PawPal: Dog's Best Friend

Aman Kumar Jha (amankj) Eric Hofesmann (erichof) Preet Gill (preetsg) Vihang Agarwal (vihang)

I. Executive Overview

In today's fast-paced world, many pet lovers avoid owning pets due to their inability to tend to them constantly. It is almost impossible to monitor the pet's activities all day long and prevent them from destroying household items, or even worse, hurt themselves. Puppies specifically need constant feedback for proper training such as telling them not to bite on stuff or not to go to certain area. Some of the popular alternatives such as having a dog-sitter every day or crating for long hours is either financially infeasible for most households or comes at negative physiological cost for pets. Even with 68% of house holds in U.S owning pets, there are no existing solutions in the market which allow autonomous active monitoring and safekeeping of pets without requiring any actions or interference from the owner.

As the worlds first virtual dog sitter, PawPal[1] tracks the activities of your dog and autonomously reinforces their behavior in real time. Our system requires just a camera which detects your dog and the objects in your home in real time by capturing video sequences. The system learns to recognize its activities and interactions using object detection and activity classification deep networks. By classifying the activity as good or bad, the system provides audio based reinforcement to the dog in the owners voice. PawPal strives to create a safe and loving home for both you and your dog. Now you can leave your dog at home with complete peace of mind!

II. Background and Impact

No matter how much one loves their dog, it is impossible to guarantee their safety and that of their surroundings unless one spends every moment with them. In 2017 alone, a total of 89.7 million dogs lived as pets in US households. $5.41 billion was spent on dog sitting in the US in 2015 alone. However even with huge demand, current technological alternatives in this growing market do not actively or autonomously interact with your pet and require owners' supervision and hence time. Alternatives such as crating dogs is becoming increasingly unpopular as studies have shown negative physiological effects of canine crating. Our solution : PawPal not only makes the house a nurturing place for pet, it also ensures pets safety and most importantly protects your house from pet destruction while cultivating good behavior.

III. Method

We have implemented a pipeline which integrates object detection and activity recognition deep networks.

[1]https://github.com/ehofesmann/PawPal

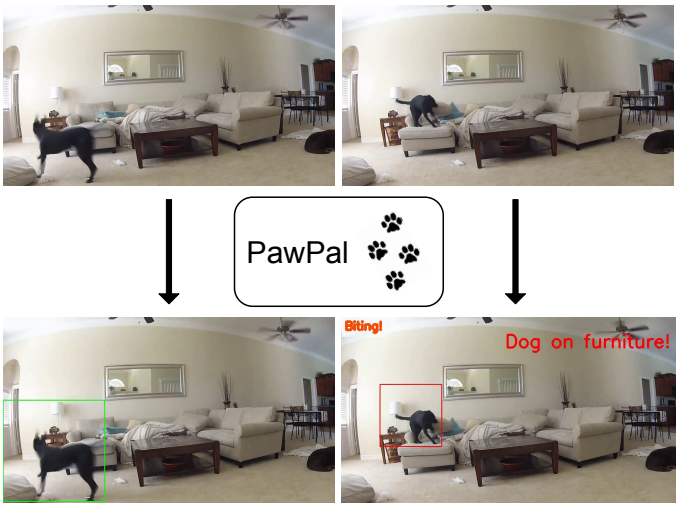

Fig. 1: PawPal is a system which takes surveillance footage of dogs alone at home and detects when they are misbehaving, for example jumping on furniture or biting household objects. In the left image, the dog is detected on the ground and while not biting. The right image automatically detected the dog on furniture and biting the couch cushion.

We implemented YOLO [8] (You Only Look Once) to obtain bounding boxes for pet-specific classes such as dogs and cats, and static objects like couch, pillow, sofa, tables etc. This allows to track objects and define relationships. Our prototype was aimed to carry out dog behavior detection in real time making processing speed a top priority. After considering all state of the art detection algorithms - YOLO, SSD500 [6], SSD300 and RetinaNet [4], YOLO made sense for a balance between real time implementation and detection accuracy. The major concept of YOLO is to build a CNN network to predict a (7, 7, 30) tensor. It uses a CNN network to reduce the spatial dimension to 77 with 1024 output channels at each feature location. YOLO performs a linear regression using two fully connected layers to make 772 boundary box predictions. To make a final prediction, those with high box confidence scores (greater than 0.25) are kept as the final predictions.

We define and maintain visual relationships between the animate and inanimate classes. We use information of bounding box coordinates of dog and furniture classes in the video feed, to keep track of the dog's movements and it's relative position with respect to various furniture in the frame.

The activity classification algorithm we used was C3D [9] which is a 3D CNN architecture for video classification and attempts at learning spatio-temporal features. We processed

video sequences in buffers of 16 frames each. Activities were recognized through these visual features on a discriminative video classification task. An ensemble of visual relationships and activity recognition is used to predict if the dog is on a piece of furniture or if it is biting objects.

IV. Prototype

A. Pipeline

The entire pipeline is shown schematically in Fig. 2. Our activity recognition network, C3D, requires a minimum of 16 frames as input. Hence, we process the incoming video feed on two levels: per frame, and on a stack of 16 frames which we store in a buffer.

Every frame is passed as an input to YOLO for object detection. We used the YOLO implementation in Tensorflow [1], based on the darknet [7] framework and with pre-trained weights on COCO [5] dataset. YOLO returns a dictionary of classes of detected objects with corresponding bounding box coordinates. A linear search across all detected classes allows us to access the bounding box coordinates of all dogs and furniture detected in the frame. In the prototype, we are focusing on tracking a single dog at a time. Hence, in frames where multiple dogs are detected, we store the bounding box coordinates corresponding to the maximum confidence detection. We also create a list of bounding box coordinates of all objects detected under classes 'sofa','couch','table','diningtable' and 'chair'.

In order to detect if the dog is partially or completely on top of a furniture, we define a 'red zone' for each piece of furniture detected in the frame, as shown in the Fig. 3. If the region defined by the bounding box coordinates of the dog falls completely within the red zone of any detected furniture, we classify this frame as 'dog on couch' situation. We have fine-tuned the parameters defining this red zone to account for most practical scenarios, including situations such as dogs overhanging from couches or in the process of climbing on a sofa, for example. If no dog is detected in a given frame, we use the bounding box information from the most recent detection in a previous frame to make a decision regarding the dog's location. This part of the pipeline allows for a frame-by-frame tracking of the dog's location and determining if it is on top of any furniture. We ran our implementation of YOLO on a CPU for which we achieved a processing time of 0.75 seconds for each frame. However, processing times as low as 0.02 seconds are achievable using GPUs [8].

Once the buffer of 16 frames is full, we proceed to create the input data set for C3D. C3D was trained using Tensorflow to perform binary classification between "dog biting" and "dog not biting" scenarios. As mentioned before, C3D takes a stack of 16 frames of 112 x 112 sized images as input to perform activity classification. To account for missing dog detections in a given 16-frame buffer, we use linear interpolation to obtain bounding boxes for frames in which no dog is detected. The frames are cropped to the region defined by dog bounding box to extract the image of the dog. The 16 cropped dog images are then re-sized into square images measuring 112x112 pixels. This set of images is passed onto C3D and the buffer is cleared. For cases where no dog is detected in a majority of frames, the buffer is cleared immediately for the next set of 16 frames. The time lag between two consecutive outputs of C3D is determined by either the frame rate of incoming video or the YOLO processing time. The time lag between two consecutive frames of a typical 30 frames-per-second video is 0.034 seconds. Depending on the implementation of YOLO, the processing times range from 0.02 seconds on GPU to 0.75 seconds on CPU. In case of a frame-rate limited implementation, the buffer build-up takes 0.53 seconds which is followed by 0.03 seconds for C3D processing. Hence the "biting"/"not biting" results for our pipeline are delayed by 0.56 seconds on average.

In order to train C3D for "biting"/"not biting" classification, we used the activity classification platform M-PACT [3]. We had to reformat the dataset to be in the form of TFRecords for efficient data loading and then used this data to train the default C3D model. We undertook several steps to improve the accuracy of "biting"/"not biting" classification including data augmentation and network modification. We found that cropping the image exactly to the extremities of the dog bounding box led to loss of important information regarding the 'biting' activity. Hence we expanded the cropping to a region 1.5 times bigger than the original bounding box. This step helped in improving the performance of C3D by including information such as the object being bitten and the dog's orientation with respect to its surroundings. Next, we Froze all layers except the last two 3D Conv and fully connected layers. We also increased the dropout rate in the FC layers to avoid overfitting on our limited dataset. Incorporating these changes led to an improvement in classification accuracy of roughly 5% on Split 1.

The next section describes the details of dataset which was used to train C3D.

B. Dataset

As mentioned before, YOLO was used with pre-trained weights on COCO dataset. We designed our own dataset to train C3D for activity classification. We downloaded a total of 200 videos from YouTube, comprised of 80 videos with clips of dogs biting, and 120 videos with clips of dogs doing non-biting activities, using the youtube-dl package for Python. We then used ffmpeg to crop each of these videos into 4 second long videos such that they contained purely biting or non-biting activities. We then extracted 16-frame clips from each of these unique four second videos to create the training dataset for C3D. In total, our dataset comprised of 337 16-frame long 'biting' clips and 525 16-frame long 'non-biting' clips. Some of these clips are depicted in Fig. 4. While collecting data, we tried our best to include a wide range of dog breeds and only include videos that were shot indoors, to prevent the network from classifying based on environment instead of activity. The 'non-biting' dataset also included a variety of 'non-biting' activities including walking, running, sleeping, playing etc.

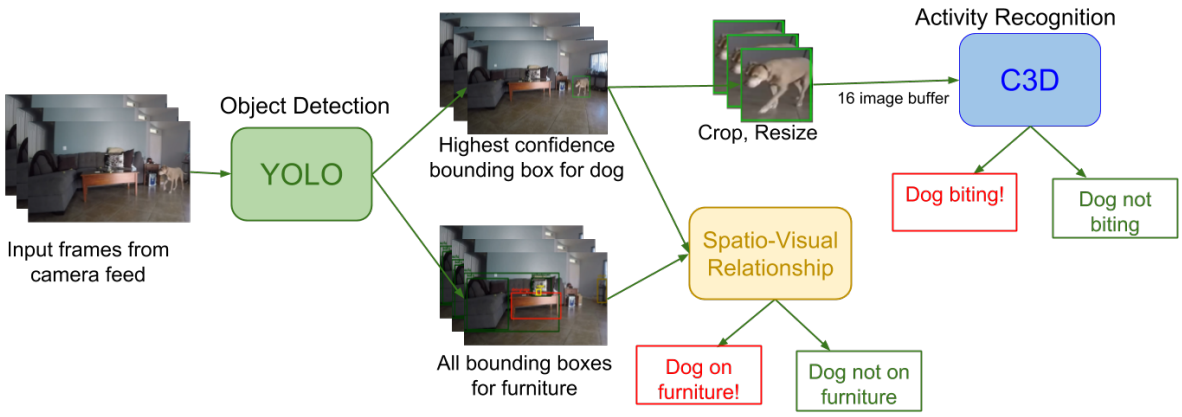

Fig. 2: A diagram of the PawPal pipeline is shown. Video frames are passed through the object detector YOLO which locates dogs and furniture in the frame. The bounding box detections are then passed through our spatio-visual relationship algorithm to determine if the frame contains a dog on top of furniture. The detections are also stored in a 16 frame buffer and passed through C3D which classifies if the clip contains a dog biting an object or not.

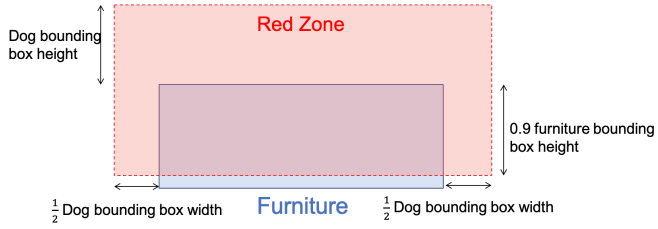

Fig. 3: A depiction of the process used to determine if a dog is on furniture is shown where the blue rectangle represents the furniture bounding box. A dog is classified as being on furniture if its bounding box is located within the "Red Zone".

V. Results

A. Dog-on-furniture Detection

Our spatio-visual relationship module may be simple but is extremely effective. As seen in Figs. 1 and 5, PawPal is able to detect when the dog has jumped onto furniture. We have not developed a metric to calculate the quantitative performance of PawPal detecting when a dog is on furniture due to a lack of ground truth temporally localized annotations of videos in which a dog is verified to have jumped onto furniture. However, the qualitative results for this task show promising results where PawPal is accurately able to detect when a dog is on furniture.

While PawPal is often able to detect a dog on furniture as expected, the procedure behind this detection could lead to possible failure cases. For example, if a video contains a couch with a non-rectangular shape, the bounding box must be rectangular and will need to encompass the entire couch along with an empty portion of the frame. If the dog is then next to the couch in this empty portion, it would be detected as on the couch. A solution to this problem could be to implement a mask detector, like Mask R-CNN [2], to segment out the couch in the frame.

C3D Recognition Accuracy (%)

Mean	Standard Deviation	Random Chance
68.41	6.10	50.00

TABLE I: Results of C3D trained and tested on our Dog Biting dataset across all 5 splits. The task is to classify if a given video is either of a dog biting an object or a dog performing any activity that is not biting.

B. Biting Recognition

Given the success of the dog-on-furniture detection task of PawPal, we decided to also incorporate the difficult task of biting recognition. C3D was trained and tested on our Dog Biting dataset. 20% of the dataset was set aside for testing and this was done across five splits to evenly train and test over all videos in the dataset. We took care to ensure that all clips from a single video were either entirely within the training set or the testing set so that we do not test on a clip similar to one that was used for training.

Figs. 1 and 6 show qualitative results of the biting recognition network. In these frames, PawPal is accurately able to predict if the dog is biting an object or not. Table I shows the quantitative results of C3D tasked with biting recognition. The overall accuracy is well above random but not perfect. There is large room for improvement to bring the mean accuracy to 100%. One improvement would be to enlarge the dataset to allow the network to train fully without requiring any frozen layers while also giving the network more examples that it can use to better generalize. Additionally if the bounding box detection can be improved to be more accurate, then this will reduce the number of difficult training examples that exist due to an incorrect or partial detection.

VI. Conclusion and Future Work

As a prototype, we have developed a system which can recognize pets climbing on top of pieces of furniture and

Fig. 4: Two examples of videos from the "Dog not biting" (in green) and two from "Dog biting" (in red) dataset are shown. The dataset is created from YouTube videos of dogs performing a variety of actions. The labels, however, are only based on whether the action that is being performed is "biting" or not.

Fig. 5: Qualitative results of PawPal showing the detection of a dog that moved from the ground onto a couch. While on the ground, the dog is detected in green but once the dog is on top of furniture, the detection turns red and the "Dog on furniture!" alarm is sounded.

biting objects by processing video feed in an online manner. We will provide more extensive tracking capabilities by extending pet activity classification to multiple classes including walking, running, playing etc. We will use robust bounding box coordinate fitting methods such as RANSAC to allow for accurate detection of each pet for households with multiple pets. The pipeline will be optimized to run in real time given a stream of video data. The hardware will include a speaker mounted on the pet which will speak to the dog in the owners voice.

Fig. 6: Qualitative results are shown of PawPal detecting a dog performing the action of "biting" in a video.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] W. Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017.

[3] E. Hofesmann, M. R. Ganesh, and J. J. Corso. M-pact: Michigan platform for activity classification in tensorflow. *arXiv preprint arXiv:1804.05879*, 2018.

[4] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[7] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015.