scientific reports

Check for updates

OPEN

DeepAction: a MATLAB toolbox for automated classification of animal behavior in video

Carl Harris[1], Kelly R. Finn[1,2], Marie-Luise Kieseler[1], Marvin R. Maechler[1] & Peter U. Tse[1]✉

The identification of animal behavior in video is a critical but time-consuming task in many areas of research. Here, we introduce DeepAction, a deep learning-based toolbox for automatically annotating animal behavior in video. Our approach uses features extracted from raw video frames by a pretrained convolutional neural network to train a recurrent neural network classifier. We evaluate the classifier on two benchmark rodent datasets and one octopus dataset. We show that it achieves high accuracy, requires little training data, and surpasses both human agreement and most comparable existing methods. We also create a confidence score for classifier output, and show that our method provides an accurate estimate of classifier performance and reduces the time required by human annotators to review and correct automatically-produced annotations. We release our system and accompanying annotation interface as an open-source MATLAB toolbox.

The classification and analysis of animal behavior in video is a ubiquitous but often laborious process in life sciences research. Traditionally, such analyses have been performed manually. This approach, however, suffers from several limitations. Most obvious is that it requires researchers to allocate much of their time to the tedious work of behavioral annotation, limiting or slowing the progress of downstream analyses. Particularly for labs without research assistants or paid annotators, the opportunity cost of annotating video can be quite high. Manual annotation also suffers from relatively poor reproducibility and reliability[1–3], largely due to the limited attentional capacity of human annotators. This issue is particularly salient in studies involving rodents. Due to their nocturnal nature, rodents are preferably studied under dimmed or infrared light[4], which makes the identification of behaviors more difficult due to more limited light and color cues. This, in turn, increases annotators' fatigue and reduces their capacity to pay attention for extended periods, introducing variation in annotation quality, thereby decreasing the quality of behavioral data[5].

Given the time and accuracy limitations of manual annotation, increasing work has focused on creating methods to automate the annotation process. Many such methods rely on tracking animals' bodies[4,6–9] or body parts[10], from which higher-level features (e.g., velocity, acceleration, and posture) are extrapolated and used to classify behavior. Jhuang et al.[7], for example, used motion and trajectory features to train a hidden Markov support vector machine to categorize eight classes of mouse behavior. Burgos-Artizzu et al.[6] used spatiotemporal and trajectory features and a temporal context model to classify the social behavior of mice using two camera views. However, approaches using these "hand-crafted features" are limited in several ways[11]. First, they require that researchers identify sets of features that both encompass a given animal's entire behavioral repertoire and can distinguish between visually similar behaviors. For example, "eating" and "grooming snout" behaviors in rodents do not have a well-defined difference in posture or movement[4], making crafting features to differentiate them difficult. Second, after features have been selected, detecting and tracking them is difficult and imperfect. Subtle changes in video illumination, animal movement, and environment can result in inaccurate keypoint detection, decreasing the fidelity of extracted features. And third, selected feature sets are often experiment-specific. Those optimal for a singly housed rodent study, for example, likely differ from those optimal for a social rodent study. This increases the complexity of the feature-selection task, impeding experimental progress and annotation accuracy.

To address these limitations, Bohnslav et al.[11] proposed an alternative to hand-crafted approaches, instead using hidden two-stream networks[12] and temporal gaussian mixture networks[13], and achieved high classification accuracy on a diverse collection of animal behavior datasets. Here, we expand on this work by introducing DeepAction, a MATLAB toolbox for the automated annotation of animal behavior in video. Our approach utilizes a two-stream[14] convolutional and recurrent neural network architecture[15,16] to generate behavioral labels

[1]Department of Psychological and Brain Science, Dartmouth College, Hanover, NH 03755, USA. [2]Neukom Institute, Dartmouth College, Hanover, NH 03755, USA. ✉email: Peter.U.Tse@dartmouth.edu

from raw video frames. We use convolutional neural networks (CNNs) and dense optical flow to extract spatial and temporal features from video[17], which are then used to train a long short-term memory network classifier to predict behavior. We evaluate our approach on two benchmark datasets of laboratory mouse video and one dataset of octopus video. We show that it outperforms existing methods and reaches human-level performance with little training data. In addition to outputting behavior labels for each video frame, we also introduce a classification confidence system that generates a measure of how "confident" the classifier is about each label. This allows researchers to estimate the quality of automatically-produced annotations without having to review them, and reduces the time required to review annotations by allowing users to selectively correct ambiguous ones, while omitting those that the classifier produced with high confidence. We show that this confidence score accurately differentiates low quality annotations from high quality ones and improves the efficiency of reviewing and correcting video. Finally, we release the code and annotation GUI as an open-source MATLAB project.

Results

The DeepAction workflow. The toolbox workflow (Fig. 1A) begins with the importation of unlabeled video into a new DeepAction project and ends with the export of annotations for all the videos in that project. The workflow consists of two parts: a classification component (steps 2–8) and a review component (steps 9 and 10). In the classification portion, we adopt a supervised learning approach in which a portion of project videos are labeled and used to train a classifier. This classifier learns to associate the content in the video frames with a set of user-defined behavior labels (e.g., "walk" or "drink"). After the classifier is trained, it can then be used to predict behaviors in the *unlabeled* video. In addition to predicting behaviors occurring in the unlabeled video, the classifier outputs a "confidence score," representing an estimate of the agreement between classifier-produced labels and human-produced ones. This confidence score is used during the review component of the workflow, in which low-confidence annotations can be preferentially reviewed and corrected, while those with high confidence are omitted. After this confidence-based review, annotations are exported for use in the researcher's given analysis.

To represent the video for input to the classifier, we opt for a "two-stream" model[18], where the first stream ("spatial stream") captures the spatial information of the video frames, and the second stream ("temporal stream") captures the motion between frames (Fig. 1E). We first extract video frames representing the spatial and temporal information ("spatial frames" and "temporal frames," respectively) in the underlying video (see "Methods: Frame Extraction" sections). To generate spatial frames, which contain information about the scenes and objects in the video, we extract the raw video frames from each video file. To generate temporal frames, which contain information about the movement of the camera and objects in the video, we use dense optical flow to calculate the movement of individual pixels between *pairs* of sequential frames. Dense optical flow generates a two-dimensional vector field for each pixel in the image, where each vector represents the estimated movement of a pixel from one image to the next[19]. We then express this entire vector field visually as an image, where a given pixel's color is governed by the orientation and magnitude of its corresponding flow vector.

We then generate a low-dimensional representation of the spatial and temporal frames by extracting their salient visual features[20] using the ResNet18 pretrained convolutional neural network (CNN; see "Methods: Feature extraction" sections). For each spatial and temporal frame, the feature extractors generate a 512-dimensional vector representing the high-level visual information contained in that frame. We then concatenate these spatial and temporal features (dimensionality of 1024) to create the initial spatiotemporal features, and then use reconstruction independent component analysis to reduce the dimensionality to 512, forming the final spatiotemporal features used to train the classifier.

Training the classifier requires a portion of video be manually labeled so it can learn the associations between the video's corresponding spatiotemporal features (input) and behavior labels (output). Rather than annotating whole videos at a time, we instead split each video into short "clips," where each clip is a short segment of the longer video, and then select a subset of these clips to annotate (Fig. 1B). This approach is preferable, as compared to annotating full videos, because it better captures the substantial variation in features and the feature-to-label relationship *across* videos, improving the generalizability of the classifier. That is, annotating short clips reduces dataset shift[21,22] between the training set (i.e., the annotated videos) and the unlabeled videos.

After the set of videos has been split into clips, a subset of these user-specified clips, $\text{prop}_{\text{validate}}$, is randomly selected for manual annotation (Fig. 1B) using a GUI included in the toolbox release (Fig. 6B). After annotation, labeled clip data (video, features, and annotations), $\mathcal{D}^{\text{labeled}}$, is used to train a recurrent neural network classifier (see "Methods: Classifier architecture" sections) and the confidence-based review system. To do so, we first further split $\mathcal{D}^{\text{labeled}}$ into a training set and a validation set (Fig. 1C). The training set, $\mathcal{D}^{\text{train}}$, comprises most of the labeled data and is directly used to train the classifier. For a given clip in $\mathcal{D}^{\text{train}}$ with n frames, a spatiotemporal feature array of size $[n, 512]$ is input into a recurrent neural network classifier, along with a series of n manually annotated behavior labels. The network then tries to predict the manual annotations using the features; training iteratively reduces the difference between classifier-predicted and human annotations. The spatiotemporal features for a given segment of video represent the visual content of that segment; so, by predicting labels using these features, the classifier is indirectly generating predictions for the underlying video data. The independent validation set, $\mathcal{D}^{\text{validate}}$, is used to tune the model training process and confidence-based review (see "Methods: Classifier training" sections). The trained classifier and confidence-based review system are then used to generate annotations and confidence scores for the remaining, unlabeled data, $\mathcal{D}^{\text{unlabeled}}$.

We then introduce a confidence-based review system. Recall that, after the classifier has been trained, it can be used to predict behaviors in unlabeled data, $\mathcal{D}^{\text{unlabeled}}$. In addition, we output a confidence score for each clip in $\mathcal{D}^{\text{unlabeled}}$ corresponding the estimated accuracy of the labels produced for that clip (see "Methods: Confidence score definition" sections). In an ideal metric, a clip's confidence score should correspond to the ground truth

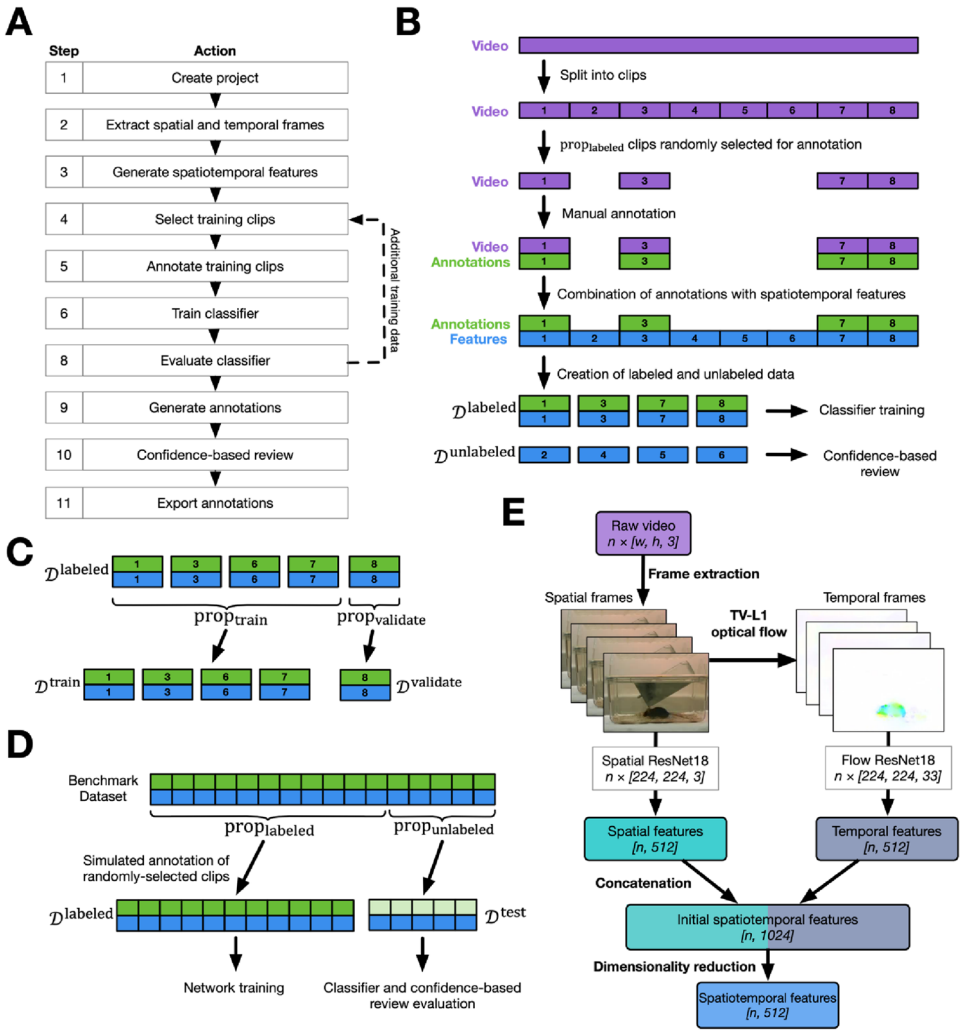

Figure 1. Toolbox workflow and data selection process. (**A**) Workflow for the DeepAction toolbox. Arrows indicate the flow of project actions, with the dashed arrow denoting that, following training of the classifier, additional training data can be annotated and used to re-train the classifier. (**B**) An overview of the clip selection process. Long videos are divided into clips of a user-specified length, from which a user-specified proportion ($\text{prop}_{\text{labeled}}$) are randomly selected for annotation ($\mathcal{D}^{\text{labeled}}$). The selected video clips are then annotated, and these annotations are used in combination with their corresponding features to train the classifier. The trained classifier is used to generate predictions and confidence scores for the non-selected clips ($\mathcal{D}^{\text{unlabeled}}$), which the user can then review and correct as necessary. (**C**) Labeled data are further divided into training ($\mathcal{D}^{\text{train}}$) and validation ($\mathcal{D}^{\text{validate}}$) data. (**D**) Process for simulating clip-selection using our benchmark datasets, where we simulate selecting $\text{prop}_{\text{labeled}}$ of the data for labeling ($\mathcal{D}^{\text{labeled}}$) and evaluate it on the unselected data ($\mathcal{D}^{\text{test}}$). (**E**) Process to generate spatiotemporal features from video frames. Raw video frames are extracted from the video file ("frame extraction"). The movement between frames is calculated using TV-L1 optical flow and then represented visually as the temporal frames. Spatial and temporal frames are input into their corresponding pretrained CNN ("spatial ResNet18" and "flow Resnet18," respectively), from which spatial and temporal features are extracted. The spatial and temporal features are then concatenated, and then their dimensionality is reduced to generate the final spatiotemporal features that are used to train the classifier. Dimensionality is shown in italicized brackets.

likelihood of the classifier-predicted behaviors being correct. The purpose of the confidence score is two-fold (see "Methods: Confidence-based review" sections). First, by generating estimated accuracies for each clip in $\mathcal{D}^{\text{unlabeled}}$, we can estimate the *overall* accuracy of $\mathcal{D}^{\text{unlabeled}}$. Just as there is variability in annotations between researchers, we can expect that even a well-performing classifier's annotations will not exactly match those that would be produced if the unlabeled data was manually annotated. But, by providing an estimate of the agreement between human- and classifier-produced labels in $\mathcal{D}^{\text{unlabeled}}$ automatically, users can easily decide whether the classifier's performance is sufficient for their given application. The second purpose is to enable researchers to preferentially review and correct clips where the classifier is less accurate over those where annotations are highly accurate. Rather than reviewing each clip in $\mathcal{D}^{\text{unlabeled}}$, researchers can review and correct only the subset of clips

where the classifier is uncertain about its predictions. If the confidence score is a precise estimate of accuracy, then the clips with a low confidence score will be the clips that the classifier performs poorly on, allowing for labels to be corrected more efficiently.

Datasets. In our primary analyses, we evaluate our approach on two publicly available "benchmark" datasets of mice in a laboratory setting (see "Methods: Datasets" sectin). Both datasets are fully annotated, allowing us to test and evaluate our model. The first dataset, referred to as the "home-cage dataset," was collected by Jhuang et al.[7], and features 12 videos (10.5 h total; Fig. 2D) of singly housed mice in their home cages performing eight stereotypical, mutually-exclusive behaviors recorded from the side of the cage (Fig. S1A). The second dataset, called "CRIM13,"[6] consists of 237 pairs of videos, recorded with synchronized side and top views, of pairs of mice engaging in social behavior, categorized into 13 distinct, mutually-exclusive actions (Fig. S1B). Each video is approximately 10 min in duration, for a total of approximately 88 h of video and annotations (Fig. 2D). In addition to these benchmark datasets, we challenge the classifier by evaluating it on an "exploratory" unpublished dataset of octopus bimaculoides behavior during a habituation task (see "Methods: Datasets" section).

For the benchmark datasets, because they are already fully labeled, we evaluate our method by simulating the labeling process (see "Methods: Simulating labeled data" section). We assume a user has chosen to annotate some proportion of the data and uses that data to train a classifier and obtain predictions for the remaining data. In practice, the user would run the classifier over the remaining data to automatically generate labels and use the confidence-based review system to review those labels as desired. Here, however, since the data have been annotated, we know the true labels for the "unlabeled" data. This allows us to test the performance of the method *as if it were being used* to produce labels for the remaining $\text{prop}_{\text{unlabeled}}$ data. This approach allows us to simulate performance across a range of labeling proportions, which in turn provides a measure of how a model can be expected to perform for a given amount of manual annotation time. So, for a given $\text{prop}_{\text{labeled}}$, we train the classifier and confidence-based review using the "labeled data," and then test how the approach performs on the remaining data (Fig. 1D).

High classification accuracy with little training data. We first evaluate the performance of the classifier (i.e., accuracy and F1; see "Methods: Classifier evaluation" section) with varying amounts of training data (Fig. 2A,E), and show that it requires remarkably little manual annotation to achieve high accuracy. For a given proportion labeled (i.e., $\text{prop}_{\text{unlabeled}}$ above), a corresponding proportion of project clips are randomly selected from all the clips in the dataset and used to train the classifier, which is then evaluated on the remaining data

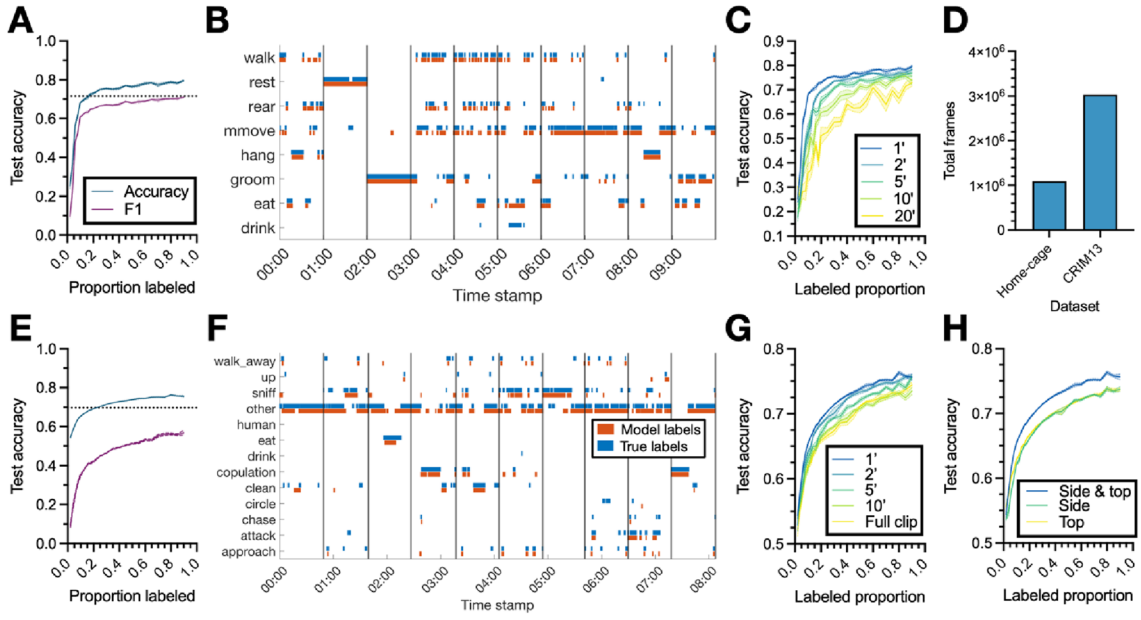

Figure 2. Classifier performance. (**A**) Test set accuracy and overall F1 score of the classifier on the home-cage dataset as a function of the proportion of the dataset used to train it. The proportion of data denoted on the *x*-axis is used to train the classifier, which is then evaluated on the remainder of the dataset. (**B**) Sample ethogram of classifier labels and ground-truth annotations from 10 randomly selected home-cage clips. Each colored line indicates the label of that behavior at the corresponding time stamp. Vertical black lines denote the divisions of the video into clips (one minute in duration). (**C**) Test set accuracy on the home-cage dataset as a function of the proportion of data used to train the classifier, for clips of varying length (clip duration denoted in minutes). (**D**) Total number of annotated frames in each dataset. (**E**–**G**) Same as (**A**–**C**), but for the CRIM13 dataset. (**H**) Test set accuracy on the CRIM13 dataset as a function of the amount of training data for classifiers trained with features from the side camera, top camera, and both the side and top cameras. Lines and shaded regions in (**A,C,E,G,H**) indicate mean and standard error, respectively, across 10 random splits of the data.

(i.e., the test set). For both datasets, accuracy and F1 improve as more training data is used, with steep increases for the first ten percent of the data, and more gradual increases after twenty percent. Example classifier output and ground truth annotations are shown in Fig. 2B,F.

We then compare the performance of our model with existing ones (Table 1; see "Methods: Comparison with existing methods" section). On the home-cage dataset, in addition to showing higher accuracy than the agreement between human annotators, our classifier outperforms existing commercial options (HomeCageScan 2.0, CleverSys Inc., evaluated by Jhuang et al.[7]), as well as approaches based on hand-crafted features[7] and 3D convolutional neural networks[23]. We do note, however, that the hidden Markov model approach detailed in Jiang et al.[24] performed marginally better on the home-cage dataset than DeepAction. The classifier demonstrates above-human performance and surpasses the sparse spatio-temporal feature approach detailed in Burgos-Artizzu et al.[6] on the CRIM13 dataset. It also performs better than prior methods based on temporal features[25], independent component analysis[26], hierarchical sparse coding[27], integrated sparse and dense trajectory features[28].

Data input process improves performance. Next, we consider how unique aspects of our data preparation process affect the performance of the classifier. Specifically, we investigate our hypothesis that, given equal annotation time (i.e., an equal labeled proportion), our classifier shows superior performance when it is trained using relatively short clips rather than longer ones. As shown in Fig. 2C,G, this is indeed the case. While presumably there is a limit to this phenomenon (i.e., if the clip length were to be only a handful of frames, the classifier would fail to gain enough context to accurately predict its labels), in the clip durations tested here, varying between one and 20 min, shorter clips are both more accurate for a given level of annotation and demonstrate a more rapid improvement as training data increases. The CRIM13 dataset is recorded using synchronized top- and side-view cameras. In our main analysis we combine the features from both cameras (see "Methods: Feature extraction" section); in Fig. 2H we confirm that this is advantageous. The classifier trained using features from both views demonstrates superior performance to one trained only features from the side camera or only those from the top camera, indicating our method effectively integrates information from multiple cameras.

DeepAction performs well across behaviors. An important consideration, in addition to overall classifier performance, is classifier performance on specific behaviors. In highly imbalanced datasets (i.e., those in which a small number of behaviors are disproportionately common), high accuracy can be achieved by a classifier with poor discriminative capacity if its predictions are the most common classes. The home-cage dataset, except for the "drink" behavior (0.26 percent of labels), is relatively well-balanced (Fig. 3A). For non-drinking behaviors, the classifier shows consistently high performance (Fig. 3B), despite modest variation in the prevalence of each label. The CRIM13 dataset displays significantly less balance (Fig. 3D), with a high proportion of behaviors classified as "other" (denoting non-social behavior). The high incidence of the "other" behavior accounts for the high performance of the classifier at near-zero training data proportions (approximately 55 percent accuracy; Fig. 2E), and a disproportionately large number of social behaviors being incorrectly labeled as "other" by the classifier (Fig. 3E). We also note that the distribution of bout lengths (i.e., the number of frames for which a behavior consecutively occurs) predicted by the classifier is qualitatively similar to the true distribution of bout length for most behaviors (Fig. 3C,F). In the home-cage dataset we see that the classifier underpredicts bout lengths for the "rest" behavior, which has an exceptionally long average bout length (2,563 frames vs. an average of 88 frames for all other behaviors), despite its high performance in predicting the rest behavior overall

Model	Accuracy (%)
Home-cage	
Human	71.6
CleverSys commercial system[7]	61.0
Jhuang et al.[7]	78.3
Le and Murari[23]	73.5
Jiang et al.[24]	81.5
DeepAction	79.5
CRIM13	
Human	69.7
Burgos-Artizzu et al.[6]	62.6
Eyjolfsdottir et al.[25]	37.2
Zhang et al.[26]	61.9
Meng et al.[27]	68.6
DeepAction	73.9%

Table 1. Performance comparison with existing methods Shown is the accuracy of various annotation methods on both datasets. "Human" denotes the agreement between two human annotator groups (see "Methods: Inter-observer reliability" section). The accuracy for DeepAction on the home-cage and CRIM13 datasets is the mean accuracy from 12-fold and two-fold cross-validation, respectively, to provide a comparable reference to Jhuang et al.[7] and Burgos-Artizzu et al.[6] (see "Methods: Comparison with existing methods" section).

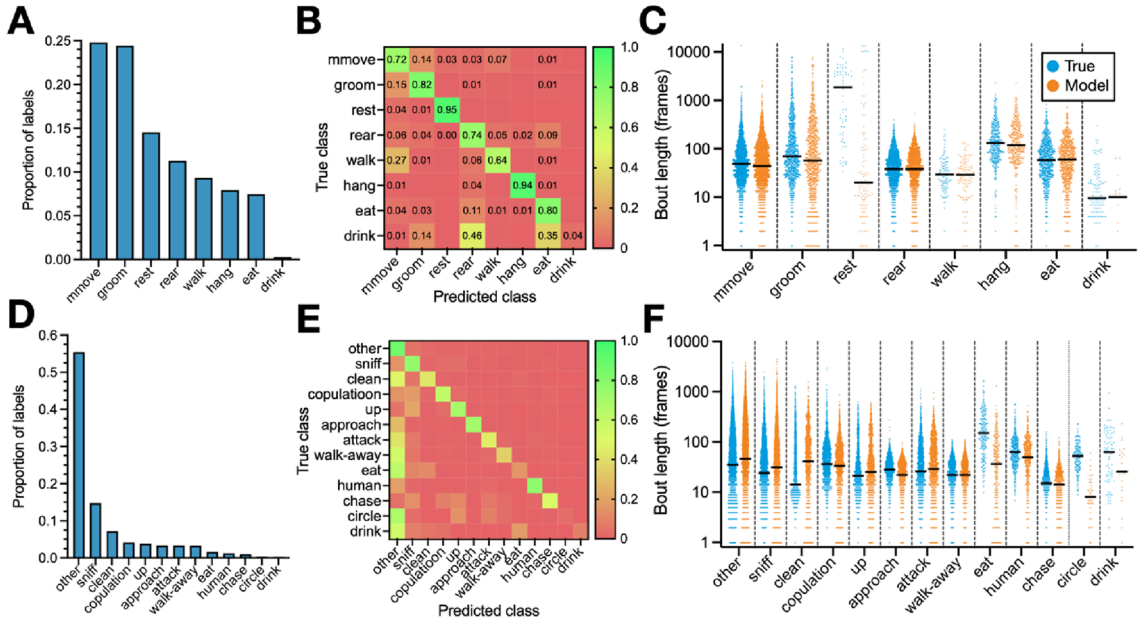

Figure 3. Dataset behavior characteristics and classifier performance. (**A**) Ground-truth distribution of behavior labels (i.e., the number of frames in which each behavior occurs as a proportion of the total number of frames in the dataset) for the home-cage dataset. (**B**) Example confusion matrix showing the classifier performance by behavior on the home-cage dataset, with cell values normalized relative to the true class. (**C**) True bout lengths and example predicted bout lengths for the home-cage dataset, grouped by behavior. A single "bout" refers to a period of continuously occurring behavior, and the corresponding bout length to the length of that period in number of frames. Median bout length is marked by the solid black lines, and each dot corresponds to a single bout. (**D**–**F**) Similar to (**A**–**C**), but for the CRIM13 dataset.

(recall: 0.95, precision: 0.98) on the same test set. In the CRIM13 dataset, we observe that the classifier underpredicts bout lengths for the behaviors it performs worst on: "eat," "human," and "drink."

To examine classifier performance as a function of the amount of data used to train it, we calculate the precision, recall, and F1 score (see "Methods: Classifier evaluation" section) for each behavior with varying labeled data proportions (Fig. 4). In the home-cage dataset, for non-drinking behaviors, we observe a similar pattern in behavior-level improvement as we do to overall accuracy—a rapid increase at low training data proportions, followed by a more gradual one at 10 to 20 percent training data (Fig. 4A–G). This pattern holds even given the relatively large difference in incidence between the least common (eat, at 7.5 percent of labels) and most common (micromovement, 24.8 percent of labels) non-drink behaviors. For drinking behavior, however, due to its exceptionally low incidence, we observe a more inconsistent, non-gradual improvement in performance across training set proportions (Fig. 4H).

This pattern generally applies in the CRIM13 dataset as well (Fig. S3). For most behaviors we observe a rapid increase in recall, precision, and F1, followed by a relative slowdown in improvement as a function of training proportion at a training proportion of approximately 0.3. There are notable exceptions to this pattern. First, we observe that, as compared to very low training proportions, the recall of "other" decreases slightly as the classifier defaulted to predicting "other" with disproportionate frequency (Fig. S3A). The F1 score, however, increased, indicating an improved balance between recall and accuracy. And second, we observe that "eat," "circle," and "drink" show sporadic improvements in recall, precision, and F1 as a function of training proportion (Fig. S3I,L,M). As with "drink" in the home-cage dataset, these are all low-incidence behaviors (approximately 2 percent of ground-truth labels or less), particularly in the case of "circle" and "drink" (approximately 0.3 percent of ground-truth labels).

DeepAction performs well on the exploratory dataset. On the exploratory dataset, we evaluated the classifier on a six-behavior dataset of seven octopus bimaculoides behavior videos collected in-house (see Fig. 5D). Overall, the classifier performs relatively well, with an accuracy of 73.1 percent; see the sample ethogram in Fig. 5C. This is much lower than human-level performance, however, given that manual annotators reached an agreement of 88.7 percent on the same, independently annotated video (see "Methods: Dataset" section). In terms of behavior-level performance, the classifier performs well on crawling, none (indicating behavior of interest) and fixed pattern, but poorly on relaxation, jetting, and expanding (Fig. 5B). The poor performance on these behaviors is likely due to their infrequency (Fig. 5A), particularly in the case of jetting and expanding.

Calibrated confidence scores accurately predict classification accuracy. Next, we turn our focus from the performance of the classifier to the performance of the confidence-based review. Recall that we generate a confidence score for each clip that represents the classifier's prediction of the accuracy of its predicted labels

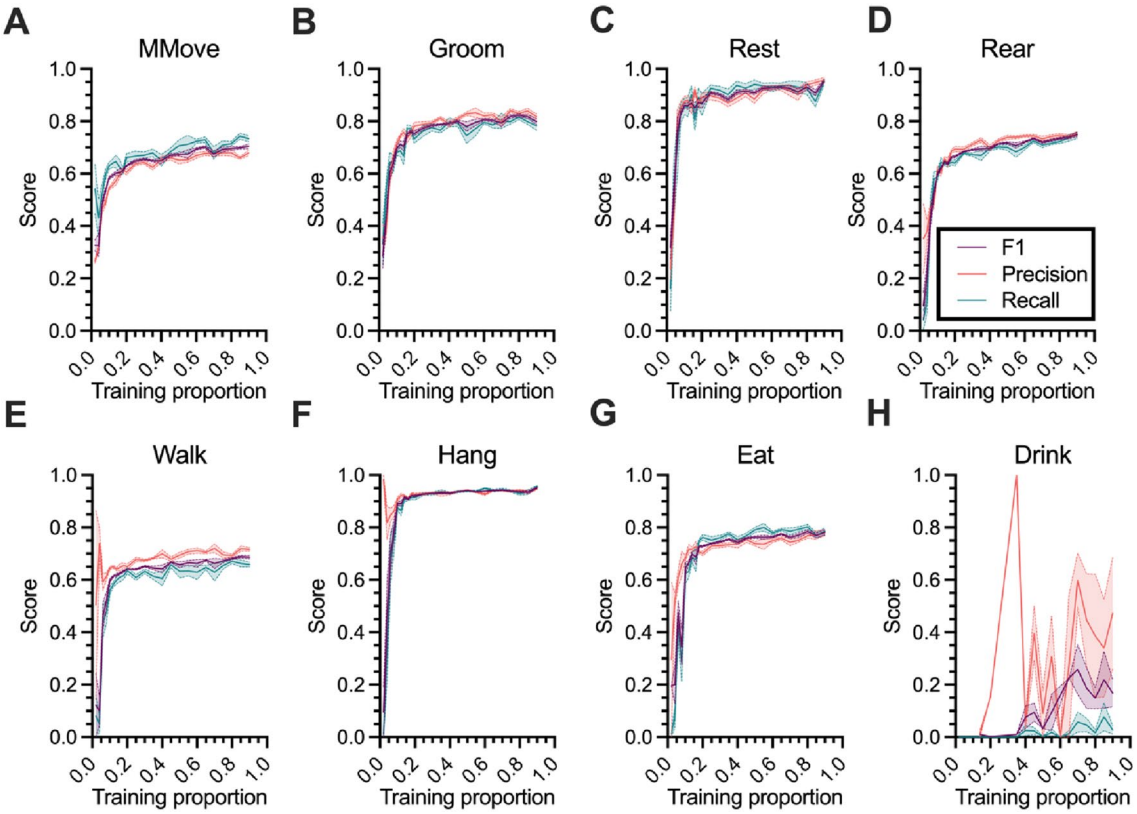

Figure 4. Home-cage behavior-level classifier performance. (**A**–**H**) Precision, recall, and F1 scores for each behavior in the home-cage dataset as a function of the proportion of data used to train the classifier. Lines and shaded regions indicate mean and standard error, respectively, across 10 random splits of the data.

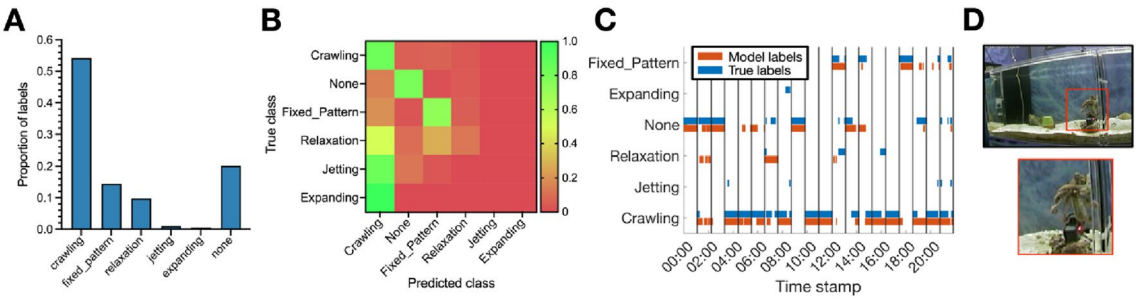

Figure 5. Exploratory dataset behavior characteristics and classifier performance. (**A**) Ground-truth distribution of behavior labels for the exploratory (octopus) dataset. (**B**) Confusion matrix of the classifier's performance on the test fold from tenfold cross validation. (**C**) Example ethogram of classifier labels and ground-truth annotations from 30 randomly selected octopus behavior clips (each one minute in duration, sampled at 10 frames per second). (**D**) Frame from an example octopus video. The red inset square in the top frame indicates the location of the animal, which is shown magnified below.

(see "Methods: Confidence score definition" section). In Fig. 6A,D we demonstrate that there is a strong correlation between confidence score and accuracy, for both confidence scores based on maximum softmax probability and those derived using temperature scaling (see "Methods: Confidence score calculation" section). We next consider the mean absolute error (MAE; see "Methods: Evaluating confidence score calibration" section) between clips' predicted accuracy (i.e., confidence score) and actual accuracy across training data proportions. Here, the MAE expresses the amount by which a randomly selected clip's confidence score differs (whether positively or negatively) from its accuracy. The MAE derived using temperature scaling performs slightly better than the one derived using softmax probabilities on the CRIM13 dataset (Fig. 6E) but not the home-cage dataset (Fig. 6B). While the MAE for both methods improves initially, it plateaus after the proportion of data labeled reaches about 20 percent, indicating that exact estimates of clip accuracy remain elusive.

Perhaps more important than predicting the accuracy of classifications on a single clip is predicting the accuracy of classifications across all unlabeled clips. While the absolute error of individual clips might fluctuate,

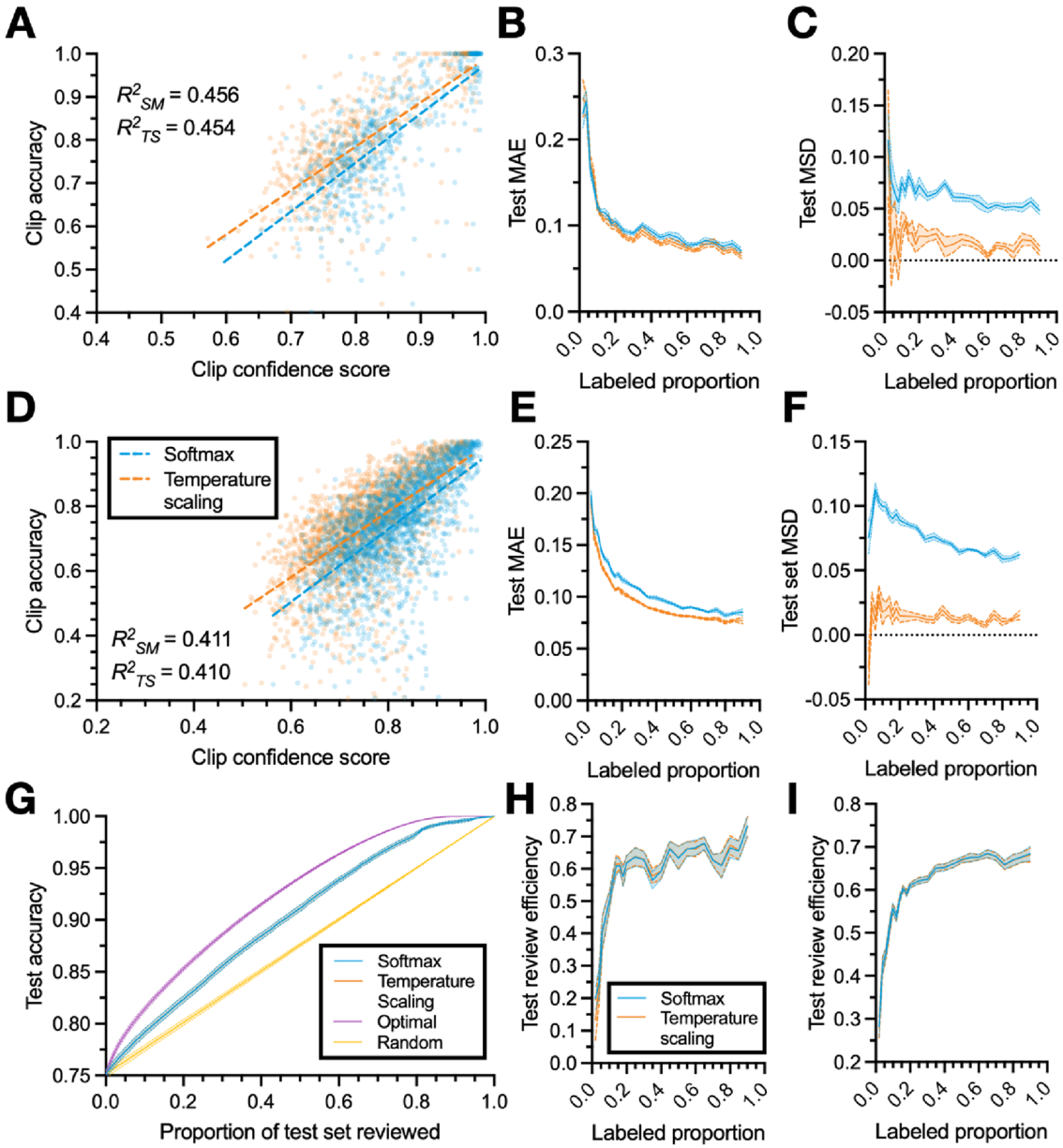

Figure 6. Confidence measure improvements across training proportions. (**A**) Example of the correlation between clip confidence score and clip accuracy. Dashed lines indicating the line of best-fit with r-squared values inset. (**B**) Mean absolute error (MAE) and (**C**) mean signed difference (MSD) between clip confidence score and clip accuracy as a function of the amount of data used to train the classifier. (**D**–**F**) Similar to (**A**–**C**), but for the CRIM13 dataset. (**G**) Example relationship between the proportion of test clips reviewed (and corrected) and test set accuracy from the home-cage dataset, where clips are reviewed in an order determined by the confidence scoring method, for various scoring methods (see "Methods: Confidence-based review" section). (**H**) Review efficiency metric, quantifying how effectively a given confidence scoring method performs when low-confidence clips are reviewed first (see "Methods: Evaluating review efficiency" section) as a function of the amount of training data, for the home-cage dataset. (**I**) Same as (**H**), but for CRIM13. Lines and shaded regions in (**B,C,E,F,H,I**) indicate mean and standard error across 10 random splits of the data.

if the differences cancel out (i.e., if predictions are just as likely to be overconfident as they are to be underconfident), the estimated accuracy of the set as whole will be accurate. This is useful in practice: if the confidence score is biased (e.g., it consistently over-estimates accuracy), then the estimated accuracy of the unlabeled data will systematically differ from its true accuracy. If the score is unbiased, however, then it is useful for evaluating whether the predicted agreement between classifier-produced and manually-produced annotations is sufficient for a given application. To investigate this, we consider the mean signed difference (MSD; see "Methods: Evaluating confidence score calibration" section), which quantifies the difference between the predicted accuracy of all predictions in the test set and the actual accuracy of the test set. As shown in Fig. 6C,F, the temperature scaling-based confidence score has a lower MSD than the softmax-based one, demonstrating that confidence

scores derived from temperature scaling are less (positively) biased. While the softmax score consistently overestimates the average accuracy of its predictions by approximately 6–8 percent regardless of training proportion, temperature scaling generally is generally overconfident by only 1–2 percent.

Uncertainty-based review reduces correction time. Having established the high correspondence between clip confidence score and clip accuracy, we investigate how well our confidence-based review system leverages those confidence scores to reduce the time it takes to review and correct classifier-produced labels. A viable confidence measure would allow clips with a lower confidence score (i.e., lower predicted accuracy) to be preferentially reviewed over those with a higher confidence score, decreasing the manual review time required to obtain acceptably high-quality annotations. Rather than reviewing all the classifier-produced labels, the user could instead review only a portion with the lowest accuracy (see "Methods: Confidence-based review" section). We provide an example of this process in practice in Fig. 6G, which simulates the relationship between the proportion of test video reviewed and the overall accuracy of the labels in the test set. If no video is reviewed, the average accuracy of the test set is the agreement between the classifier produced labels and the ground truth annotations. If one then begins to review and correct videos, the total accuracy increases, since we assume that incorrect classifier-produced labels are corrected. If videos are selected randomly, the relationship between the proportion of the test set reviewed and the test set accuracy is approximately linear—if each video selected is equally likely to have the same number of incorrect labels, then the increase in overall accuracy from correcting those labels is the same for all videos.

If, however, one sorts by confidence measure and reviews the lowest confidence clips first, then, ideally, the subset of videos reviewed will tend to be those with relatively lower accuracy than those not reviewed. The upper bound on the performance of the confidence-based review is a review where the clips are sorted by their actual accuracy (which is what the confidence score approximates). While this is unknown in practice (since the data being reviewed are unlabeled) we simulate it here to provide an upper bound for the performance of the confidence-based review. To compare the performance of the confidence-based review across labeled data proportions, we calculate a metric called "review efficiency" for each split of the data, which expresses the performance of the confidence score bounded by the best (optimal selection, review efficiency of 1) and worst (random selection, review efficiency of 0) possible performance (see "Methods: Evaluating review efficiency" section). As shown in Fig. 6H,I, as the proportion of data labeled increases, both confidence scores become closer to optimal in sorting videos for review. The softmax- and temperature scaling-based scores perform approximately the same.

Annotation GUI improves annotation and review. While we evaluate our method here using fully annotated datasets, the central purpose of this work is to improve the annotation of behavior in experimental settings. For this reason, we release the entire system as a MATLAB toolbox as a GitHub repository that includes example projects and GUI interfaces for defining the behavior set of interest (Fig. 7A) and conducting manual annotation and confidence-based review (Fig. 7B). For example, we integrate clip-wise annotation by pre-dividing project videos into clips and presenting clips, rather than videos, for users to annotate. In addition, we incorporate the confidence-based review process into the GUI: incomplete (i.e., unreviewed annotations) are shown in a table, with low-confidence clips (and their corresponding confidence scores) appearing at the top so that users can select them for review first. We also include information about the status of the project (e.g., number and duration of videos annotated, video and clip information, etc.) within the GUI. During confidence-based review, we also provide an estimate of $\text{acc}(\mathcal{D}^{\text{unlabeled}})$ directly, updating it as more annotations are completed. Users can easily load videos, annotate them using the keyboard, add or remove behaviors, and export the results entirely within the GUI.

Discussion

Here we present a method for the automatic annotation of laboratory animal behavior from video. Our classifier produces high accuracy annotations, rivaling or surpassing human-level agreement, while requiring relatively little human annotation time, and performs well across behaviors of varying incidence and timescale. Our confidence scores accurately predict accuracy and are useful in reducing the time required for human annotators to review and correct classifier-produced annotations. Finally, we release the system as an open-source GitHub repository, complete with an annotation GUI and example projects.

The primary strength of our method is the classifier's capacity to generate accurate classifications from raw video frames. By classifying behavior using raw frame information, DeepAction removes the need to annotate keypoints and create hand-crafted features that adequately encapsulate a given animals' behavioral repertoire. This removes both a tedious aspect of manual annotation (i.e., keypoint annotation in addition to behavioral annotation), and alleviates the need for researchers to construct behavior-encapsulating features, which is both time-consuming and often suboptimal. We also note that the performance of DeepAction surpasses that of approaches developed using hand-crafted features on both rodent datasets analyzed (Table 1), indicating that the automated feature extraction approach does not compromise performance. On the benchmark octopus dataset, we demonstrate the generalizability of the classifier to non-rodent animal models. We do, however, note that its performance is not as strong as on the rodent datasets. This is likely due either to the smaller amount of training data (6.15 h total), or the fact that the size of the octopus was smaller, relative to the field of view, than in the rodent datasets.

The base level performance of the classifier has the potential to significantly expedite the behavioral research process. Here, a useful benchmark is to compare the accuracy of the classifier (defined as the agreement between classifier-produced labels and the primary set of annotations; see "Methods: Inter-observer reliability" section) to the agreement between independent annotators (agreement between the primary set of annotations and a second,

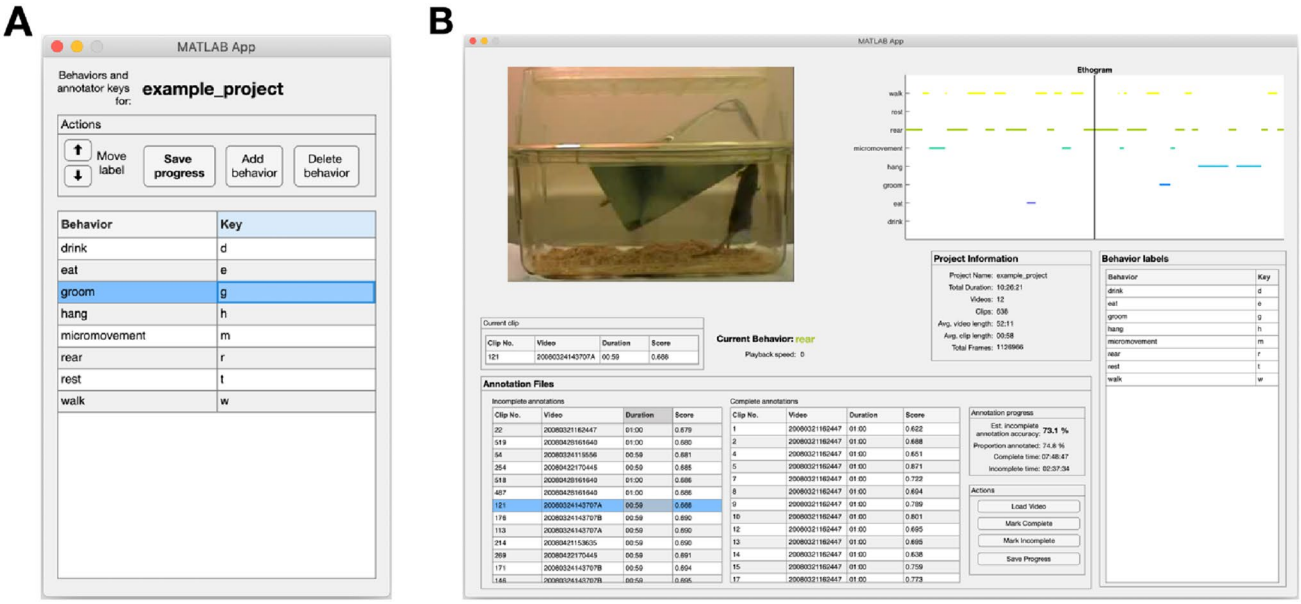

Figure 7. Example usage of the MATLAB apps included in the toolbox. (**A**) GUI for defining the set of behaviors in a dataset. Each behavior label corresponds to a unique keyboard key ("key"), which is used to designate the start and stop of behaviors during manual annotation. (**B**) An example of the annotation GUI used in confidence-based review to correct false classifier-produced predictions. It features tables of the complete (i.e., human annotated or reviewed) and unreviewed (i.e., classifier-annotated) clips in the project. During review, the tables include a confidence score for each clip ("score") as well as an estimated overall accuracy for all unannotated data. Users select clips to review from the annotation tables, which are then shown in the video viewer box (top left) along with their predicted labels. Users create or correct the labels of the behaviors appearing in the video, with both annotation and video playback controlled via keyboard. Behaviors and their corresponding keystrokes are shown in the "Behavior Labels" panel. After completing the annotation of each clip, users press the "Mark Complete" button to save their progress.

independent set used to evaluate inter-observer reliability). In our analysis, we find that the classifier requires that only 18 percent of data be annotated to surpass the agreement (71.6 percent) between human annotators on the home-cage dataset (see Fig. 2A). Given that the home-cage dataset took 264 h to manually annotate[7], if human-level agreement is defined as the threshold for acceptable annotations, our method would reduce this time to 47 h, saving researchers 82% of the time required to carry out the tedious step of video annotation. Similarly, DeepAction surpasses human-level agreement (69.7 percent) on the CRIM13 dataset with 25 percent of data annotated (see Fig. 2E), saving researchers 75% of their time. Since CRIM13 took 350 h to annotate[6], using our method instead of manual annotation would have reduced this time to 88 h, while maintaining annotation quality at the level of human annotators.

Our confidence-scoring system is important for two reasons. The first is a modest increase in review efficiency—if one is to manually review and check some number of automatically-generated behavioral labels, selecting those with the lowest confidence scores is preferable to doing so randomly. We show that this is true across training dataset sizes, and that review using confidence scores becomes closer to optimal as more data is annotated. The second, and perhaps more important, reason is that the temperature scaling-based confidence score generates an accurate estimate of the overall agreement between classifier- and human-produced labels on unlabeled data (i.e., where the "human-produced labels" are unknown). This means that researchers could annotate data until the estimated accuracy of the unlabeled data reached a given threshold of acceptable agreement for their given behavioral analysis, and then export the automated annotations without having to review and correct them.

Our tool has several practical advantages. First is a GUI for annotation and confidence-based review. Second is adaptability; in our GitHub release we provide additional pretrained CNNs (e.g., ResNet50 and Inception ResNetv2) with which potentially more useful features could be extracted, a computationally faster optical flow algorithm[29], and options to parallelize a number of the computationally intensive project functions (e.g., temporal frame generation and feature extraction). A final advantage is modularity: users can use the classification portion of the workflow without the review component, the annotator can for its interface alone, etc.

Though the toolbox presented here represents a significant advancement as compared to entirely manual annotation, there are several avenues for further exploration and potential improvement. While our clip selection process demonstrates superior performance to whole-video annotation, in the results here we select clips randomly. In practice, the confidence-based review system can be used to iteratively train the classifier (Fig. 1A), where low-confidence clips are reviewed, corrected, and used to re-train the classifier (though we do not explore whether this is preferable to random selection here). An alternative approach would be to adapt methods used in video summarization to cluster video clips by their similarity, and then select the subset of clips

best representative of the overall dataset[30,31]. While our classifier is based on a LSTM with bidirectional layers, it is possible that alternate architectures would demonstrate superior performance[11,32]. Relatedly, the classifier described here assumes behaviors are mutually exclusive; that is, none of the behaviors can occur at the same time. However, for datasets in which this is not the case, the cross-entropy loss function used here could easily be adjusted to allow for co-occurring behaviors. A final avenue for further exploration is our approach to calculating confidence scores. Though our system is already close to optimal, given enough training data (Fig. 6H,I), there are a number of density-based metrics[33] or those that utilize Bayesian dropout[34] that might provide superior performance to the temperature scaling-based one employ here.

Methods

Datasets. Given that rodents are widely used in behavioral research, and mice are the most studied rodents[35], we chose two publicly-available datasets featuring mice engaging in a range of behaviors in our main analysis. The first dataset, referred to as the "home-cage dataset," was collected by Jhuang et al.[7] and features 12 videos (approximately 10.5 h and 1.13 million frames in total) of singly housed mice in their home cages, recorded from the side view. Video resolution is 320 × 240 pixels. The authors annotate each video in full and identify eight mutually exclusive behaviors (Fig. S1A) of varying incidence (Fig. 3A). This dataset allows us to benchmark our approach against existing methods, allows us to evaluate our method on a common use-case, and is relatively well-balanced in terms of the incidence of each behavior.

The second dataset used is the Caltech Resident-Intruder Mouse dataset (CRIM13), collected by Burgos-Artizzu et al.[6]. It consists of 237 pairs of videos, recorded from synchronized top- and side-view cameras, at 25 frames per second and an 8-bit pixel depth. Videos are approximately 10 min long, and the authors label 13 mutually exclusive actions (Fig. S1B). Of these actions, 12 are social behaviors, and the remaining action is the category "other," which denotes periods where no behavior of interest occurs[6]. This dataset features a number of challenges absent from the Jhuang et al.[7] dataset. In addition to including social behavior (in contrast to the home-cage dataset, which features singly-housed mice), it presents two algorithmic challenges. First, videos are recorded using a pair of synchronized cameras. This allows us to test multiple-camera integration functionality (see "Methods: Feature extraction" section), to evaluate classifier performance using features from multiple cameras. And second, it is highly unbalanced, with a slight majority of all annotations being the category "other" (periods during which no social behavior occurred; Fig. 3D).

We also include an exploratory dataset, to demonstrate the applicability of our model to non-rodent models, comprised of seven unique videos of single-housed octopus bimaculoides during a study of octopus habituation behaviors in the Dartmouth octopus lab. One video (approximately 62 min in length) was annotated by two different annotators, allowing us to assess inter-observer reliability by calculating the agreement between these two independent annotations. The videos span approximately 6.75 h in total, with 6.15 h annotated. Video was recorded at 10 frames per second with a resolution of 640 × 436 pixels. We define five behaviors of interest: crawling, fixed pattern (crawling in fixed formation along the tank wall), relaxation, jetting (quick acceleration away from stimuli), and expanding (tentacle spread in alarm reaction or aggressive display), and an indicator for when none of these behaviors occur (none). In the original dataset, there were three additional behaviors (inking/jetting, display of dominance, color change), comprising a very small number of the total frames, which could co-occur with the other six behaviors (crawling, fixed pattern, relaxation, jetting, expanding, none). However, because our classification model can only predict mutually-exclusive classes at the current time, we removed these three behaviors from our input annotations.

Inter-observer reliability. Both datasets include a set of annotations performed by two groups of annotators. The primary set of annotations was produced by the first group of annotators and includes all video in the dataset. The secondary set of annotations was performed by a second, independent set of annotators on a subset of videos. We use the primary set of annotations to train and evaluate our method, and the secondary set to establish inter-observer reliability; that is, how much two, independent human annotator's annotations can be expected to differ. Given this, classifier-produced labels can be most precisely interpreted as the predicted behavior *if the video was annotated by the first group of annotators*. This distinction becomes important because we benchmark the accuracy of our method (i.e., the agreement between the classifier's predictions and the primary set of annotations) relative to the inter-observer agreement (i.e., the agreement between the first and second group of annotators, on the subset of video labeled by both groups). So, for example, when we note that our model achieves accuracy "above human agreement," we mean that our classifier predicts the labels from the first human annotator group better than the second human annotator group does. In the case of the home-cage dataset, the agreement between the primary and secondary sets was 78.3 percent, compared on a 1.6 h subset of all dataset video[7]. For CRIM13, agreement was 69.7 percent, evaluated on a random selection of 12 videos[6].

Simulating labeled data. To simulate our approach's performance with varying amount of training data, in our primary analyses we train the classifier using the following amounts of labeling:

$$\text{prop}_{\text{labeled}} = [0.02 : 0.02 : 0.2, 0.2 : 0.05 : 0.9].$$

That is, we use a proportion of all data, $\text{prop}_{\text{labeled}}$, to construct our training and validation sets (i.e., $\mathcal{D}^{\text{labeled}}$), and the remaining $1 - \text{prop}_{\text{labeled}}$ data to create our test set, $\mathcal{D}^{\text{test}}$ (Fig. 1B,D). We use an increment of 0.02 for low training proportions (up to 0.20), because that is when we see the greatest change relative to a small change in added training data (Fig. 2A,E). We increment values from 0.25 to 0.90 by 0.05. This gives us a set of 24 training proportions per analysis. Additionally, for each training proportion, unless otherwise noted, we evaluate the model on 10 random splits of the data. In our main analyses, we use a clip length of one minute for both datasets.

Comparison with existing methods. When comparing our model to existing methods, we employ *k*-fold validation instead of evaluating on random splits of the data. In the case of the home-cage dataset, the existent methods cited employ a "leave one out" approach—using 11 of the 12 videos to train their methods, and the remaining video to test it. In our approach, however, we rely on splitting the data into clips, so instead we use 12-fold cross-validation, where we randomly split the dataset clips into 12 folds and then employ cross-validation on the clips, rather than entire videos. In evaluating their approach's performance on the CRIM13 dataset, Burgos-Artizzu et al.[6] selected 104 videos for training and 133 for testing, meaning that they trained their program on 44 percent of the data, and tested it on 56 percent. Here, we evaluate our method relative to theirs using two-fold cross validation (50 percent test and 50 percent train split) to retain similar levels of training data.

Frame extraction. To generate spatial frames, we extract raw video frames from each video file. Rather than save each image as an image file in a directory, we save the entire sequence of images corresponding to a single video to a sequence file, using the implementation provided by Dollár[36] with JPG compression. This has the advantage of making the video frames easier to transfer between file systems and readable on any operating system (which is useful for users running the toolbox on high performance computing clusters). To generate the temporal component, we use the TV-L1 algorithm[19,37], which shows superior performance to alternate optical flow algorithms[15], to calculate the dense optical flow between pairs of sequential video frames and represent it visually via the MATLAB implementation by Cun[38]. In the visual representation of optical flow fields, hue and brightness of a pixel represent the orientation and magnitude of that pixel's motion between sequential frames. By representing motion information of the video as a set of images, we can use a similar feature extraction method for both spatial and temporal frames. Just as the features derived from the spatial images represent the spatial information in the video, the features derived from the temporal images should provide a representation of the motion information in the video.

Feature extraction. We utilize the pretrained ResNet18 convolutional neural network (CNN) to extract high-level features from the spatial and temporal video frames. Often used in image processing applications, CNNs consist of a series of layers that take an image as an input and generate an output based on the content of that image. Intuitively, classification CNNs can be broken down into two components: feature extraction and classification. In the feature extraction component, the network uses a series of layers to extract increasingly complex features from the image. In the classification component, the network uses the highest-level features to generate a final classification for the image (e.g., "dog" or "cat"). In the case of pretrained CNNs, the network learns to extract important features from the input image through training—by generating predictions for a set of images for which the ground truth is known, and then modifying the network based on the deviation of the predicted classification from the true classification, the network learns which features in the image are important in discriminating one object class from another. In pretrained CNNs, such as the ResNet18, which was trained to categorize millions of images from the ImageNet database into one thousand distinct classes[39], early layers detect generic features (e.g., edges, textures, and simple patterns) and later layers represent image data more abstractly[40].

Here, we leverage transfer learning—where a network trained for one context is used in another—to extract a low-dimensional representation of the data in the spatial and temporal video frames. The idea is that, since the ResNet18 is trained on a large, general object dataset, the generality of the network allows us to obtain an abstract representation of the salient visual features in the underlying video by extracting activations from the later layers of the network in response to a completely different set of images (in this case, laboratory video of animal behavior). To extract features from the ResNet18 network for a given image, we input the image into the network and record the response ("activations") from a specified layer of the network. In this work, we chose to extract activations from the global average pooling layer ("pool5" in MATLAB) of the ResNet18, close to the end of the network (to obtain high-level feature representations). This generates a feature vector of length 512, representing high level CNN features for each image.

By default, the ResNet18 accepts input images of size $[224, 224, 3]$ (i.e., images with a width and height of 224 pixels and three color channels), so we preprocess frames by first resizing them to a width and height of 224 pixels. In the case of spatial frames, the resized images are input directly into the unmodified network. For temporal frames, however, rather than inputting frames into the network individually, we "stack" each input frame to the CNN with the five frames preceding it and the five frames following it, resulting in an input size of $[224, 224, 33]$. This approach allows the network to extract features with longer-term motion information and has been shown to improve discriminative performance[14,18]. We select a stack size of 11 based on the findings from Simonyan and Zisserman[18]. By default, the ResNet18 network only accepts inputs of size $[224, 224, 3]$, so to modify it so that it accepts inputs of size $[224, 224, 33]$ we replicate the weights of the first convolutional layer (normally three channels) 11 times. This allows the modified "flow ResNet18" to accept stacks of images as inputs, while retaining the pretrained weights to extract salient image features.

After spatial features and temporal features have been separately extracted from the spatial and temporal frames, respectively, we combine them to produce the *spatiotemporal* features that will be used to train the classifier (Fig. 1E). To do so, we simply concatenate the spatial and temporal features for each frame. That is, for a given segment of video with n frames, the initial spatiotemporal features are a matrix of size $[n, 512 \times 2] = [n, 1024]$, where 512 represents the dimensionality of the features extracted from the ResNet18. If multiple synchronized cameras are used (as is the case in one of our benchmark datasets), we employ the same process, concatenating the spatial and temporal features for each frame *and each camera*. In the case of two cameras, for example, this implies the initial spatiotemporal features is a matrix of size $[n, 512 \times 2 \times 2] = [n, 2048]$. To decrease training time, memory requirements, and improve performance[41,42], we utilize dimensionality reduction to decrease the

size of the *initial* spatiotemporal features to generate *final* spatiotemporal features of size $[n, 512]$. We selected reconstruction independent component analysis[43,44] as our dimensionality reduction method, which creates a linear transformation by minimizing an objective function that balances the independence of output features with the capacity to reconstruct input features from output features.

Classifier architecture. The labeled and unlabeled data consist of a set of clips, generated from project video, which the classifier uses to predict behavior. Clips in $\mathcal{D}^{\text{labeled}}$ and $\mathcal{D}^{\text{unlabeled}}$ are both constituted of a segment of video and a corresponding array of spatiotemporal features extracted from that video. Clips in $\mathcal{D}^{\text{labeled}}$ also include an accompanying set of manual annotations (Fig. 1B). For a given clip in $\mathcal{D}^{\text{labeled}}$ with n_{labeled} frames, the classifier takes a $[n_{\text{labeled}}, 512]$-dimensional vector of spatiotemporal features (Fig. 1E) and a one-dimensional array of n_{labeled} manually-produced labels (e.g., "eat," "drink," etc.) as inputs, and learns to predict the n_{labeled} labels from the features. After training, for a given clip in $\mathcal{D}^{\text{unlabeled}}$ with $n_{\text{unlabeled}}$ frames, the classifier takes as an input a $[n_{\text{unlabeled}}, 512]$-dimensional vector of spatiotemporal features and outputs a set of $n_{\text{unlabeled}}$ behavioral labels, corresponding to the predicted behavior in each of the $n_{\text{unlabeled}}$ frames. To implement this transformation from features to labels, we rely on recurrent neural networks (RNNs). Prior to inputting clips into the RNN, we further divide them into shorter "sequences," corresponding to 15 s of video to reduce overfitting[45] and sequence padding[46]. Unlike traditional neural networks, recurrent neural networks contain cyclical connections which allows information to persist over time, enabling them to learn dependencies in sequential data[47]. Given that predicting behavior accurately requires the integration of information over time (i.e., annotators generally must view more than one frame to classify most behavior, since behaviors are often distinguished by movement over time), this persistence is critical.

We opt for a long short-term memory (LSTM) network with bidirectional LSTM layers (BiLSTM) as the core of our classification model. LSTMs are better able to learn long-term dependencies in data than traditional RNNs in practice[48,49], and the use of bidirectional layers allows the network to process information in both temporal directions[50] (i.e., forward and backward in time, rather than forward only in the case of a traditional LSTM layer). As shown in Figure S4, our network's architecture begins with a sequence input layer, which accepts a two-dimensional array corresponding to spatiotemporal video features (with one row per frame and one column per feature). We then apply two BiLSTM layers, which increases model complexity and allows the model to learn more abstract relationships between input sequences and correct output labels[51]. To reduce the likelihood of model overfitting, we use a dropout layer after each BiLSTM layer, which randomly sets some proportion of input units (here, 50 percent) to 0, which reduces overfitting by curbing the power of any individual neuron to generate the output[52]. The second dropout layer is followed by a fully-connected layer with an output size of $[n, K]$, where K is the number of classes and n is the number of frames in the input clip. The softmax layer then normalizes the fully-connected layer's output into a set of class probabilities with shape $[n, K]$, where the sum of each row is 1 and the softmax probability of class k in frame j is given by the entry jk. Following the softmax layer, the sequence-to-sequence classification layer generates a one-dimensional categorical array of n labels corresponding to the behavior with the highest softmax probability in each frame. We select cross-entropy loss for K mutually exclusive classes[53] as our loss function, since the behaviors in both datasets are mutually exclusive. All classifiers were trained using a single Nvidia Tesla K80 GPU running on the Dartmouth College high performance computing cluster.

Classifier training. In this analysis, we use the hyperparameters specified in Table 2 when training the network. To avoid overfitting, we select 20 percent of $\mathcal{D}^{\text{labeled}}$ to use in our validation set (i.e., $prop_{train} = 0.20$; see Fig. 1C). We then evaluate the network on this validation set every epoch (where "epoch" is defined as a single pass of the entire training set through the network) and record its cross-entropy loss. If the loss on the validation set after a given epoch is larger than or equal to the smallest previous loss on the validation set more than twice, training terminates.

Hyperparameter	Value
Maximum epochs	16
Validation frequency (per epoch)	1
Validation patience	2
Initial learning rate	0.001
Learning rate drop period	4
Learning rate drop factor	0.1
Minibatch size	8

Table 2. Default hyperparameters. The maximum number of epochs the network can be trained for is 16. The cross-entropy loss of the validation dataset is calculated for each epoch, and if this value is less than the prior minimum validation loss for more than two epochs, training terminates. The initial learning rate is 0.001, and every four epochs the learning rate drops by a factor of ten. We use a minibatch size of eight to minimize deficits in generalizability that could occur at larger values[54].

Classifier evaluation. To evaluate the classifier, we consider its performance on the test set, $\mathcal{D}^{\text{test}}$ (Fig. 1B,D). For each clip, the classifier outputs a set of predicted labels for each frame, corresponding to the predicted behavior in that frame. In evaluating the classifier, we are interested in how closely these predicted labels match the true ones. We first consider overall prediction accuracy. We let correct denote the number labels in which the network's prediction is the same as the true label and incorrect the number of labels in which the network's prediction is not the same as the true label. Then accuracy can be quantified as the following proportion:

$$\text{accuracy} = \frac{\text{correct}}{\text{correct} + \text{incorrect}}.$$

Next, we consider the performance of the network by behavior. To do so, we let TP_k denote the number of true positives (predicted class k and true class k), FP_k the number of false positives (predicted class k, but true label not class k), and FN_k the number of false negatives (true class k, predicted not class k) for class k (where k is between 1 and the total number of classes, K).

We then calculate the precision, recall, and F1 score for each label[11,55], where the precision and recall for class k are defined as follows:

$$\text{precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k},$$

$$\text{recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}.$$

Precision is the proportion of correct predictions out of all cases in which the *predicted class* is class k. Recall, meanwhile, denotes the proportion of correct predictions out of all the cases in which the *true class* is class k. From the precision and recall, we calculate the F1 score for class k. The F1 score is the harmonic mean of precision and recall, where a high F1 score indicates both high precision and recall, and deficits in either decrease it:

$$\text{F1}_k = 2 \cdot \frac{\text{precision}_k \cdot \text{recall}_k}{\text{precision}_k + \text{recall}_k}$$

After calculating the F1 score for each class, we calculate the average F1 score, F1_{all} as follows: $\text{F1}_{\text{all}} = \frac{1}{K}\sum_{k=1}^{K} \text{F1}_k$.

Confidence score definition. For each input clip, the classifier returns a set of predicted annotations corresponding to the predicted behavior (e.g., "walk," "drink," "rest," etc.) occurring in each frame of that clip. We denote the set of classifier-predicted labels for clip number i, clip_i, as $\{\widehat{y}_j | j \in \text{clip}_i\}$. Each clip also has a set of "true" labels, corresponding to those that would be produced if the clip was manually annotated. In the case of the labeled data, the true labels are known (and used to train the classifier). In the case of unlabeled data, they are not known (prior to manual review). We denote the set of true labels for clip_i as $\{y_j | j \in \text{clip}_i\}$. For each frame in a clip, in addition to outputting a prediction for the behavior occurring in that frame, we also generate an estimate of how likely that frame's classifier-assigned label is correct. That is, for each clip, we generate a set of predicted probabilities $\{\widehat{p}_j | j \in \text{clip}_i\}$ such that $\widehat{p}_j$ denotes the estimated likelihood that $\widehat{y}_j$ is equal to y_j. In an optimal classifier, $\mathbb{P}(\widehat{y}_j = y_j) = \widehat{p}_j$. That is, $\widehat{p}_j$ is an estimate of the probability the classification is correct; and, in an optimal confidence-scorer, the estimated probability the classification is correct will be the ground truth likelihood the classification is correct[56].

Now that we have established an estimated probability that a given *frame* in a clip is correct, we extend the confidence score to an entire clip. As in training data annotation, the review process is conducted at the level of an entire clip, not individual video frames. That is, even if there are a handful of frames in a clip that the classifier is relatively unconfident about, we assume that a human reviewer would need to see the entire clip to have enough context to accurately correct any misclassified frames. Since $\widehat{p}_j$ is the estimated probability a given frame j is correct, it follows that the average $\widehat{p}_j$ for $j \in \text{clip}_i$ is the estimated probability a randomly selected frame in clip_i is correct. We define this quantity to be the clip confidence score; formally, $\text{conf}(\text{clip}_i) = \frac{1}{|\text{clip}_i|}\sum_{j \in \text{clip}_i} \widehat{p}_j$, where $\text{conf}(\text{clip}_i)$ is the clip confidence score of clip_i and $|\text{clip}_i|$ is the number of frames in clip_i. We then consider that accuracy is the true probability a randomly selected frame in clip_i is correct by definition. That is, $\text{acc}(\text{clip}_i) = \frac{1}{|\text{clip}_i|}\sum_{j \in \text{clip}_i} \mathbf{I}(\widehat{y}_j = y_j)$, where $\text{acc}(\text{clip}_i)$ is the accuracy of clip_i and $\mathbf{I}$ is the indicator function. In the case of an optimal confidence score, we'll have that $\text{conf}(\text{clip}_i) = \text{acc}(\text{clip}_i)$. If we compare $\text{conf}(\text{clip}_i)$ with $\text{acc}(\text{clip}_i)$ on our test data, we can establish how well the confidence score can be expected to perform when the ground truth accuracy, $\text{acc}(\text{clip}_i)$, is unknown. In Methods: Confidence score calculation, we discuss our approach for obtaining $\widehat{p}_j$, after which finding clip-wise confidence scores is trivial.

Confidence score calculation. Here, we first examine how to calculate the frame-wise confidence score $\widehat{p}_j$. To do so, we consider the classifier structure (Fig. S4) in more detail. In particular, we focus on the last three layers: the fully-connected layer, the softmax layer, and the classification layer. To generate a classification for a given frame, the softmax layer takes in a logits vector from the fully-connected layer. This logits vector represents the raw (unnormalized) predictions of the model. The softmax layer then normalizes these predictions into a set of probabilities, where each probability is proportional to the exponential of the input. That is, given K classes, the K-dimensional vector from the fully-connected layer is normalized to a set of probabilities, representing the

probability of each class. The class with the highest probability is then returned as the network's predicted label (e.g., "eat" or "walk") for that frame. We can then interpret this probability as a confidence score derived from the softmax function[56]. Formally, if we let logits vector $\boldsymbol{z}_j$ represent the output from the fully-connected layer corresponding to frame j, the softmax-estimated probability that the predicted label of frame j is correct is $\widehat{p}_j^{\mathrm{SM}} = \max_k \sigma(\boldsymbol{z}_j)^{(k)}$, where σ is the softmax function. We refer to this confidence score as the "max softmax score," since it is derived from the maximum softmax probability.

One of the challenges with using the max softmax probability as a confidence score, however, is that it is often poorly scaled. Ideally, estimated accuracy for a prediction would closely match its actual expected accuracy, but in practice the softmax function tends to be "overconfident"[56]. That is, $\widehat{p}_j^{\mathrm{SM}}$ tends to be larger than $\mathbb{P}(\widehat{y}_j = y_j)$. To generate a more well-calibrated confidence score (i.e., one in which $\widehat{p}_j$ is closer to $\mathbb{P}(\widehat{y}_j = y_j)$, we use an approach called temperature scaling. Temperature scaling uses a learned parameter T (where $T > 1$ indicates decreased confidence and $T < 1$ increased confidence) to rescale class probabilities so that the confidence score more closely matches the true accuracy of a prediction[57]. We define the temperature scaling-based confidence for frame j as $\widehat{p}_j^{\mathrm{TS}} = \max_k \sigma(\boldsymbol{z}_j/T)^{(k)}$, where T is selected to minimize the negative log likelihood on the validation set. Now that we have established the process for generating a frame-wise confidence score, we can generate the clip-wise confidence score that is used in the confidence-based review. As previously described, for clip_i this is simply $\mathrm{conf}(\mathrm{clip}_i) = \frac{1}{|\mathrm{clip}_i|}\sum_{j\in\mathrm{clip}_i}\widehat{p}_j$, where $\widehat{p}_j$ is either generated via the softmax function ($\widehat{p}_j = \widehat{p}_j^{\mathrm{SM}}$) or temperature scaling ($\widehat{p}_j = \widehat{p}_j^{\mathrm{TS}}$).

Confidence-based review. Now that we have generated a confidence score for a given clip, we use it in two ways. First, recall that one of the purposes of the confidence-based review is to estimate the accuracy of the unlabeled data, $\mathcal{D}^{\mathrm{unlabeled}}$. If, for example, a user decided that an accuracy of 80 percent was acceptable for their given behavior analysis application (i.e., $\mathrm{acc}(\mathcal{D}^{\mathrm{unlabeled}}) \geq 0.8$), then given an acceptably reliable confidence score, unlabeled data for which $\mathrm{conf}(\mathcal{D}^{\mathrm{unlabeled}}) \geq 0.8$ would be sufficient for export and use in their given analysis without manual review. Before obtaining an estimate for $\mathrm{conf}(\mathcal{D}^{\mathrm{unlabeled}})$, we first consider that the true (unknown) accuracy of the annotations in $\mathcal{D}^{\mathrm{unlabeled}}$ is the weighted sum of the accuracies of the clips in $\mathcal{D}^{\mathrm{unlabeled}}$, where weight is determined by the number of frames in each clip. Formally, we can express the accuracy of $\mathcal{D}^{\mathrm{unlabeled}}$ as:

$$\mathrm{acc}\left(\mathcal{D}^{\mathrm{unlabeled}}\right) = \sum\nolimits_{i\in\mathcal{D}^{\mathrm{unlabeled}}}\left(\mathrm{acc}(\mathrm{clip}_i) \times \frac{|\mathrm{clip}_i|}{\sum_{j\in\mathcal{D}^{\mathrm{unlabeled}}}\left|\mathrm{clip}_j\right|}\right),$$

where $\frac{|\mathrm{clip}_i|}{\sum_{j\in\mathcal{D}^{\mathrm{unlabeled}}}|\mathrm{clip}_j|}$ weights the accuracy of $\mathrm{acc}(\mathrm{clip}_i)$ by the number of frames in clip_i (i.e., $|\mathrm{clip}_i|$) relative to the total number of clips (i.e., $\sum_{j\in\mathcal{D}^{\mathrm{unlabeled}}}\left|\mathrm{clip}_j\right|$). We then estimate the accuracy of the unlabeled data by substituting the known $\mathrm{conf}(\mathrm{clip}_i)$ for the unknown $\mathrm{acc}(\mathrm{clip}_i)$:

$$\mathrm{conf}\left(\mathcal{D}^{\mathrm{unlabeled}}\right) = \sum\nolimits_{i\in\mathcal{D}^{\mathrm{unlabeled}}}\left(\mathrm{conf}(\mathrm{clip}_i) \times \frac{|\mathrm{clip}_i|}{\sum_{j\in\mathcal{D}^{\mathrm{unlabeled}}}\left|\mathrm{clip}_j\right|}\right).$$

In this way, $\mathrm{conf}(\mathcal{D}^{\mathrm{unlabeled}})$ represents the approximate accuracy of the classifier on unlabeled data. If the confidence score functions well, then $\mathrm{conf}(\mathcal{D}^{\mathrm{unlabeled}})$ will closely match $\mathrm{acc}(\mathcal{D}^{\mathrm{unlabeled}})$.

Next, we consider the confidence-based review. In this component of the workflow, user can review and correct labels automatically generated by the classifier for $\mathcal{D}^{\mathrm{unlabeled}}$. A naïve approach would be to review all the video clips contained in $\mathcal{D}^{\mathrm{unlabeled}}$. While this would indeed ensure all the labels produced by the classifier are correct, if $\mathcal{D}^{\mathrm{unlabeled}}$ is large it can prove quite time-consuming. So instead, we leverage confidence scores to allow users to only annotate the subset of clips with relatively low confidence scores (i.e., relatively low predicted accuracy), for which review is most productive, while omitting those with relatively high confidence scores.

If a user reviews only a portion of the clips, it should be the portion with the lowest accuracy, for which correction is the most important. To express this formally, consider an ordered sequence of the n clips in $\mathcal{D}^{\mathrm{unlabeled}}$, $(\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_n)$, sorted in ascending order by accuracy (i.e., $\mathrm{acc}(\mathrm{clip}_i) \leq \mathrm{acc}\left(\mathrm{clip}_j\right)$, for $i < j$ and all $i, j \leq n$). If we review only k of the n clips, where $k \leq n$, we are best off reviewing clips $\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_k$ from the list since they have the lowest accuracy. For unlabeled data, however, recall that we can't precisely sort clips by accuracy, since without ground truth annotations $\mathrm{acc}(\mathrm{clip}_i)$ is unknown. However, since $\mathrm{conf}(\mathrm{clip}_i)$ approximates $\mathrm{acc}(\mathrm{clip}_i)$, we can instead sort unlabeled clips by their (known) confidence scores, and then select the clips with the lowest confidence scores to review first. This forms the basis of the confidence-based review. Given a set of clips in $\mathcal{D}^{\mathrm{unlabeled}}$, we simply create a sequence of clips $(\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_n)$ sorted by confidence score (i.e., such that $\mathrm{conf}(\mathrm{clip}_i) \leq \mathrm{conf}(\mathrm{clip}_j)$, for all $i < j$) and then have users review clips in ascending order. If the confidence score is an effective estimate of the clip accuracies, sorting based on confidence score will approximate sorting by accuracy.

Evaluating confidence score calibration. To examine the relationship between confidence scores and accuracy, we first consider the relationship between individual clips' predicted accuracy (as derived from confidence cores) and actual accuracy. The prediction error (PE) for a given clip is defined as the signed difference between its predicted accuracy and its actual accuracy. For clip_i, the PE is then $\mathrm{PE}(\mathrm{clip}_i) = \mathrm{conf}(\mathrm{clip}_i) - \mathrm{acc}(\mathrm{clip}_i)$.

Positive values indicate an overconfident score, and negative value and underconfident one. The absolute error (AE) is the magnitude of the prediction error and is defined as $\mathrm{AE}(\mathrm{clip}_i) = |\mathrm{PE}(\mathrm{clip}_i)|$. The AE is always positive, with a higher $\mathrm{AE}(\mathrm{clip}_i)$ indicating a greater absolute deviation between $\mathrm{conf}(\mathrm{clip}_i)$ and $\mathrm{acc}(\mathrm{clip}_i)$.

While PE and AE are defined for a single clip, we also consider the mean absolute error and mean prediction error across all the clips in $\mathcal{D}^{\mathrm{unlabeled}}$. Here, we let $\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_n$ denote a set of n clips. The mean absolute error (MAE) is defined as $\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{AE}(\mathrm{clip}_i)$. MAE expresses the average magnitude of the difference between predicted accuracy and actual accuracy for a randomly selected clip in the set. So, for example, if $\mathrm{MAE} = 0.1$, then a randomly selected clip's confidence score will differ from its accuracy score by about 10 percent, in expectation. The mean signed difference (MSD), meanwhile, is defined as $\mathrm{MSD} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{PE}(\mathrm{clip}_i)$. MSD expresses the signed difference between the total expected accuracy across clips and the total actual accuracy. So, for example, is $\mathrm{MSD} = -0.05$, then the total estimated accuracy of annotations for the set $\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_n$ is five percent lower than the true accuracy.

Evaluating review efficiency. To develop a metric for the performance of the confidence-based review, we first consider a case where a user has generated predicted labels for n clips, which have not been manually labeled, and selects k of them to review, where $k \leq n$. The remaining $n - k$ clips are not reviewed and are exported with unrevised classifier-generated labels. Then, for each of the k clips the user has selected, he or she reviews the clip and corrects any incorrect classifier-generated labels. In this formation, after reviewing a given clip, that clip's accuracy (defined as the agreement between a clip's labels and the labels produced by manual annotation), is 1, since any incorrect classifier-produced labels would have been corrected.

Next, we assume that we have been provided with a *sequence* of n clips, $\mathcal{D} = (\mathrm{clip}_1, \mathrm{clip}_2, \ldots, \mathrm{clip}_n)$, from which we select the first k clips in the sequence to review. If we denote $\mathrm{clip}_i^{\mathrm{unrev}}$ as clip i prior to being reviewed, and $\mathrm{clip}_i^{\mathrm{rev}}$ as clip i after being reviewed, then we can express the sequence of the first k clips after they have been reviewed as $\mathcal{D}_k^{\mathrm{rev}} = (\mathrm{clip}_1^{\mathrm{rev}}, \mathrm{clip}_2^{\mathrm{rev}}, \ldots, \mathrm{clip}_k^{\mathrm{rev}})$. We then express the remaining $n - k$ clips as the sequence $\mathcal{D}_k^{\mathrm{unrev}} = (\mathrm{clip}_{k+1}^{\mathrm{unrev}}, \mathrm{clip}_{k+2}^{\mathrm{unrev}}, \ldots, \mathrm{clip}_n^{\mathrm{unrev}})$. We then consider that the overall accuracy of the sequence of clips, $\mathrm{acc}(\mathcal{D})$, is simply weighted average of the accuracy of the reviewed videos, $\mathcal{D}_k^{\mathrm{rev}}$, and the unreviewed ones, $\mathcal{D}_k^{\mathrm{unrev}}$, where the weight is a function of the number of frames in each clip. Formally,

$$\mathrm{acc}(\mathcal{D}_k) = \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{rev}}\right) \times \frac{\left|\mathcal{D}_k^{\mathrm{rev}}\right|}{|\mathcal{D}|} + \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{unrev}}\right) \times \frac{\left|\mathcal{D}_k^{\mathrm{unrev}}\right|}{|\mathcal{D}|},$$

where $|\mathcal{D}|$ is the total number of video frames in the clips in set $\mathcal{D}$ (i.e., $|\mathcal{D}| = \sum_{i\in\mathcal{D}} |\mathrm{clip}_i|$). We then consider that, after reviewing and correcting the first k clips, the accuracy of each reviewed clip is now 1. That is, $\mathrm{acc}\left(\mathrm{clip}_i^{\mathrm{rev}}\right) = 1$ for all $\mathrm{clip}_i^{\mathrm{rev}} \in \mathcal{D}_k^{\mathrm{rev}}$. Therefore, the total accuracy of sequence $\mathcal{D}$, after reviewing the first k clips, is

$$\mathrm{acc}(\mathcal{D}_k) = \frac{\left|\mathcal{D}_k^{\mathrm{rev}}\right|}{|\mathcal{D}|} + \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{unrev}}\right) \times \frac{\left|\mathcal{D}_k^{\mathrm{unrev}}\right|}{|\mathcal{D}|}.$$

This method for calculating the accuracy of dataset $\mathcal{D}$ after reviewing the first k clips becomes useful for analyzing the performance of the confidence-based review. To see why, we first consider the lower bound on $\mathrm{acc}(\mathcal{D}_k)$. In the worst case, our confidence score will convey no information about the relative accuracies of the clips in $\mathcal{D}$. Without a relationship between $\mathrm{acc}(\mathrm{clip}_i)$ and $\mathrm{conf}(\mathrm{clip}_i)$, sorting based on confidence score is effectively the same as randomly selecting clips. In this way, we can compare the accuracy after labeling the first k clips via confidence-score with the accuracy that *would have been obtained if the first* k *clips were reviewed*. We denote this improvement in accuracy using confidence metric conf as the "improvement over random" and formalize it as $\mathrm{IOR}_k^{\mathrm{conf}} = \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{conf}}\right) - \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{rand}}\right)$, where $\mathcal{D}_k^{\mathrm{conf}}$ and $\mathcal{D}_k^{\mathrm{rand}}$ denote dataset $\mathcal{D}$ sorted by confidence score and randomly, respectively.

Next, we place an upper bound on IOR_k by considering the maximum accuracy that $\mathcal{D}$ could have after reviewing k clips. In the best case, the first k clips reviewed would be the k clips with the lowest accuracy. Here, since we're evaluating on $\mathcal{D}^{\mathrm{test}}$, where accuracy is known, we can calculate this. If we let $\mathcal{D}^{\mathrm{acc}}$ denote the sequence of clips sorted in ascending order by their true accuracy, then the maximum accuracy of $\mathcal{D}$ after reviewing k clips is $\mathrm{acc}\left(\mathcal{D}_k^{\mathrm{acc}}\right)$. Then, similar to the analysis above, we calculate the improvement of optimal review (i.e., review based on true accuracy) over random review as $\mathrm{IOR}_k^{\mathrm{opt}} = \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{acc}}\right) - \mathrm{acc}\left(\mathcal{D}_k^{\mathrm{rand}}\right)$. Semantically, $\mathrm{IOR}_k^{\mathrm{opt}}$ expresses how much higher the accuracy of the test set it after reviewing k clips in the optimal order than it would be if clips had been reviewed randomly.

We can then derive a series of global measures for the confidence-based review. While IOR_k is defined for a single number of clips reviewed, k, we look to generate a measure that expresses IOR_k across a range of k values. To do so, we calculate the average improvement over random across the number of clips reviewed, from 0 to the total number, n, as follows:

$$\overline{\mathrm{IOR}}_n^{\mathrm{method}} = \frac{1}{n}\sum_{k=0}^{n} \mathrm{IOR}_k^{\mathrm{method}}.$$

$\overline{\mathrm{IOR}}_n^{\mathrm{method}}$ expresses the mean improvement over random of method method over n clips. After calculating $\overline{\mathrm{IOR}}_n^{\mathrm{conf}}$ and $\overline{\mathrm{IOR}}_n^{\mathrm{opt}}$ (i.e., $\overline{IOR}_n$ for confidence-based and optimal sorting), we can generate a final measure for the review efficiency by expressing the average improvement of confidence score conf over random relative to the maximum possible improvement over random (optimal review):

$$\text{review\_efficiency}_n^{\text{conf}} = \frac{\overline{\text{IOR}}_n^{\text{conf}}}{\overline{\text{IOR}}_n^{\text{opt}}}.$$

This metric expresses how close review using metric conf is to optimal. If sort order based on conf exactly matches that of sorting by accuracy, $\text{review\_efficiency}_n^{\text{conf}} = 1$. If the sort order is no better than random, $\text{review\_efficiency}_n^{\text{conf}} = 0$.

Implementation details and code availability. We implement the toolbox in MATLAB version 2020b. The GUI for annotation and confidence-based review is included in the toolbox as a MATLAB application. Figures are produced using Prism9 and OmniGraffle. The entire toolbox, along with example scripts, documentation, and additional implementation details is hosted via a public GitHub repository at: https://github.com/carlwharris/DeepAction. We provide the intermediary data generated for the home-cage dataset (e.g., spatial and temporal frames and features, annotations, etc.) as an example project linked in the GitHub repository. Data produced to generate results for the CRIM13 project is available upon request, but not provided as an example project due to its large file sizes. Full data (i.e., results for each test split in both projects) needed to replicate the results is also available on request. Data for the exploratory data set is proprietary.

Data availability

Supplementary information is available for this paper. Correspondence and requests for materials should be addressed to the corresponding author.

Received: 3 August 2022; Accepted: 7 February 2023
Published online: 15 February 2023

References

1. Crabbe, J. C., Wahlsten, D. & Dudek, B. C. Genetics of mouse behavior: Interactions with laboratory environment. *Science* **284**, 1670–1672 (1999).
2. Wahlsten, D. *et al.* Different data from different labs: Lessons from studies of gene–environment interaction. *J. Neurobiol.* **54**, 283–311 (2003).
3. Würbel, H. Behavioral phenotyping enhanced–beyond (environmental) standardization. *Genes Brain Behav.* **1**, 3–8 (2002).
4. van Dam, E. A. *et al.* An automated system for the recognition of various specific rat behaviours. *J. Neurosci. Methods* **218**, 214–224 (2013).
5. Drai, D., Kafkafi, N., Benjamini, Y., Elmer, G. & Golani, I. Rats and mice share common ethologically relevant parameters of exploratory behavior. *Behav. Brain Res.* **125**, 133–140 (2001).
6. Burgos-Artizzu, X. P., Dollár, P., Lin, D., Anderson, D. J. & Perona, P. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 1322–1329 (IEEE).
7. Jhuang, H. *et al.* Automated home-cage behavioural phenotyping of mice. *Nat. Commun.* **1**, 1–10 (2010).
8. Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S. & Branson, K. JAABA: Interactive machine learning for automatic annotation of animal behavior. *Nat. Methods* **10**, 64–67 (2013).
9. Lorbach, M., Poppe, R., Dam, E. A. V., Noldus, L. P. & Veltkamp, R. C. in *International Conference on Image Analysis and Processing*. 565–574 (Springer).
10. Lorbach, M. *et al.* Learning to recognize rat social behavior: Novel dataset and cross-dataset application. *J. Neurosci. Methods* **300**, 166–172 (2018).
11. Bohnslav, J. P. *et al.* DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. *Elife* **10**, e63377 (2021).
12. Zhu, Y., Lan, Z., Newsam, S. & Hauptmann, A. in *Asian conference on computer vision*. 363–378 (Springer).
13. Piergiovanni, A. & Ryoo, M. in *International Conference on Machine learning*. 5152–5161 (PMLR).
14. Feichtenhofer, C., Pinz, A. & Zisserman, A. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1933–1941.
15. Ma, C.-Y., Chen, M.-H., Kira, Z. & AlRegib, G. TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *Signal Process. Image Commun.* **71**, 76–87 (2019).
16. Wang, L. *et al.* Temporal segment networks: Towards good practices for deep action recognition. in *European conference on computer vision*. 20–36 (Springer).
17. Kramida, G. *et al.* in *Proc. Vis. Observ. Anal. Vertebrate Insect Behav. Workshop (VAIB)*. 1–3.
18. Simonyan, K. & Zisserman, A. Two-stream convolutional networks for action recognition in videos. *Adv. Neural Inf. Process. Syst.* **27** (2014).
19. Zach, C., Pock, T. & Bischof, H. in *Joint pattern recognition symposium*. 214–223 (Springer).
20. Eroglu, Y., Yildirim, K., Çinar, A. & Yildirim, M. Diagnosis and grading of vesicoureteral reflux on voiding cystourethrography images in children using a deep hybrid model. *Comput. Methods Programs Biomed.* **210**, 106369 (2021).
21. Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V. & Herrera, F. A unifying view on dataset shift in classification. *Pattern Recogn.* **45**, 521–530 (2012).
22. Quinonero-Candela, J., Sugiyama, M., Schwaighofer, A. & Lawrence, N. D. *Dataset shift in machine learning*. (Mit Press, 2008).
23. Le, V. A. & Murari, K. Recurrent 3D convolutional network for rodent behavior recognition. In *ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1174–1178 (2019).
24. Jiang, Z. *et al.* Context-aware mouse behavior recognition using hidden markov models. *IEEE Trans. Image Process.* **28**, 1133–1148 (2018).
25. Eyjolfsdottir, E. *et al.* Learning animal social behavior from trajectory features. Hosted by the School of Informatics at the University of Edinburgh (Scotland). https://homepages.inf.ed.ac.uk/rbf/VAIB12PAPERS/eyjolfsdottir.pdf (2012).
26. Zhang, S. *et al.* Action recognition based on overcomplete independent components analysis. *Inf. Sci.* **281**, 635–647 (2014).
27. Meng, Q., Zhu, H., Zhang, W., Piao, X. & Zhang, A. Action recognition using form and motion modalities. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **16**, 1–16 (2020).
28. Chen, W. Human and Animal Behavior Understanding. *Graduate Theses, Dissertations, and Problem Reports, West Virginia University* (2014). https://doi.org/10.33915/etd.192
29. Farnebäck, G. in *Scandinavian conference on Image analysis*. 363–370 (Springer).

30. Gianluigi, C. & Raimondo, S. An innovative algorithm for key frame extraction in video summarization. *J. Real Time Image Proc.* **1**, 69–88 (2006).
31. Wu, J., Zhong, S.-H., Jiang, J. & Yang, Y. A novel clustering method for static video summarization. *Multimed. Tools Appl.* **76**, 9625–9641 (2017).
32. Batty, E. *et al.* BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos. *Adv. Neural Inf. Process. Syst.* (2019).
33. Papernot, N. & McDaniel, P. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint* arXiv:1803.04765 *(2018).*
34. Gal, Y. & Ghahramani, Z. in *international conference on machine learning.* 1050–1059 (PMLR).
35. Cryan, J. F. & Holmes, A. The ascent of mouse: Advances in modelling human depression and anxiety. *Nat. Rev. Drug Discov.* **4**, 775–790 (2005).
36. Dollár, P. (software reference): "Piotr's Computer Vision Matlab Toolbox (PMT)" by Piotr Dollar in 2016. Available at: https://github.com/pdollar/toolbox (2014).
37. Pérez-González, A., Jaramillo-Duque, Á. & Cano-Quintero, J. B. Automatic boundary extraction for photovoltaic plants using the deep learning U-net model. *Appl. Sci.* **11**, 6524 (2021).
38. Cun, S. Dual TVL1 Optical Flow. (software reference): "Dual TV-L1 Optical Flow" by Xiaodong Cun in 2017. Available at: https://github.com/vinthony/Dual_TVL1_Optical_Flow (2017).
39. Deng, J. *et al.* in *2009 IEEE conference on computer vision and pattern recognition.* 248–255 (IEEE).
40. Hussain, M., Bird, J. J. & Faria, D. R. in *UK Workshop on computational Intelligence.* 191–202 (Springer).
41. Duda, R. O., Hart, P. E. & Stork, D. G. Pattern classification 2nd edition. *New York, USA: John Wiley&Sons*, 35 (2001).
42. Murphy, K. P. *Machine learning: a probabilistic perspective*. (MIT press, 2012).
43. Le, Q., Karpenko, A., Ngiam, J. & Ng, A. ICA with reconstruction cost for efficient overcomplete feature learning. *Adv. Neural Inf. Process. Syst.* (2011).
44. Nocedal, J. & Wright, S. J. *Numerical optimization* (Springer, 1999).
45. Merity, S., Keskar, N. S. & Socher, R. Regularizing and optimizing LSTM language models. *arXiv preprint* arXiv:1708.02182 *(2017).*
46. Dwarampudi, M. & Reddy, N. Effects of padding on LSTMs and CNNs. *arXiv preprint* arXiv:1903.07288 *(2019).*
47. Graves, A. in *Supervised sequence labelling with recurrent neural networks* 5–13 (Springer, 2012).
48. Graves, A., Mohamed, A.-r. & Hinton, G. in *2013 IEEE international conference on acoustics, speech and signal processing.* 6645–6649 (IEEE).
49. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
50. Ogawa, A. & Hori, T. Error detection and accuracy estimation in automatic speech recognition using deep bidirectional recurrent neural networks. *Speech Commun.* **89**, 70–83 (2017).
51. Beaufays, F., Sak, H. & Senior, A. in *Interspeech.* 338–342.
52. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
53. Bishop, C. M. & Nasrabadi, N. M. *Pattern Recognition and Machine Learning* Vol. 4 (Springer, 2006).
54. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M. & Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint* arXiv:1609.04836 *(2016).*
55. Yildirim, M. & Çinar, A. A new model for classification of human movements on videos using convolutional neural networks: MA-Net. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.* **9**, 651–659 (2021).
56. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. in *International Conference on Machine Learning.* 1321–1330 (PMLR).
57. Kull, M. *et al.* Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Adv. Neural Inf. Process. Syst.* (2019).

Acknowledgements

Supported by National Science Foundation Award #1632738 (PUT), the Neukom Institute for Computational Science at Dartmouth College (Neukom Scholars award to CH and Neukom Post-doctoral Fellowship to KF), and the David C. Hodgson Endowment for Undergraduate Research Award (to CH). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Author contributions

C.H. conceived of the project, developed the approach and software, and analyzed the results with input from K.F. C.H. wrote the manuscript with input from P.T. and K.F. M.K. and M.M. procured the exploratory dataset.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-29574-0.

Correspondence and requests for materials should be addressed to P.U.T.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.