Predicting How a Dog Would Act From Ego-Centric Images

Thomas Nguyen
Department of Computer Science
University of Washington
Seattle, WA 98195
tomn@cs.washington.edu

Abstract

I use the DECADE [1] dataset to evaluate different models where images from the perspective of a dog are input, and predictions of actions of the dog immediately following those images are output and compared with the ground truth as a form of self-supervision. In the original paper [1] accompanying this dataset, an encoder-decoder style network is used to predict the dog's actions where a stream of images are fed in and the decoder outputs predicted actions for different body parts of the dog. I evaluate the effect of different models for feature extraction on the performance of predicting future limb movements in the encoder-decoder network.

1 Introduction

1.1 DECADE dataset

The DECADE dataset consists of 24500 frames from a camera mounted on a dog's head. These frames come from multiple different video sessions from the dog's perspective such as going for a walk or playing fetch. Along with each frame are annotations of the orientation of the dog's body, four legs, and tail in quaternions. Inertial measurement units (IMUs) are used to record the absolute orientation of the dog's body and limbs, that is to say with respect to magnetic north and gravity. The frames are sampled at 5 frames per second. For evaluating movements, there are eight action classes based on the IMU readings. The difference between quaternions in consecutive frames are defined as actions representing the way a limb moves. These actions are binned into eight action classes through k-means. The encoder-decoder network predicts action classes for limbs in future time-steps.
The dataset uses IMUs that constantly recalibrate and can occasionally lose calibration so there is occasional noise in the dataset by drifting of limbs.

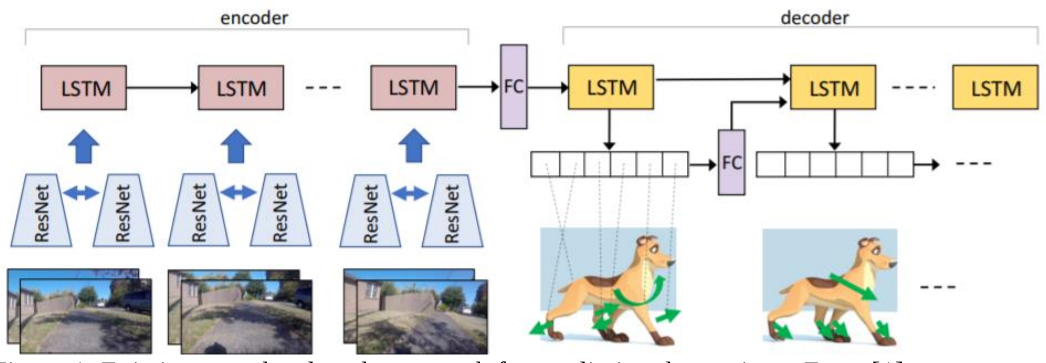

Figure 1. Existing encoder-decoder network for predicting dog actions. From [1]

1.2 Existing prediction model

See Figure 1 extracted from [1] for the original existing prediction model; the model consists of two ResNet-18 towers with shared weights used to create features for image pairs in a stream that are fed into an LSTM, followed by a fully connected layer, and lastly another LSTM to generate the output probability of actions at multiple periods in time. Each ResNet tower takes in one image of an image pair and outputs the feature representation layer which is the output of the last convolutional layer before the final fully connected layer at the end of the ResNet. These twin tower feature representations are concatenated and fed into the encoder LSTM. For all experiments in this paper, the input consists of 5 timesteps, where the LSTM first takes in frames t and $t+1$ as timestep 1, then $t+1$ and $t+2$ as timestep 2 and so forth. The output is for each of the 5 timesteps following the input, predictions of action classes for each of the joints. The ResNet towers were pre-trained on ImageNet [2] and fine-tuned by predicting joint movements between pairs of frames.

1.3 Representation learning

In machine learning, it is possible to use the information gained from training one model in another separate similar task. An example is if you wanted to classify images of birds by their species, you could train an arbitrary model such as ResNet-18 from scratch with random weight initialization. You could also take a ResNet-18 model that has already been trained on ImageNet, replace the last fully connected classification layer with a randomly initialized new fully connected layer mapping your species encoded as a one-hot vector, and re-train the network. The second scenario where you use a pre-trained model should train faster and have good accuracy. This is useful if you don't have many images in the new task you want to learn. The idea is that the model trained on ImageNet already extracts good features for classification, and you can fine-tune the layers to figure out how to adjust and map those features into your bird species. In the worst case scenario where the pre-trained model is bad, it could still be considered a random initialization of the network for the new task and the model will learn regardless.
We can use the hidden representation layer for other tasks as well such as object detection, walkable surface estimation, and predicting limb actions. Currently, many people use models pre-trained on ImageNet as a baseline. ImageNet was built upon a lot of manual labor where humans had to go through and label images one by one. There has been a lot of work to learn visual representations without manual labels [3]. Using dogs as a supervisor is one potential method.
[3] Argues that "the next step in visual learning requires signal/supervision from physical exploration of our world". Intelligent beings interact with the world around them- touching, smelling, hearing, etc. - and build a stronger visual representation with every interaction yet we expect our models to learn a good representation through strictly seeing millions of images. ImageNet currently is a large bank of static images where models cannot effectively learn interactively.
[4] Relates visual learning and ego-motion with the famous "kitten carousel" [5] experiment. Two newborn kittens were isolated to darkness and occasionally brought out onto a carousel where one "active" kitten could move freely and the other "passive" kitten was not allowed to move on its own but rather carried along in a basket mirroring the freely moving active kitten. "The active kitten simultaneously experienced signals about his own motor actions, the passive kitten did not". While both kittens were exposed to the same visual information, the passive kitten "suffered fundamental problems". They "contend that today's visual recognition algorithms are crippled much like the passive kitten". ImageNet is like the passive kitten, and using intelligent beings such as dogs as a supervisory signal for interaction with the world can help strengthen learned representations and improve performance in other tasks.

2 New feature extraction

2.1 Model architectures

Residual networks, sometimes referred to as ResNets, are typically deep networks that alleviate degradation issues with skip connections. With regular convolutional neural networks, results typically get worse as more layers are added [6]. Skip connections add an identity function from one layer to a later one; bad or not useful nodes can arbitrarily be replaced by the identity mapping skipping over it and deep network connections can be represented with shorter ones if nodes don't help with accuracy. We can train very deep networks and degradation is alleviated with skip connections. ResNets can also help with vanishing or exploding gradient although that problem was previously largely addressed with many different normalization techniques [6].

Inception networks use inception modules. Inception modules use a variety of filters and concatenate their output [7]. This has many performance gains such as being parallelizable, more feature recognition, and that many smaller stacked convolutions may be more efficient than less larger convolutions. Inception modules using a variety of filters in different paths and concatenating their output means each different path can be run at the same time. Also with a variety of filters, more image features can be captured since typically important image features in an image vary in size from image to image. Lastly replacing a 5x5 convolution with two 3x3 convolutions may actually be faster computationally since a 5x5 sums 25 points and two 3x3's sum 18 points. It turns out these stacked smaller convolutions can perform as well as larger convolutions.

2.2 InceptionV3 stream extraction

In the network of Figure 1, I swap the ResNet-18 towers for InceptionV3 towers with shared weights pretrained on ImageNet. The representation layer has 4096 features with InceptionV3. One difference is that the InceptionV3 features are processed once and the network is not fine-tuned end-to-end. I could not find code from the original paper for training end-to-end and extracting features, rather a bag of preprocessed features for different timesteps was provided. The code I wrote for extracting features from InceptionV3, testing, and analysis can be found at https://gitlab.cs.washington.edu/tomn/testdogtorch. This code is forked from https://github.com/ehsanik/dogTorch, https://github.com/eriklindernoren/PyTorch-YOLOv3, and https://github.com/Cadene/pretrained-models.pytorch. The code allows swift and easy loading of many different models pretrained on ImageNet.

2.3 Training

Aside from freezing the feature extraction towers, and the number of features being generated by the network, the rest of the encoder-decoder portion is kept consistent throughout this paper, namely the size of the hidden layers in the LSTM is 512, the number of input timesteps is 5, and 5 outputs are predicted. I trained for 50 epochs each, all networks converged on the validation set well before 50 epochs. I only tested on ResNet-18 and InceptionV3. I retrained the encoder-decoder for ResNet-18 features without fine-tuning the ResNet portion to be consistent with training the encoder-decoder from scratch for the Inception Network, and so these results could compared the existing prediction models that were trained end-to-end.

3 Results

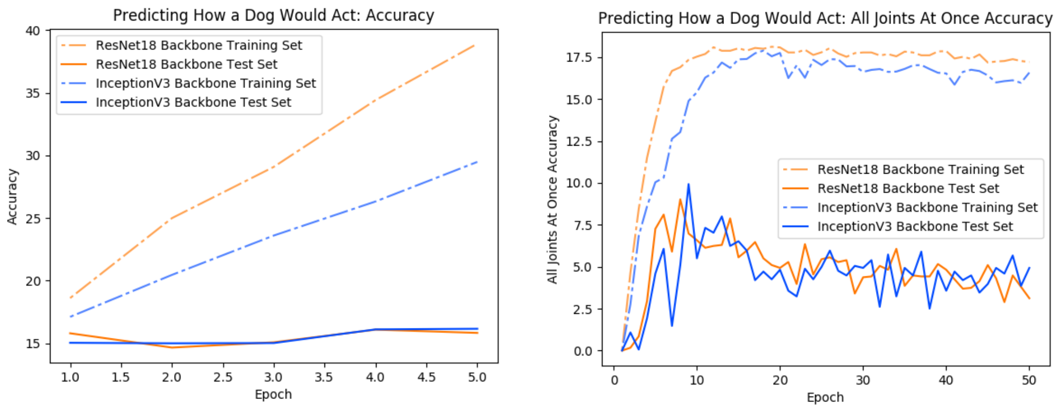

Figure 2. Accuracy of limb action predictions

3.1 Multiple image feature extraction

Figure 2 shows the accuracy of predictions from feeding different network features into the encoder-decoder and training. Accuracy is defined as correctly predicting the action class a limb took for that unseen timestep. All joints at once accuracy is correctly predicting all (8) action classes for all 6 body parts for a certain timestep. It appears as though without fine-tuning the network end-to-end, both InceptionV3 and ResNet-18 features perform equally for both metrics on the validation and test set. One benefit of InceptionV3 was that despite having the same about the test set accuracy as ResNet-18 for both metrics throughout the epochs, InceptionV3's training set accuracy diverged from the test set more slowly. Otherwise, the peak all joints at once accuracy for both networks was similar at 8.00% for Inception and 7.75% for ResNet-18. On a per limb basis, the accuracy was 15.76% for InceptionV3, and 15.65% for ResNet-18. These are not large enough margins for a significant conclusion. In the future more trials and training end-to-end would produce better results.

Comparing these results to the original paper's existing prediction model where they trained end-to-end, they achieved an all joints at once test accuracy of 9.49% and a per limb accuracy of 21.62% with ResNet-18. They also had a baseline convolutional neural network trained end-to-end that achieved an all joints at once test accuracy of 8.67%, and a per limb accuracy of 19.84.

4. Results

Through swapping the features input to the encoder-decoder network for prediction of dog actions, I have found that there is no noticeable improvement in test set performance when training the encoder-decoder network between ResNet-18 and InceptionV3. Both networks produced similar results on the test set and actually had similar performance on the training set at every epoch. Future steps would include retraining end-to-end on both networks to see if better features could be extracted in one. Future steps would also include testing on other tasks such as walkable path estimation and ImageNet classification accuracy both before and after end-to-end training to see how the representation improves for some tasks and gets worse in others as the networks are fine-tuned on the DECADE dataset. [1] Shows that improvement after fine-tuning on the DECADE dataset, training end-to-end produced better results in features for walkable surface estimation with a ResNet-18 backbone.

References

[1] K. Ehsani, et al. Who Let The Dogs Out? Modeling Dog Behavior From Visual Data. In CVPR, 2018.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge.

International Journal of Computer Vision (IJCV), 2015.

[3] L. Pinto, D. Gandhi, Y. Han, Y. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In ECCV, 2016.

[4] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In ICCV, 2015

[5] R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. Journal of comparative and physiological psychology, 1963

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[7] C. Szegedy *et al*., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 1-9.

[8] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. Technical report, 2018.