A Survey of Deep CNNs for Mouse Paw Localization

Molly I. Lucas1,2 & Konstantin Kapagoves1

Motivation

To better understand the neural basis of motion, neuroscience records neural activity in the brain while mice perform an action. In order to link movement behavior with brain activity, mouse bout placement is manually labeled by experimenters in each image frame. This is an incredibly time-consuming and also involves human error (difference is how each individual label is a foot using an X,Y coordinate). A recent Nature Neuroscience paper used convolutional neural network (CNN) to automate this task. DeepLabCut has a high accuracy rate but takes excessive computational resources and time. Hence, we both explored ways to improve this new field standard (DeepLabCut) and also test two variants of an object detection algorithm, Faster R-CNN, to see if we can improve performance (realistic high accuracy while improving speed) of labeling mouse feet in this type of image.

Data & Preprocessing

- Gray-scale images were collected using a high-resolution (800x800 pixels) camera under infrared light illumination at 100 frames per second. Each frame constitutes a single image.
- Mouse position is fairly constant on the ball (left sagittal view). Example image (left foot marked in pink) seen on right. Left foot X,Y coordinates were used as the label in each image (hand labeled by experts in Jing Sun lab at Stanford University).
- Data was not pre-processed or augmented (low class results did not indicate more training data was needed).
- Training set: 973 images (80% of data), Development set: 315 images (20% of data), Test set: 315 images (20% of data). The same training, development, and test images were used in each independently tested model. Development set was used to test variations within each model. Test set was used only once as a final comparison across the three models.

Models

We tested four models: DeepLabCut, DeepLabCut with RoNetC102, Faster R-CNN, and Faster R-CNN combined with RoNetC102.

DeepLabCut

DeepLabCut is the current best field standard for markerless labeling of experimental subjects’ joints. DeepLabCut (DLC) uses the feature detector portion of the best human pose estimation model(s). It uses a RoNet architecture pre-trained on the imagenet dataset, by using desalionalized layers, the final layers are modified to output a feature map that matches the input size. This feature map is then passed through a regression layer to refine the x-y position and finally a sigmoid activation layer. Cross entropy loss is combined with Huber loss (for regression layer) and the network is trained with SGD.

Faster R-CNN

- Faster R-CNN is an object detection network that increases the speed of Fast R-CNN by setting a novel Region Proposal Network (RPN). This RPN is a fully connected deep network combining bounding box prediction with object scores, which eliminates the massive computational bottleneck of Fast R-CNN. The output layer of Faster R-CNN classifies proposed regions to match input layers of Interact. Loss function uses binary classification (positive/negative) evaluated by measuring intersection over union of proposed bounding boxes and the true target bounding box. It is trained using SGD.

Faster R-CNN with RoNetC102

- Here, we exchanged the convolutional layers of Faster R-CNN with RoNetC102. The same Regional Proposal Network was used to estimate bounding box locations, and the same classifier layer was used as the model output.

Error Analysis

- DeepLabCut showed both low bias and low variance. Human error for this task can be approximated by human ability, which gives a 0% error rate (with some individual variability in X,Y coordinate label placement). Therefore, even the low error rate seen initially has room for improvement. To improve bias, we used a deeper CNN (replacing the RoNetC102 component in DeepLabCut with a RoNetC102).
- Based on the difference between train and CV RMSE, the larger network (DeepLabCut, RoNetC102) -- especially the version with intermediate supervision -- is overfitting the training data so increasing the amount of training data may help overcome this problem and further decrease the CV RMSE.

Results

- A main difference between DeepLabCut and Faster R-CNN is that Faster R-CNN uses bounding boxes as labels, whereas DeepLabCut uses X,Y coordinates. We initially used a small box to resemble an X,Y point. This resulted in 95% accuracy even on training (faster R-CNN). After manually adding 300 incorrectly labeled images, we found that the model was identifying light/dark edges, likely due to the X,Y label being placed at the edge of the foot. To fix this, we made the input label bounding box larger and centered it around the foot. We tested 5 variants of this before finding the input label with the highest accuracy on training (RMSE of 80 pixels). This was used in subsequent testing for Faster R-CNN.
- Using Faster R-CNN, variance was quite low (minimal difference between training and development error), but bias was higher when using DeepLabCut (higher error both in training and development). Therefore, we tried changing the Faster R-CNN architecture to improve DeepLabCut, hoping this hybrid could improve the bias while still maintaining the speed./ low computational requirements of Faster R-CNN.

Experiment 1: Improving DeepLabCut (RoNetC102)

- A: While DeepLabCut is a field standard, it is very new (2018). Here, we tested learning rate and model architecture to try to improve accuracy.
- The scaling factor of 0.8 was the best, but there was surprisingly no major difference from 0.1 to 1.2 scaling -- indicating that at our resolution, detecting a simple white paw on a black background is fairly high resolution.
- The deeper RoNet architectures overfit the training data, adding more training data could overcome this problem.

Experiment 2: DeepLabCut with RoNetC102

- A: To see whether a deeper network would improve accuracy.
- The deeper RoNet architectures overfit the training data, adding more training data could overcome this problem.

Experiment 3: Comparing Faster R-CNN to DeepLabCut

- A: Since DeepLabCut is the only algorithm used for this task, we tested whether another object detection model (Faster R-CNN) performed similarly to DeepLabCut (RMSE as output metric).
- Learning rate 0.005 showed the lowest development error. The model had greater bias (assuming bias error is close to zero) but fairly low variance. This suggests either the model architecture is not optimal or else more training data could be useful.

Experiment 4: Creating a hybrid between DeepLabCut and Faster R-CNN

- A: Since Faster R-CNN trained and ran on a faster than DeepLabCut but had a lower accuracy on both training and development sets. To try to improve accuracy while maintaining superior speed, we integrated RoNetC102 (as used in DeepLabCut) with Faster R-CNN to create a hybrid between the two. The goal was to use accuracy as a satisfying measure (RMSE < 10 pixels) and use testing speed as our limiting measure.

Final Comparison Across Four Models

- The best model performs at 2.9460 pixels RMSE. We argue that this performance is more than satisfactory, as human variability in this task is reported to have an RMSE of 3.7 pixels (1). Further, a mouse’s paw takes up an area of 1000-2000 pixels at this resolution. DLC and DLC-102 were the only models to achieve a satisfying metric (RMSE < 12 pixels).
- Faster R-CNN had a much faster train and run time, even using a smaller GPU, than DeepLabCut, however the RMSE was much higher. Faster R-CNN with RoNetC102 cut the RMSE in half while maintaining relatively low computing time.

Conclusions & Future Directions

While DeepLabCut still held the greatest accuracy (lowest RMSE) of all of the models tested, we were able to achieve fairly high accuracy using a modified version of Faster R-CNN (using RoNetC102) instead of the usual convolutional layers. This new model was significantly faster both at training and testing new data compared to DeepLabCut, even while using a much smaller GPU on workstation compute. We believe further modifying this new algorithm (adding more layers, training data, and training time) could ultimately lead to similar performance as DeepLabCut while requiring far less computational resources. This would be a significant improvement for the field.