Localization and Detection in Dense Environment for Tracking

Pallabi Ghosh, Shabaz Basheer Patel
Stanford University
pallabig@stanford.edu,shabaz@stanford.edu

Abstract

Given an image from a video, we learn features in a supervised manner using deep networks and detect ants in a crowded environment. In order to track the motion of an object in a video, it is essential that we are able to localize and detect most of these objects in every video frame. In this paper, we propose an architecture for our convolutional neural network using which we achieved a precision of 61% and a recall of 50% in detecting ants in the video frame.

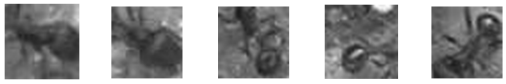

Figure 1. With ants

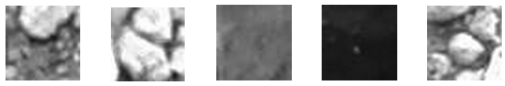

Figure 2. Without ants

1. Introduction

Convolutional Neural Networks has recently achieved great success in computer vision problems. It has been applied to object classification as well as detection. We apply both of these to ant colonies. This problem is difficult as our aim is to find most of the ants in an extremely crowded situation.

Our data-set is composed of the video captured in Professor Gordons Lab at the Stanford Biology Department. Their lab works with tracking the ants to study their communication system and other behavior. Each frame is of size 1920x1080. Until now, manual labeling of ants was done by using linear interpolation techniques to find tracks between two clicked ant positions which might not be at consecutive frames. This task was taking too long and thus there is a requirement for automatic detection techniques to speed up the process. Thus, our problem is to solve this by finding most of the ants in the video frame. Once we locate the ants, we can track their movement. The video data-set was partially labeled manually which served as our training data.

Our approach to solving this problem was to use the features of ants that a human looks for, to identify them. It becomes difficult even for a human to identify all ants. The characteristics we look for are color, shape, and texture. However, its also the movement of an ant, which makes us realize that the previously undetected object is actually an ant. Hence, our initial approach towards the problem was to also take advantage of having a video with a static background rather than just an image.

2. Related Work

Work by Pierre et. al.[1] addresses the similar problem by exploring the entire image by densely running the network at each location and at multiple scales. This approach yields more significant views to vote which improves its robustness. They also generate object bounding box by combining the regression predictions along with the classification result at each location. Similar problem has also been addressed by Girshick, Ross, et. al.[2], the tool is also called rCNN. However, to the best of our knowledge, this is the first work to look at images with very high object densities. In our application, every video frame contains around 100 ants. Some examples of training images are shown in Figure 1 and 2.

3. Technical Approach

The following describes the approaches taken by us, in order to improve performance.

3.1. Using Background Subtraction

In order to take an advantage of having a video with a fixed background, we use background subtraction, which is a widely used image processing technique to detect moving objects in videos from a static camera. Using this, we

Arch.	conv1	conv2	conv3	conv4	full5	full6	full7
CNN-F	64X11X11 st.1, pad 5 LRN, x2 pool	256X5X5 st.1, pad 2 LRN, x2 pool	256X3X3 st.1, pad 1	256X3X3 st.1, pad 1	64X8x8 dropout	2X1X1 dropout	Softmax

Table 1. Pre-trained VGG's CNN-F Architecture[3]: Each architecture contains 5 convolutional layers (conv 14) and three fully-connected layers (full 57). The details of each of the convolutional layers are given in three sub-rows: the first specifies the number of convolution filters and their receptive field size as num x size x size; the second indicates the convolution stride (st.) and spatial padding (pad); the third indicates if Local Response Normalization (LRN) is applied and the max-pooling downsampling factor. For full 13, we specify their dimensionality, which is the same for all three architectures. Full5 and full6 are regularized using dropout, while the last layer acts as a multi-way soft-max classifier. The activation function for all weight layers (except for full7) is the REctification Linear Unit (RELU).

are able to differentiate the foreground from a background in a video frame. A background-subtracted (BS) image is shown in Figure 3. Baseline result is achieved by forming bounding boxes around the ants using BS as shown in Figure 4. To improve results, we proceeded on by applying BS image as one of our channels along with RBG channels. On training a 5 layers CNN using this four-channel image gave reasonable results. However, results were not good in terms of number of ants predicted i.e. recall although our precision was good. Possible reason for this is because of the ants which are still in the video because of which it is also being considered as our background. The other three channels indicate shape and color as an important feature for there being an ant. Hence, BS image does not help in training the CNN as it gives extra information for training images in which ants were moving, for the rest it doesnt.

3.2. Ensemble Technique

Hence, we removed the fourth channel BS image and trained our CNN using only the RGB image. We then used five layers CNN and started our experiments and we observed results better than the previous values. Following this, we then decided to use an ensemble technique of using multiple CNNs in order to decide a presence of ants in a better manner. In order to get different features, we used CNNs of different number of layers. Thus, we used CNNs having 5,6 and 7 layers and formed an ensemble using these.

3.3. Non-Maximum Suppression

From the above-described ensemble technique, our recall had increased but the precision decreased as even the adjacent image blocks around the ant, which had a part of its body, were being detected as ants. Thus, in order to improve the precision, we apply Non-Maximum Suppression (NMS) technique[3], in order to remove most of the false positives. NMS is used after the detection made by the CNNs. This algorithm has the effect of suppressing all the image information that is not part of local maxima. Thus, with this architecture, we observed better precision and recall. The block diagram of the architecture is depicted in Figure 5. The architecture has been discussed in more detail in the experiment section.

3.4. Transfer Learning

We also applied transfer learning from the CNN-F(CNN Fast) VGG model[4], by taking the first 4 layers, which extracts the essential feature. The architecture has been shown in table 1. This technique doesn't give better results. Results are discussed in detail, in the next section.

3.5. Data Augmentation

It was also challenging to have the labeled data. Hence, we used data augmentation techniques, such as horizontal flip to all training data, in order to improve the training of our CNNs, which also improved our results.

4. Experiments

As described in previous sections, we were using convolutional neural networks to detect ants in the video frames. It is a very complicated scene with high object density. This algorithm can be generalized to any kind of detection and localization of fixed sized objects in highly crowded scenes.

4.1. Formation of Training images

For our simulations, we use MatConvNet [5] and obtained some manually labeled frames as our training data-set. Each video frame in this manually labeled data-set did not have all the ants marked, but only the ones that had been clicked on. Rest of the ants were a part of linear interpolation in the tracking step. So we decided to use 40x40 blocks around these clicked points as the positives in the training set. 40x40 pixel size was decided as it was about the size of an ant. Then to get the negative training set, we had to use background subtraction results. We used only the regions with a high probability of being a background. Thus, we obtained about 8000 positive training set and 40000 negative training set. Out of these 48000 images, 10000 were randomly chosen to form the validation set and the rest formed the training set. We also tried to manually correct the background detected using background subtraction, to improve our results. We did this to remove ants that were not moving and hence being

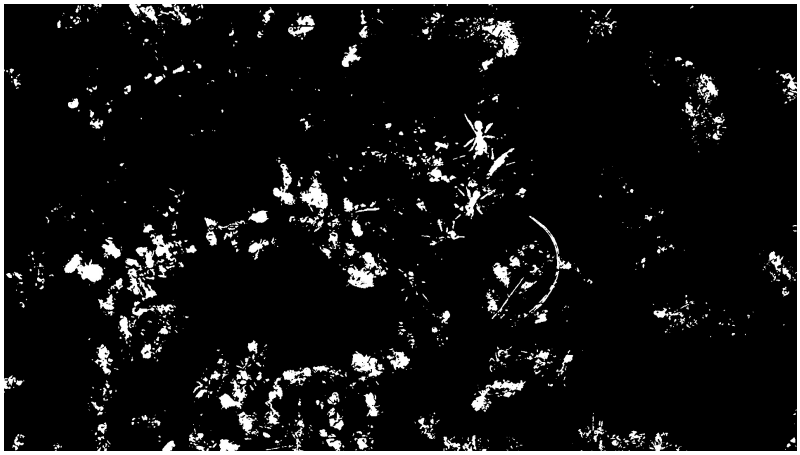

Figure 3. Background Subtracted Image

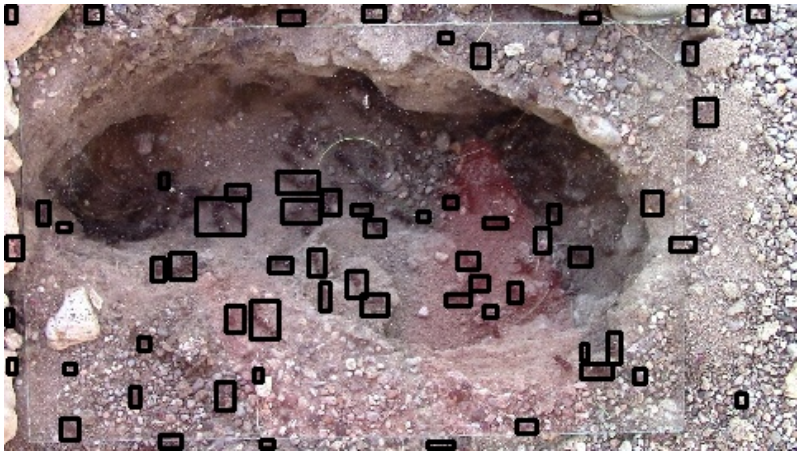

Figure 4. Output using background subtracted Image

detected as background by the background subtraction algorithm.

4.2. Performance of different architectures

Using this training set, we carried out a number of experiments and tried out a number of different CNN architectures. Using the ensemble of 3 different architectures, we achieved the best results as yet. More information about these three architectures are given is table 2. We trained our CNNs along with dropout with a probability of 0.5.

The error plots corresponding to each of the above architectures are shown in figure 6, 7 and 8. From these plots, we observe there is no over fitting. Also, we tried out different block sizes other than 40x40 like 20x20 and 60x60. The comparisons are given in table 3.

We then also experimented using pre-trained models

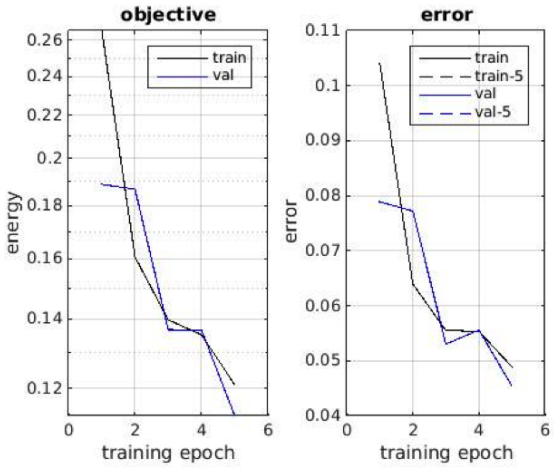

Figure 6. Energy and Error vs Epoch plot for architecture 1

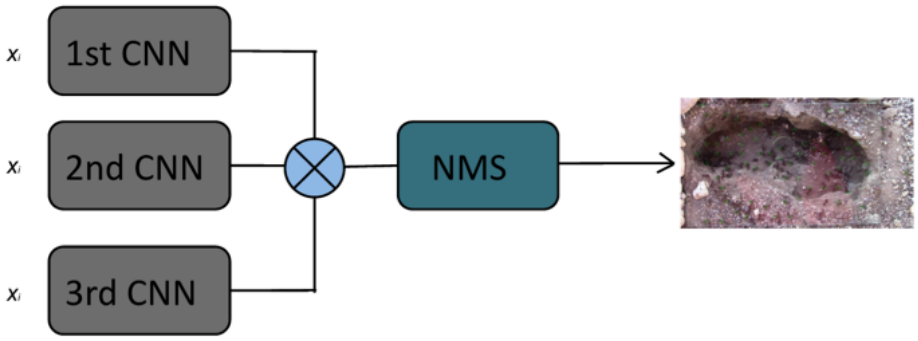

Figure 5. Block Diagram for the architecture

Layer Number	Architecture 1	Architecture 2	Architecture3
1	Conv-Max-Relu 32X5X5 st. 1, pad 2 3x3pool	Conv+max+relu 32X5X5 st. 1, pad 2 3x3pool	Conv+max+relu 32X5X5 st. 1, pad 2 3x3pool
2	Conv-Max-Relu 32X5X5 st. 1, pad 2 3x3pool	Conv+max+relu 32X5X5 st. 1, pad 2 3x3pool	Conv+max+relu 32X5X5 st. 1, pad 2 3x3pool
3	Full 64 Channels dropout	Conv+max+relu 64X5X5 st. 1, pad 2 3x3pool	Conv+max+relu 64X5X5 st. 1, pad 2 3x3pool
4	Full 2 Channels dropout	Full 64 Channels dropout	Conv+max+relu 64X5X5 st. 1, pad 2 3x3pool
5	Softmax	Full 2 Channels dropout	Full 64 Channels dropout
6	-	Softmax	Full 2 Channels dropout
7	-	-	Softmax

Table 2. CNN Architectures used for Ensemble method

from VGG. We used the first four layers of the VGG's CNN-F model. While keeping the learning rate for these four layers to be zero, we trained the FC layers. Although, the results from this architecture was not better than the fully trained ensemble architecture. This might be possible as the images from ImageNet are not similar to the images from our training set.

To test the output of our results, we used a frame of the video that had not been used as the training image. This served as our test image and we manually labeled all the ants. Then we segmented the image into 40x40 blocks and used the trained neural network to classify each block as an ant or not an ant. We used the manually labeled points as the ground truth and treated the algorithm output points that are closer to the ground truth than a specific threshold to be the correct output. So we compute the false positives and the true negatives from which we are able to compute the precision and recall.The precision and recall values of some of our trained networks are given in table 4.The output computed a test image is shown in Figure 9.

Based on the results above we have been able to achieve

Architecture	Epoch	Window Size	Error(%)
[Conv-Max-Relu]X3-[full]X2-softmax (6 layers)	5	40X40	5.5
[Conv-Max-Relu]X2-[full]X2-softmax (5 layers)	5	40X40	5
[Conv-Max-Relu]X4-[full]X2-softmax (7 layers)	20	40X40	2.5
[Conv-Max-Relu]X3-[full]X2-softmax (6 layers)	22	20X20	6.5
[Conv-Max-Relu]X3-[full]X2-softmax (6 layers)	5	60X60	1.5

Table 3. Comparison between different architecture and block sizes

Architecture	Precision(%)	Recall(%)
5 Layer	49.35	44.19
6 Layer	49.33	43.02
7 Layer	57.58	44.19
Ensemble of all above	60.56	50.00
Transfer Learning from VGG	41.03	18.60

Table 4. CNN Architectures used for Ensemble method

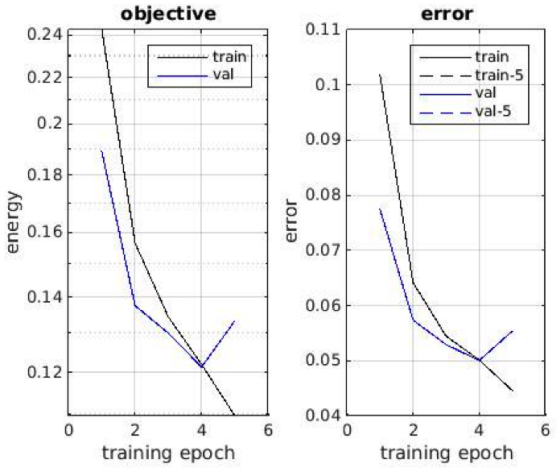

Figure 7. Energy and Error vs Epoch plot for architecture 2

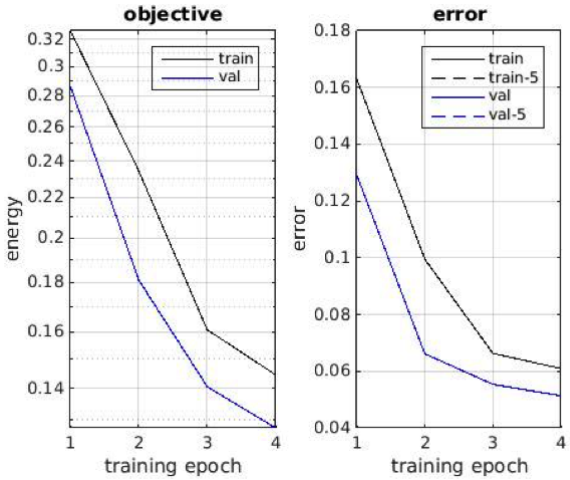

Figure 8. Energy and Error vs Epoch plot for architecture 3

a precision on 61% and recall of 50%. The errors in the output of this algorithm, can be manually corrected and then the results can be passed on to the tracking framework.

5. Conclusion

This work describes a complete pipeline for the detection and localization fixed size object detection in very crowded environment.

We have tried different architectures, to track ants in a very crowded environment. This can be generalized to any small object such as humans from a video captured by a quadcopter or a surveillance camera. Through the course of trying out these different algorithms, we learned the pros and cons of depth of a neural network, data augmentation, hyper-parameters etc. We also generated the training data-set and learned the importance of the quality of the data-set on the final accuracy.

We now intend to develop a full framework, where we use our architecture to detect the ants and then after manual correction of the detection results, we apply the tracking algorithm such as K shortest path routing. We also plan to improve the detection algorithm maybe by collecting a larger labeled data-set or applying a deeper architecture. Presently, the data-set is not large enough for a deeper CNN and hence trying such a network results in over-fitting.

Finally, cause of the way our training data was labeled, we had to take a window around a clicked point. We saw that although this is a decent idea, but there are some problems, like many of the positive training set images contained only a partial view of the ant. So we felt the neural network couldn't really train on the shape of the ant which is a very important feature. Hopefully in the future, we will get manually labeled bounding boxes around the ants instead of points, which should improve our results a lot.

Figure 9. Output from the CNN architecture

6. References

[1] Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).

[2] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.

[3] Chatfield, Ken, et al. "Return of the devil in the details: Delving deep into convolutional nets" arXiv preprint arXiv:1405.3531 (2014).

[4] Wang, Tao, et al. "End-to-end text recognition with convolutional neural networks." Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE, 2012.

[5] Vedaldi, A, et al. "MatConvNet – Convolutional Neural Networks for MATLAB" arXiv preprint arXiv:1412.4564 (2014).

[6] Sobral, Andrews, and Thierry Bouwmans. "BGS Library: A Library Framework for Algorithms Evaluation in Foreground/Background Segmentation." Handbook on" Background Modeling and Foreground Detection for Video Surveillance: Traditional and Recent Approaches, Implementations, Benchmarking and Evaluation. 2014.