Fast Unsupervised Object Localization

Dwaraknath, Anjan
anjandn@stanford.edu

Menghani, Deepak
deemeng@stanford.edu

Mongia, Mihir
mmongia@stanford.edu

Abstract

As opposed to image classification, object localization is a computer vision problem whose solutions have not measured up to human level performance, even with the use of deep learning. Common approaches to address this problem in the deep learning framework, are moving sliding windows across the image, training neural architectures to predict bounding boxes, and using classic image processing techniques such as SIFT and Region Proposals. None of these methods utilize the innards of the classification neural network to localize images. In fact, one could argue that these methods are created with the assumption that neural networks are just abstruse black boxes that give us an output for every input. We posit that a neural network for classification has enough usable spatial information to localize objects. We introduce a powerful and novel technique that harnesses only a pretrained CNN to localize an object in an image. To be clear, we do not train on any bounding box data. We obtained very promising results. In our validation we found the center of the localization method's bounding box was within the ground truth bounding box in more than 90% of the cases. The core idea of our technique can be extended to other computer vision problems such as multi-class object localization and image segmentation. In addition, since our technique works so well with only a classification neural network, there is good reason to believe that we can improve the methods that currently train on bounding box data.

1. Introduction

There are several application domains such as biology or astronomy where one can get a large data set, but specifically annotating bounding boxes is impossible because this would require domain level expertise and individuals who have domain level expertise do not have time to spend weeks annotating bounding boxes. Beyond that, many algorithms that do rely on annotated bounding box data also rely on region proposals [3],[6] that give possible locations of objects. These methods plug these regions into the CNN and see what class scores are achieved. If a specific class score is high, then the algorithm knows that the proposed region corresponds to a specific class. Usually these region proposals are generated by algorithms that were developed before deep learning became very popular and thus have no grounding in neural networks.

In the former case(where there is no bounding box data), the only tool one may have at hand is a classification neural network. In the latter, one has a lot of information at hand but using a method like region proposals seems inaccurate and wasteful, especially when a classification neural network has been trained to differentiate between different objects which requires having spatial "knowledge". In addition, CNNs are just a series of convolutions applied one after another, and outputs of convolution operations correspond to specific spatial locations in the previous layer. Rather than using region proposals from algorithms such as Selective Search [10], it would be interesting and potentially more useful to see regions proposed by a classification neural network. Thus it seems natural to delve into the innards of a pretrained CNN in order to somehow produce locations of objects. In fact it seems egregious, that though it seems so natural to use a classification network for localization, there is very little to zero work on this problem.

Since there are very good practical and scientific motivations, we set out to show that classification networks can indeed provide very useful information to localize images.Overall, our answer to the problem of localizing images with only a pretrained CNN, relies on a combination of two techniques. There has been much work on visualizing what features of images make neurons fire. Methods such as "deconv" [8] and "guided backpropagation"[11] have proven to be really effective at visualizing neurons. We use "guided backpropagation" to map important neurons back into the image. We decide which neurons are important by the DAM heuristic which we will introduce in this paper. By combining the ability to find important neurons and the ability to map neurons back into the image, we have technique that can locate objects of interest.

To be precise, we develop techniques that take in an input image and produce a bounding box for the object located in the input image.

2. Background

As far as we know, there is no work on producing locations of images just based on a classification CNN. There are methods [3],[7],[6],[9] which train on bounding box data and use a classification network to some extent. In OverFeat and integrated localization, classification, and detection framework[7], a classification CNN is trained. Then a bounding box regression network is trained on the activations of a certain layer inside the CNN. In the R-CNN framework [3], an algorithm such as "Selective Search" [10] proposes about 2000 regions of interest which are passed into a CNN. At some layer in the CNN, the activations are passed into a regression network for predicting bounding box data. Although, these methods are using the CNN to extract features, they also have the additional information of bounding box data, which in some sense makes the problem easier. In addition, these systems,with the exception of [6], are not totally deep learning systems especially because they require having region proposals that come out of the old guard of computer vision. Although this should not be a knock on these methods per se, one would expect a deep learning system, as is the case in classification, to operate totally on its own without extra aid from signal processing.

In addition to these methods, in a paper by Oquab[4], a localization method is developed that does not use bounding box data. The difference between our solution and the solution presented here, is that the authors train a different architecture. Rather than having K outputs corresponding to K possible classes. The authors have an m by n by K output, where m and n index a possible range of locations in the image. Thus if there is a high output in location (1,2,10) and a high output in (7,8,12) it means in the image object 10 is in a certain location and object 12 is another location farther away from object 10. The authors do this by enlarging images up to 500 by 500 and sliding the neural network along the image(since the input size to the neural network is actually smaller). Thus the output of the CNN operating on the image is a 3 dimensional array(2 dimensions for locations that the neural network operated on and a 3rd dimension for class scores). In a follow up paper by Oquab[5], the affine layer is transformed into an equivalent convolution layer, which prevents sliding windows. In [5] and [4], the authors develop a training objective that accounts for multiple locations and the fact that images have several objects in them. In [5], although there is no bounding box data, the authors are able to perform very good localization. They are able to get localization error up to about 70 percent although the authors are using a data set that has far fewer classes compared to something like ImageNet. This work demonstrates one does not need box data to train to have good localization performance which is of valuable importance for practical applications.

A commonality among [5],[3],[7],[6],[4] is that these methods create several copies of the image by rescaling the image. They do this in order to train their classifiers and regressors to be better be able to handle objects at different scales.

In terms of understanding the innards of a CNN there are 2 techniques that are very useful. Methods suchs as "deconv"[8] and "guided backpropogation" [11] have been used to visualize what image features make neurons highly activated. These techniques can create figures in the input image as in Figure 2.Although these images look very nice, there is yet no theoretical basis for why these images look so visually appealing compared to using something like normal backpropagation[2]. We intuitively believe that the images created by "guided backpropogation" represent what makes a neuron fire, but there is no good theoretical reason why this actually works. Given that "guided backpropagation" maps onto distinct features in the image then the ability to map neurons back into image space would be very useful in locating objects.

Overall, in the current literature there are hints that a classification CNN could be used by itself to locate objects even without bounding box data [5],[4],[8],[11]. There is however, no piece of literature that has been able to localize objects solely based on a Pretrained CNN. We hope to combine new insights with previous literature, namely "guided back propagation", in order to accomplish this goal.

3. Approach

Below we describe the overall algorithm for localizing the object in the image. We then explain each point of the algorithm in detail in the ensuing paragraphs.

Algorithm 1 Localization Algorithm

1: **procedure** FASTLOCALIZATION($k, kmax$)
2: Pass the image through the VGGNET-16 to obtain the classification
3: Identify the $kmax$ most important neurons via the DAM heuristic
4: Use "Guided Backpropagation" to map the neuron back into the image
5: Score each region corresponding to individual neurons by passing those regions into the CNN
6: Take the union of the mapped regions corresponding to the k highest scoring neurons, smooth the image using classic image processing techniques, and find a bounding box that encompasses the union.
7: **end procedure**

3.1. Passing Image through VGGNET

In order for our algorithm to work we need to have the activations of the neurons at the 11th layer as well as the

class score of the image. Thus we pass our image through VGGNET and cache the relevant values.

3.2. Finding the $kmax$ most important neurons

We focus on the neurons in the 11th layer.We do this because all high level layers contain high level features. (The fact that we choose 11 over 12, is because our algorithm was working when using layer 11) We do not use an affine layer however, because affine layers lose spatial information. For each input image, the highly activated neurons in the 11th layer are the ones of interest because these neurons are intuitively more likely affecting the final output score. Unfortunately, at layer 11 there are usually approximately 1000 neurons that activate. In addition, if one were to do "guided backpropagation" from the neuron back into image, one would find that some of these neurons are mapping into regions not at all related to the classified object. Thus we need a heuristic that can cull a subset of these 1000 or so neurons. In order to find more appropriate neurons, we also measure the affect of the neurons on the class score. We do this by calculating the gradient of the class score with respect to the neurons in the 11th layer.

Thus we first gather 2 quantities.

$$Activations \tag{1}$$

Which is the activations of the neurons at the 11th layer, and also

$$Effect = \frac{dClassScore}{d11thLayerNeurons} \tag{2}$$

which is the gradient of the class score with respect to neurons at the 11th layer. Notice that both $Activations$ and $11thLayerNeurons$ have the same dimensions. Also notice that Effect can be calculated computationally with a backward pass that extends from the final affine layer to the 11th layer. Now in order to compute a ranking score for each neuron we point wise multiply $Activations$ and $Effect$.

Thus intuitively if a neuron has a high activation but affects the class score only a little or negatively with small perturbations in its own value, then we do not give it a high score. If however, the neuron has a high activation and affects the class score highly for small perturbations in its own value, then we give it a high score.

$$RankScore = Activations * Effect \tag{3}$$

At this point we choose the $kmax$(which is approximately 10 in our experiments) neurons with the highest values and we proceed to next part of the algorithm.

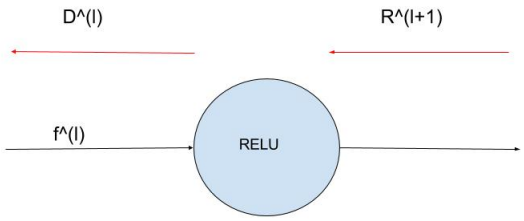

Figure 1. Backward and Forward Signals around Relu

3.3. Guided Backpropagation

In the literature, simply calculating the derivative of the neuron with respect to the image has been experimentally shown not to reveal visually appealing features. On the other hand "guided backpropagation" [8] has been shown experimentally to produce visually appealing features in an image that seem to correspond to what usually activates a neuron highly. In Figure 1 we show the difference between using a simple gradient versus "guided back propagation". Because "guided back propagation" seems to offer experimental advantages it is our method of choice to map neurons back into the original image.

There is only a small difference between normal back propagation and "guided backpropagation". Here we explain the difference. Suppose as shown in Figure 1 , that we are using normal backpropagation, and in the backward pass we are at juncture where the backward signal is going through a Relu. Normally the signal D^l going out on the left side of the Relu node will be

$$D^l = R^{l+1} * (f^l > 0). \tag{4}$$

In "guided backpropagation"

$$D^l = R^{l+1} * (R^{l+1} > 0) * (f^l > 0). \tag{5}$$

3.4. Image Processing with Blobs

After we "guided backprop" a neuron into the image, we then find the region in the image which has been activated. We do this by simply keeping the pixels that have values in the 80th percentile. We us image processing techniques like dilation and erosion to guarantee that the remaining blob

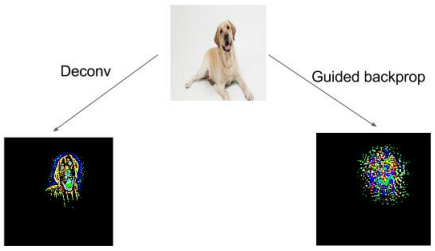

Figure 2. Difference Between using "guided backpropagation" and Normal Backpropagation. Both of these processes start at the same neuron

is continuous simply connected.The erosion operations ensures there are no small islands, and the dilation operations ensure there are no small holes. We do this with $kmax$ neurons.

3.5. Choosing k of kmax neurons

We now have to choose the best k of $kmax$ neurons. In order to do this we individually pass the regions corresponding to $kmax$ neurons through the CNN. Of those $kmax$ neurons we choose the top k scoring neurons. We union together the 80th percentile regions of the top k scoring neurons.

In order to find the bounding box, we simply find the minimum and maximum coordinates in the x and y dimension of the image. We let these 4 values set the dimensions of our bounding box. In other words, the 4 corners of our bounding box are (xmin,ymin), (xmin,ymax) , (xmax,ymin) , and (xmax,ymax).

3.6. Additional Capabilities

The ingredients above provide a recipe to localize the object that has been classified by the CNN. We can easily modify the above algorithm to find multiple objects in an image. We wont validate this on a big data set but we can find for example a pig among many dogs. Briefly we describe a simple modification in Algorithm 2 that accomplish such a thing.

We can simply just find class scores that are high by some metric and feed those classes into the heuristic.

In addition "guided backpropagation" often forms a very good contour of objects. Thus "guided backpropogation" paired with the DAM heuristic could very well segment images as well.

Algorithm 2 Multiple Localization Algorithm

procedure FASTMULTIPLELOCALIZATION($k, kmax$)
    Pass the image through VGGNET to obtain classes that have similarly high scores
    Identify the $kmax$ most important neurons for each class via the DAM heuristic
    Use "Guided Backpropagation" to map the neuron back into the image
    Score each region corresponding to individual neurons by passing those regions into the CNN.
    Take the union of the mapped regions corresponding to the k highest neurons for each class and find a bounding box that encompasses the union
end procedure

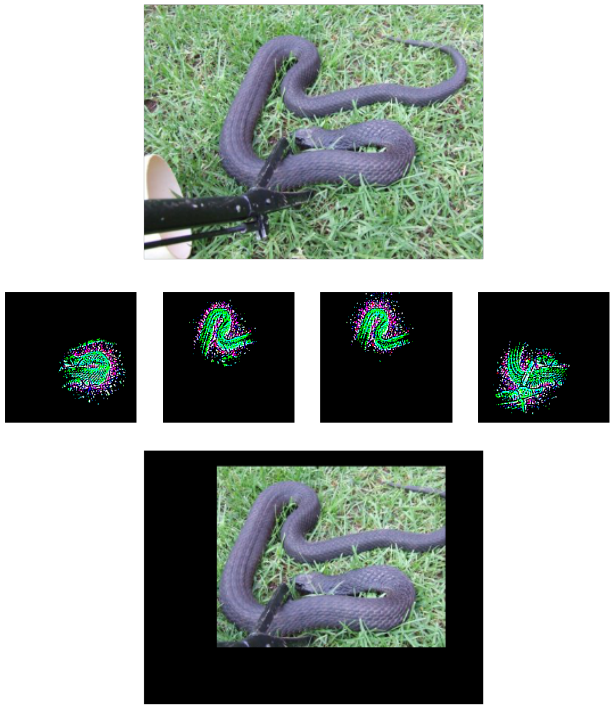

Figure 3. The Figure shows the algorithm steps. Top Image is the original input. Middle Layer shows the deconvolution output of neurons selected using the DAM heuristic. The Final image is the output box reconstructed from the union of processed deconvolutions from the middle layer

4. Experiments

We arrived at the choice of layer 11 for "guided backpropagation" after a few experiments. In particular, we tried layers 9,10,11, 12, and affine layers. It was our observation that layer 11 and 12 produced similar output. We decided on layer 11 because of the computational savings. We did not use affine layers because they produced really bad results.

We also tried increasing the parameter $kmax$ from 10

to 30, but it did not have a significant impact on the localization accuracy. This means our DAM heuristic does indeed extract the best neurons for reconstruction of the object bounding box. In the next section we describe results where $kmax = 10$, $k = 5$ and layer = 11.

We experimented with different values of the parameter k, with mixed success. Larger objects did well with high values of k and smaller object did well with lower values of k. This is a part of the algorithm we would like to improve in the future. At this point we were trying to demonstrate that objects can be localized from a classification CNN. Keeping this in mind, we decided to keep k constant.

5. Validation

Although we used no training data for the localization algorithm, we can still use the localization data sets for validation. Below we outline our methodology of validation and the results and observations about the performance of our algorithm

5.1. Validation Methodology

We obtained the dataset from ImageNet. In particular the dataset consists of URLs of images and an XML file for each image describing the image. The XML file contains bounding box and object class information for the objects present in the image. Since our method currently is designed for localizing only one object, we consider images with only one copy of the object.

The procedure we adopted is as follows. We randomly sample from the 1000 classes, draw a random image of that class from the dataset, ensure that there is only one object of that class. We give the image to the algorithm and ask it to for a bounding box. We then compare our bounding box with the ground truth bounding box using some metric.

5.2. Scoring Methods

To evaluate how well the algorithm is performing we need to come up with a number describing how similar two bounding boxes are. Since the accuracy of the bounding box is a very subjective quantity, no method is going to be perfect. We discuss two methods here and argue that while both are imperfect together they give a good idea of how the algorithm is performing.

5.2.1 Intersection over Union

This metric is straightforward and is used extensively in the literature. It is calculates the area of the intersection of the two bounding boxes and dividing it by the area of the union of the two bounding boxes. The ratio ranges from 0 to 1, with 0 signifying no overlap and 1 denoting a perfect match. This metric is a very simple metric to measure overlap, however it has some weaknesses. In particular if the algorithm has a slightly conservative box, i.e. it includes a little more of the image at the boundaries making a larger bounding box, it gets severely penalized in the intersection over union metric. Although subjectively the bounding box is fine. For example, if we take a look at Figure 4, we that the ground truth is a very tight box, but the algorithm's output although subjectively fine has poor intersection over union value (IoU) of 0.6. Thus most correct localizations have IoUs of 0.5-0.6, however having an IoU of 0.5-0.6 doesn't guarantee a correct localization.

Figure 4. Original Image, Algorithm output, Ground truth

5.2.2 Center Correctness

In order avoid the limitations of the IoU metric, we have used another metric which might better capture the correctness of the bounding box. Similar methods have been used in [5]. In this metric, we check if the center of our bounding box is within a region around the ground truth bounding box's center. We set a margin of say 20%, we then construct a smaller box within the ground truth bounding box excluding 20% margins on all sides. We then check if the center of the algorithm's bounding box is within this size restricted bounding box. If it is, then we say that the algorithm correctly localized the object when a 20% margin is used. Higher the margin more restrictive the measure. 0% margin would just check if the center is within the bounding box. A margin close to 50% would enforce the bounding box centers to be extremely close. This metric is more robust to boundary effects that the IoU method is prone to and we hope that it gives a better idea of the accuracy of the algorithm's localization.

5.3. Discussion of Validation Results

Our validation data set consists of around 4000 images. This is after removing noisy images which were all white or black. The 4000 images includes all classes, we also have 3 smaller data sets of single class images for around 300 images each. The 3 classes are bottles, bi-cycles and cats. We evaluate the above metrics on each image and we present here the accumulated results. Figure 5 shows the histogram of all the IoU scores. The mean IoU score was found to be **0.4679**. As discussed above good bounding boxes get values of 0.5-0.6. So this shows that the algorithm gets a large number of bounding boxes correct. If we look at the histogram we see that that many images fall in the range of 0.6 to 1 as well. In a later section we also discuss the reasons

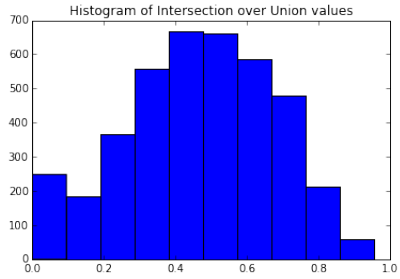

Figure 5. Histogram of IoU scores for 4000 images

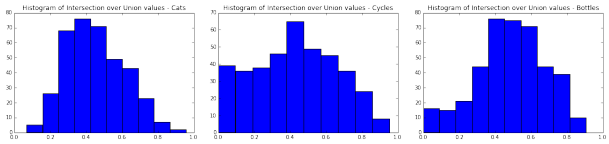

Figure 6. Histogram of IoU scores for 3 classes

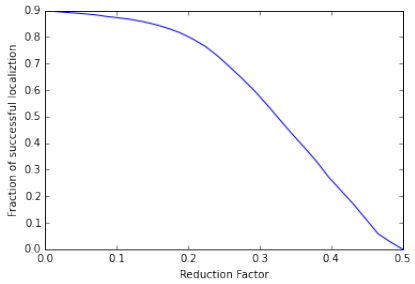

Figure 7. Center Correctness score vs. margin fraction for 4000 images

the cases where the algorithms performs poorly. In figure 6, we see that we obtain similar results in the class specific case as well. We have tabulated the results for the 3 specific classes in the table below.

Class	Mean IoU	No. of images
cat	0.451	370
cycle	0.427	386
bottle	0.486	411

We also evaluated the center correctness metric, which we feel gives a better indication of the algorithm's accuracy. We plot in Figures 7,8 the fraction of data classified vs the margin fraction. The margin fraction ranges from 0 to 0.5. As we see in the graph, for a margin of 0, we classify 90% of our images correctly. This means that in 90% of the cases our bounding box center was within the ground truth bounding box. For a margin of 0.2, we get 80% accuracy. This shows that our algorithm is doing very well even though it was never trained for localization.

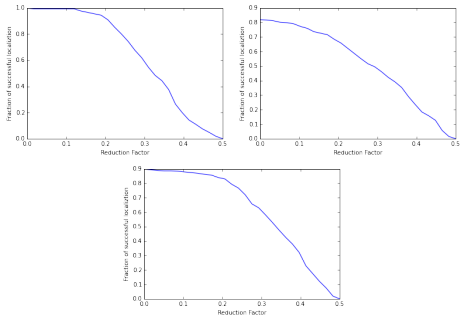

Figure 8. Center Correctness score vs. margin fraction for 3 classes

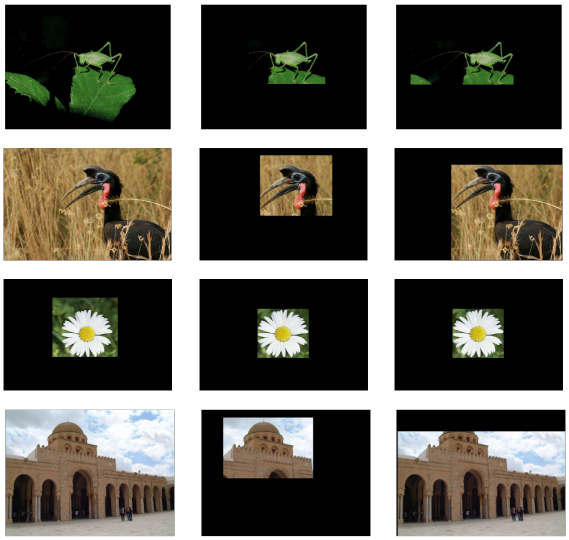

Figure 9. Original Image, Algorithm output, Ground truth

Figure 10. Result for pre-specified class object Detection. Left Image is Input, Right Image plots the localized area

6. Discussions

In this section, we provide a qualitative discussion of the algorithm and briefly discuss the different conditions in which the algorithm doesn't do well.

6.1. Localization Effectiveness

The high localization accuracy in the validation results presented in Section 5 shows that the method is very effective at extracting spatial information from the CNN. Figure 9 shows the results of our algorithm for some random

images from the ImageNet localization challenge dataset.

The algorithm performs well in localizing a specified class object from the image of multiple objects, as evidenced by the example in Figure 10. The algorithm was given an input to search for a pig in the image, which it could successfully do, without needing to run sliding bounding boxes across the image.

6.2. Problem Areas

Using the validation data set to visualize our results versus the ground truth, we have identified three major problems-

1. Misclassifcation: Like most other localization methods, our method heavily relies on the CNN being able to correctly classify the object. The neuron selection heuristic, is highly dependent on the gradients of neuron activations with respect to the correct class score. If the class is not correctly identified, the selection of neurons fails, which causes the wrong localization output.

2. Small Object Localization: Our method has low success with localizing small objects. We hypothesize multiple reasons for this. Firstly, small objects in the image have a high misclassification rate and hence, are hard for us to localize. Secondly, we are using a constant number of neurons(k=5) from the DAM heuristic which could be very high for tiny objects. Thirdly, the neural activations corresponding to small objects might not be enough for them to be picked up by the DAM heuristic. These examples can be visualized in Figure 11.

3. Head Body problem: The algorithm described is good at localizing the most important aspects of the object but tends to leave out the extremities of the object. The reason for this is two fold. Firstly, we are currently using a fixed number of neurons for reconstruction. Secondly, the top neurons selected tend to have a higher "guided backpropagation" area overlap and they tend to capture the most important parts eg. a Dog's face for the dog. We believe that future work to improve the selection and union of neuron "guided backpropagation" areas should help improve performance. This problem is illustrated in Figure 12

Besides the above issues, noise in data such as empty/bad images and wrong bounding box data due to human error/ambiguity, also affect our results negatively.

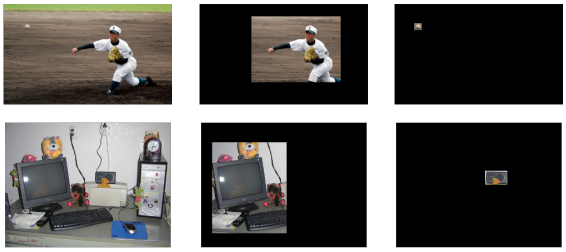

Figure 11. Original Image, Algorithm output, Ground truth - Small Object Localization Problems

Figure 12. Original Image, Algorithm output, Ground truth - "Head-Body" Problem

7. Conclusion And Future Work

As stated above, our unsupervised method works quite well. The key point here is that deep neural networks are trained to differentiate between classes, and as long one believes that "guided backpropogation" conveys information that activates a neuron and that the DAM heuristic is an effective ranking of neurons, then our method is picking up distinguishing features. We however might not pick up legs and more generic features that are common to all sorts of classes. In order to do very well, the neural network needs some knowledge of what humans consider good boxes for objects. A very promising direction in the future would be to combine our method with bounding box training data and region proposal methods. Our method presumably lowers the set of useful region proposals and the bounding box generated by our own method could be a feature for training a neural network on bounding box data. A more unsupervised approach to this problem, is to make the choice of neurons more intelligently. If we can obtain the neurons for other parts of the object, such as legs then we can obtain a better bounding box.

Another interesting line of work would be to flesh out how to separate multiple objects in an image. We have stated a potential algorithm on how to do this, but have not fully fleshed out the details. Along with this, segmentation would be an interesting line of research using our method.

Overall, neural networks have lots of information in them that is indeed uninterpretable. But if we can develop techniques to probe the neural network, we can make sense

of what the neural network has learned from data, which is very important for furthering this field.

Our code is publicly available in GitHub [1]

References

[1] Code repository for the project. `https://github.com/deepakrox/FastObjectLocalization`.

[2] A. G. Baydin, B. A. Pearlmutter, and A. A. Radul. Automatic differentiation in machine learning: a survey. *CoRR*, abs/1502.05767, 2015.

[3] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[4] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1717–1724, Washington, DC, USA, 2014. IEEE Computer Society.

[5] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? – Weakly-supervised learning with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, United States, June 2015.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[7] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[9] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014.

[10] J. R. R. Uijlings, K. E. A. Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[11] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.