pryke, S., & Griffith, S. (2011). Using an electronic monitoring system to link offspring provisioning and foraging behavior of a wild passerine. *Auk*, *128*, 26–35.

Markowitz, J. E., Gillis, W. F., Beron, C. C., Neufeld, S. Q., Robertson, K., Bhagat, N. D., ... others (2018). The striatum organizes 3d behavior via moment-to-moment action selection. *Cell*, *174*(1), 44–58.

Mathis, A., Biasi, T., Mert, Y., Rogers, B., Bethge, M., & Mathis, M. W. (2020). Imagenet performance correlates with pose estimation robustness and generalization on out-of-domain data. In *International conference on machine learning.*

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*. Retrieved from `https://www.nature.com/articles/s41593-018-0209-y`

Mathis, A., Schneider, S., Lauer, J., & Mathis, M. W. (2020). A primer on motion capture with deep learning: Principles, pitfalls and perspectives. *arXiv preprint arXiv:2009.00564*.

Mathis, A., & Warren, R. A. (2018). On the inference speed and video-compression robustness of deeplabcut. *bioRxiv*. Retrieved from `https://www.biorxiv.org/content/early/2018/10/30/457242` doi: 10.1101/457242

Mathis, M. W., & Mathis, A. (2020). Deep learning tools for the measurement of animal behavior in neuroscience. *Current opinion in neurobiology*, *60*, 1–11.

McDaid, A. F., Greene, D., & Hurley, N. (2011). Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*.

McDonald, K. (2016, February 18). *Parametric-t-sne: Running parametric t-sne by laurens van der maaten with octave and oct2py.* `https://github.com/kylemcdonald/Parametric-t-SNE`.

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, *2*(11), 205.

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mendes, C. S., Bartos, I., Akay, T., Márka, S., & Mann, R. S. (2013). Quantification of gait parameters in freely walking wild type and sensory deprived drosophila melanogaster. *elife*, *2*, e00231.

Mersch, D., Crespi, A., & Kelle, L. (2013). Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, *340*, 1090–1093.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, *50*(2), 159–179.

Moody, C. E. (2017, March 1). *topicsne: t-sne experiments in pytorch.* `https://github.com/cemoody/topicsne`; archived at: `https://github.com/jgraving/topicsne`.

Morand-Ferron, J., Hamblin, S., Cole, E., Aplin, L., & Quinn, J. (2015). Taking the operant paradigm into the field: Associative learning in wild great tits. *Plos One*, *10*.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, *5*(1), 32–38.

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

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

Nagy, M., Vasarhelyi, G., Pettit, B., Roberts-Mariani, I., Vicsek, T., & Biro, D. (2013). Context-dependent hierarchies in pigeons. In *Proceedings of the national academy of sciences of the united states of america* (Vol. 110, p. 13049–13054).

Narayan, A., Berger, B., & Cho, H. (2020). Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability. *bioRxiv*.

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019a). Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*.

Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. (2019b). Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, *14*(7), 2152–2176.

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *Computer vision - ECCV 2016 - 14th european conference, amsterdam, the netherlands, october 11-14, 2016, proceedings, part VIII* (pp. 483–499). doi: 10.1007/978-3-319-46484-8_29

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., ... others (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, *15*(10), 805–815.

Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in neural information processing systems* (pp. 2338–2347).

Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review. *Acm Sigkdd Explorations Newsletter*, *6*(1), 90–105.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8024–8035).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S.-H., Murthy, M., & Shaevitz, J. W. (2019). Fast animal pose estimation using deep neural networks. *Nature methods*, *16*(1), 117.

Pereira, T. D., Tabris, N., Li, J., Ravindranath, S., Papadoyannis, E. S., Wang, Z. Y., ... others (2020). Sleap: Multi-animal pose tracking. *bioRxiv*.

Pérez-Escudero, A., Vicente-Page, J., Hinz, R. C., Arganda, S., & De Polavieja, G. G. (2014). idtracker: tracking individuals in a group by automatic identification of unmarked animals. *Nature methods*, *11*(7), 743.

Pezzotti, N., Höllt, T., Lelieveldt, B., Eisemann, E., & Vilanova, A. (2016). Hierarchical stochastic neighbor embedding. In *Computer graphics forum* (Vol. 35, pp. 21–30).

Pham, D. T., Dimov, S. S., & Nguyen, C. D. (2005). Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *219*(1), 103–119.

Poličar, P. G., Stražar, M., & Zupan, B. (2019). Embedding to reference t-sne space addresses batch effects in single-cell classification. In *International conference on discovery science* (pp. 246–260).

Poličar, P. G., Strazar, M., & Zupan, B. (2019). opentsne: a modular python library for t-sne dimensionality reduction and embedding. *BioRxiv*, 731877.

Poole, B., Ozair, S., Oord, A. v. d., Alemi, A. A., & Tucker, G. (2019). On variational bounds of mutual information. *arXiv preprint arXiv:1905.06922*.

Pratt, L. Y. (1993). Discriminability-based transfer between neural networks. In *Advances in neural information processing systems* (pp. 204–211).

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

Price, E., Lawless, G., Ludwig, R., Martinovic, I., Bülthoff, H. H., Black, M. J., & Ahmad, A. (2018). Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles. *IEEE Robotics and Automation Letters*, *3*(4), 3193–3200.

Psorakis, I., Voelkl, B., Garroway, C., Radersma, R., Aplin, L., Crates, R., ... Sheldon, B. (2015). Inferring social structure from temporal data. *Behavioral Ecology and Sociobiology*, *69*, 857–866.

Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., & Teh, Y. W. (2018). Tighter variational bounds are not necessarily better. *arXiv preprint arXiv:1802.04537*.

Ran, F. A., Hsu, P. D., Wright, J., Agarwala, V., Scott, D. A., & Zhang, F. (2013). Genome engineering using the crispr-cas9 system. *Nature protocols*, *8*(11), 2281.

Razavi, A., van den Oord, A., & Vinyals, O. (2019). Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems* (pp. 14866–14876).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Rezaabad, A. L., & Vishwanath, S. (2019). Learning representations by maximizing mutual information in variational autoencoder. *arXiv preprint arXiv:1912.13361*.

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*.

Robie, A., Seagraves, K., Egnor, S., & Branson, K. (2017). Machine vision methods for analyzing social interactions. *Journal of Experimental Biology*, *220*, 25–34.

Robinson, I., & Pierce-Hoffman, E. (2020). Tree-sne: Hierarchical clustering and visualization using t-sne. *arXiv preprint arXiv:2002.05687*.

Rojas Mora, A., Forstmeier, W., & Fusani, L. (2014). The importance of validating experimental setups: Lessons from studies of food choice copying in zebra finches. *Ethology*, *120*, 913–922.

Rolinek, M., Zietlow, D., & Martius, G. (2019). Variational autoencoders pursue pca directions (by accident). In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 12406–12415).

Romero-Ferrero, F., Bergomi, M. G., Hinz, R. C., Heras, F. J., & de Polavieja, G. G. (2019). idtracker. ai: tracking all individuals in small or large collectives of unmarked animals. *Nature methods*, *16*(2), 179.

Romero-Ramire, F. J., Muñoz-Salinas, R., & Medina-Carnicer, R. (2019). Fractal markers: a new approach for long-range marker pose estimation under occlusion. *IEEE Access*, *7*, 169908–169919.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Rosenthal, S. B., Twomey, C. R., Hartnett, A. T., Wu, H. S., & Couzin, I. D. (2015). Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion. *Proceedings of the National Academy of Sciences*, *112*(15), 4690–4695.

Roy, A. G., Conjeti, S., Navab, N., & Wachinger, C. (2018). Bayesian quicknat: Model uncertainty in deep whole-brain segmentation for structure-wise quality control. *CoRR*, *abs/1811.09800*. Retrieved from `http://arxiv.org/abs/1811.09800`

Ruploh, T., Bischof, H., & Engelhardt, N. (2014). Social experience during adolescence influences how male zebra finches (taeniopygia guttata) group with conspecifics. *Behavioral Ecology and Sociobiology*, *68*, 537–549.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856–3866).

Sainburg, T., Thielk, M., & Gentner, T. Q. (2019). Latent space visualization, characterization, and generation of diverse vocal communication signals. *bioRxiv*, 870311.

Saini, N., Price, E., Tallamraju, R., Enficiaud, R., Ludwig, R., MartinoviÄ‡, I., . . . Black, M. (2019, October). Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles. In *International conference on computer vision.*

Samusik, N., Good, Z., Spitzer, M. H., Davis, K. L., & Nolan, G. P. (2016). Automated mapping of phenotype space with single-cell data. *Nature methods*, *13*(6), 493.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4510–4520).

Santema, P., Schlicht, E., Schlicht, L., & Kempenaers, B. (2017). Blue tits do not return faster to the nest in response to either short- or long-term begging playbacks. *Animal Behaviour*, *123*, 117–127.

Saravanan, V., Berman, G. J., & Sober, S. J. (2019). Application of the hierarchical bootstrap to multi-level data in neuroscience. *BioRxiv*, 819334.

Scheibe, K., Eichhorn, K., Wiesmayr, M., Schonert, B., & Krone, O. (2008). Long-term automatic video recording as a tool for analysing the time patterns of utilisation of predefined locations by wild animals. *European Journal of Wildlife Research*, *54*, 53–59.

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

Schlicht, L., Valcu, M., & Kempenaers, B. (2015). Male extraterritorial behavior predicts extrapair paternity pattern in blue tits,cyanistes caeruleus. *Behavioral Ecology*, *26*, 1404–1413.

Schuett, W., Dall, S., & Royle, N. (2011). Pairs of zebra finches with similar 'personalities' make better parents. *Animal Behaviour*, *81*, 609–618.

Seethapathi, N., Wang, S., Saluja, R., Blohm, G., & Kording, K. P. (2019). Movement science needs different pose tracking algorithms. *arXiv preprint arXiv:1907.10226*.

Sixt, L., Wild, B., & Landgraf, T. (2018). Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, *5*, 66.

Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems* (pp. 1857–1865).

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., & Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in neural information processing systems* (pp. 3308–3318).

Steinmeyer, C., Mueller, J., & Kempenaers, B. (2013). Individual variation in sleep behaviour in blue titscyanistes caeruleus: assortative mating and associations with fitness-related traits. *Journal of Avian Biology*, *44*, 159–168.

Stephens, G. J., de Mesquita, M. B., Ryu, W. S., & Bialek, W. (2011). Emergence of long timescales and stereotyped behaviors in caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, *108*(18), 7286–7289.

Stephens, G. J., Johnson-Kerner, B., Bialek, W., & Ryu, W. S. (2008). Dimensionality and dynamics in the behavior of c. elegans. *PLoS Comput Biol*, *4*(4), e1000028.

Still, S., & Bialek, W. (2004). How many clusters? an information-theoretic perspective. *Neural computation*, *16*(12), 2483–2506.

Stoddard, M. C., Yong, E. H., Akkaynak, D., Sheard, C., Tobias, J. A., & Mahadevan, L. (2017). Avian egg shape: Form, function, and evolution. *Science*, *356*(6344), 1249–1254.

Stowers, J. R., Hofbauer, M., Bastien, R., Griessner, J., Higgins, P., Farooqui, S., ... others (2017). Virtual reality for freely moving animals. *Nature methods*, *14*(10), 995.

Strandburg-Peshkin, A., Farine, D. R., Couzin, I. D., & Crofoot, M. C. (2015). Shared decision-making drives collective movement in wild baboons. *Science*, *348*(6241), 1358–1361.

Strandburg-Peshkin, A., Farine, D. R., Crofoot, M. C., & Couzin, I. D. (2017). Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. *Elife*, *6*, e19505.

Strandburg-Peshkin, A., Twomey, C. R., Bode, N. W., Kao, A. B., Katz, Y., Ioannou, C. C., ... others (2013). Visual sensory networks and effective information transfer in animal groups. *Current Biology*, *23*(17), R709–R711.

Stringer, C., Pachitariu, M., Steinmetz, N., Carandini, M., & Harris, K. D. (2019). High-dimensional geometry of population responses in visual cortex. *Nature*, *571*(7765), 361–365.

Stringer, C., Pachitariu, M., Steinmetz, N., Reddy, C. B., Carandini, M., & Harris, K. D. (2019). Spontaneous behaviors drive multidimensional, brain-wide activity. *cell*, *1*(10), 100.

Sussillo, D., Jozefowicz, R., Abbott, L., & Pandarinath, C. (2016). Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*.

Szubert, B., Cole, J. E., Monaco, C., & Drozdov, I. (2019). Structure-preserving visualisation of high dimensional single-cell datasets. *Scientific reports*, *9*(1), 1–10.

Tirosh, I., Izar, B., Prakadan, S. M., Wadsworth, M. H., Treacy, D., Trombetta, J. J., ... others (2016). Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq. *Science*, *352*(6282), 189–196.

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Todd, J. G., Kain, J. S., & de Bivort, B. L. (2017). Systematic exploration of unsupervised methods for mapping behavior. *Physical biology*, *14*(1), 015002.

Togasaki, D., Hsu, A., Samant, M., Farzan, B., DeLanney, L., Langston, J., ... Quik, M. (2005). The webcam system: a simple, automated, computer-based video system for quantitative measurement of movement in nonhuman primates. *Journal of Neuroscience Methods*, *145*, 159–166.

Tomczak, J. M., & Welling, M. (2017). Vae with a vampprior. *arXiv preprint arXiv:1705.07120*.

Tome, D., Russell, C., & Agapito, L. (2017). Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2500–2509).

Tran, D., Hoffman, M. W., Moore, D., Suter, C., Vasudevan, S., & Radul, A. (2018). Simple, distributed, and accelerated probabilistic programming. In *Advances in neural information processing systems* (pp. 7609–7620).

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

Valentin, J., Keskin, C., Pidlypenskyi, P., Makadia, A., Sud, A., & Bouaziz, S. (2019). Tensorflow graphics: Computer graphics meets deep learning..

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. (2016). Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems* (pp. 4790–4798).

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in neural information processing systems* (pp. 6306–6315).

van der Maaten, L. (2009). Learning a parametric embedding by preserving local structure. In *Artificial intelligence and statistics* (pp. 384–391).

van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, *15*(1), 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

Versace, E., Caffini, M., Werkhoven, Z., & de Bivort, B. L. (2019). Individual, but not population asymmetries, are modulated by social environment and genotype in drosophila melanogaster. *bioRxiv*, 694901.

Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, *11*, 2837–2854.

Wang, M., & Wang, D. (2016). Vmf-sne: Embedding for spherical data. In *2016 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 2344–2348).

Waskom, M., Botvinnik, O., Ostblom, J., Gelbart, M., Lukauskas, S., Hobson, P., ... Brian (2020, April). *mwaskom/seaborn: v0.10.1 (april 2020).* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.3767070` doi: 10.5281/zenodo.3767070

Weigert, M., Schmidt, U., Boothe, T., Müller, A., Dibrov, A., Jain, A., ... others (2018). Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, *15*(12), 1090.

Weissbrod, A., Shapiro, A., Vasserman, G., Edry, L., Dayan, M., Yitzhaky, A., ... Kimchi, T. (2013). Automated long-term tracking and social behavioural phenotyping of animal colonies within a semi-natural environment. *Nature Communications*, *4*.

Werkhoven, Z., Bravin, A., Skutt-Kakaria, K., Reimers, P., Pallares, L. F., Ayroles, J., & De Bivort, B. L. (2019). The structure of behavioral variation within a genotype. *BioRxiv*, 779363.

Werkhoven, Z., Rohrsen, C., Qin, C., Brembs, B., & de Bivort, B. (2019). Margo (massively automated real-time gui for object-tracking), a platform for high-throughput ethology. *BioRxiv*, 593046.

Wham, D. C., Ezray, B. D., & Hines, H. M. (2019). Measuring perceptual distance of organismal color pattern using the features of deep neural networks. *bioRxiv*, 736306.

Whitehead, H. (2008). *Analysing animal societies: quantitative methods for vertebrate social analysis*. Chicago, USA: Chicago Univ. Press.

Wild, B., Sixt, L., & Landgraf, T. (2018). Automatic localization and decoding of honeybee markers using deep convolutional neural networks. *CoRR*, *abs/1802.04557*. Retrieved from `http://arxiv.org/abs/1802.04557`

Wiltschko, A. B., Johnson, M. J., Iurilli, G., Peterson, R. E., Katon, J. M., Pashkovski, S. L., ... Datta, S. R. (2015). Mapping sub-second structure in mouse behavior. *Neuron*, *88*(6), 1121–1135.

Wu, A., Buchanan, E. K., Whiteway, M., Schartner, M., Meijer, G., Noel, J.-P., ... others (2020). Deep graph pose: a semi-supervised deep graphical model for improved animal pose tracking. *bioRxiv*.

Wuerz, Y., & Kruger, O. (2015). Personality over ontogeny in zebra finches: long-term repeatable traits but unstable behavioural syndromes. *Frontiers in Zoology*, *12*, 9.

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478–487).

Xiu, Y., Li, J., Wang, H., Fang, Y., & Lu, C. (2018). Pose flow: Efficient online pose tracking. *arXiv preprint arXiv:1802.00977*.

Yang, L., Cheung, N.-M., Li, J., & Fang, J. (2019). Deep clustering by gaussian mixture variational autoencoders with graph embedding. In *Proceedings of the ieee international conference on computer vision* (pp. 6440–6449).

Zann, R. (1994). Reproduction in a zebra finch colony in south-eastern australia: the significance of monogamy, precocial breeding and multiple broods in a highly mobile species. *Emu*, *94*, 285–299.

Zhang, Q., Zhou, J., He, J., Cun, X., Zeng, S., & Zhang, B. (2019). A shell dataset, for shell features extraction and recognition. *Scientific data*, *6*(1), 1–9.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 586–595).

Zhao, S., Song, J., & Ermon, S. (2017). Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

Zuffi, S., Kanazawa, A., Berger-Wolf, T., & Black, M. J. (2019, October). Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *International conference on computer vision.*

Zuffi, S., Kanazawa, A., Jacobs, D. W., & Black, M. J. (2017). 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 6365–6373).

Acknowledgements

Chapter 1

We thank Jana Hörsch, Laila Darouich, Alex Bruttel, Daniel Zuñiga and the animal caretaker team at the Max Planck Institute for Ornithology in Radolfzell for assistance with data collection and monitoring animal health each day. We thank Iain Couzin, Andrew Straw and two anonymous reviewers for helpful comments on the manuscript. We also thank Iain Couzin for his support of the project. This work was funded by the Max Planck Society. D.R.F. and A.A.M.-C. received additional funding from the German Research Foundation (DFG grant FA 1420/4-1 awarded to D.R.F.).

Chapter 2

We are indebted to Talmo Pereira et al. and A. Mathis et al. for making their software open-source and freely-available—this project would not have been possible without them. We also thank M. Mathis and A. Mathis for their comments, which greatly improved the manuscript. We thank François Chollet, the Keras and TensorFlow teams, and Alexander Jung for their open source contributions, which provided the core programming interface for our work. We thank A. Strandburg-Peshkin, Vivek H. Sridhar, Michael L. Smith, and Joseph B. Bak-Coleman for their helpful discussions on the project and comments on the manuscript. 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, Markus Miller 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.

Chapter 3

We thank Gordon Berman and members of the Berman lab, Tim Landgraf, Ben Wild, David Dormagen, Conor Heins, Blair Costelloe, Ian Etheredge, and Ari Strandburg-Peshkin for their helpful comments on the manuscript. We are also grateful to Mike Costelloe for his creative advice on the figures, Einat Couzin-Fuchs and Dan Bath for their expert input, and Michael Smith for the use of his GPU. J.M.G. and I.D.C. acknowledge support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2117 - 422037984. I.D.C. acknowledges support from NSF Grant IOS-1355061, Office of Naval Research Grant (ONR, N00014-19-1-2556), the Struktur-und Innovationsfonds für die Forschung of the State of Baden-Württemberg, and the Max Planck Society.

Author Contributions

Chapter 1

D.R.F. conceived the project; G.A.-N. and J.A.K.-I. developed the backpack design and deployment procedure; IM monitored the health of the birds and impacts of the backpacks; G.A.-N. tested image and video capture methods, and collected the data; J.M.G. developed the pinpoint software and assisted with the design of the image and video capture methods; A.A.M.-C. and J.A.K.-I. performed the example data analyses; G.A.-N. and D.R.F. wrote the first draft of the manuscript. All authors contributed to editing and revising the final manuscript.

Chapter 2

- **J.M.G.** — Conceptualization, Data curation, Software, Formal analysis, Validation, Investigation, Visualization, Methodology, Writing–original draft, Project administration, Writing–review and editing
- **D.C.** — Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Data curation, Writing-review and editing
- **H.N.** — Conceptualization, Methodology, Writing-review and editing
- **L.L.** — Validation, Investigation, Resources, Data curation, Writing-reviewing and editing
- **B.K.** — Validation, Investigation, Resources, Data curation, Writing-reviewing and editing
- **B.R.C.** — Validation, Investigation, Resources, Data curation, Writing-reviewing and editing, Project administration, Funding acquisition
- **I.D.C.** — Conceptualization, Resources, Writing-reviewing and editing, Supervision, Project administration, Funding acquisition

Chapter 3

- **J.M.G.** — Conceptualization, Data curation, Software, Formal analysis, Validation, Investigation, Visualization, Methodology, Writing–original draft, Project administration, Writing–review and editing
- **I.D.C.** — Conceptualization, Resources, Writing–reviewing and editing, Supervision, Project administration, Funding acquisition