Bounding Out-of-Sample Objects
A weakly-supervised approach

Li Quan Khoo
Stanford University (SCPD)
lqkhoo@stanford.edu

Abstract

In the context of image processing, the most salient parts of the image tend to trigger the largest activations in convnets. This also means that, whatever the task of the convnet is, it is also implicitly doing some form of localization as part of the loss-minimization process. We want to find a way to harness the this information to annotate bounding boxes around out-of-sample objects. We say that an object is out-of-sample if the bounding box network has never been trained on that object class before. This is useful when we have libraries of object categories with no associated bounding box annotation to perform supervised learning with. Our method operates as a separate, portable "auxillary network", which could be attached to a host network that simply performs image classification.

1. Introduction

ConvNets have been an integral part of image processing ever since AlexNet [6] was entered into ILSVRC-2010. Although the idea is not new, advances in our understanding, as well as continued improvements in harnessing GPU computation and memory meant that convolutional architectures have stayed at the forefront of image (and video) processing, as they are a natural fit to the nature of the data.

Because convnets' output features or activation volumes have the notion of spatial extent, they are not just detecting salient features from their input - they are also implicitly localizing the activation at some area within their activation volume. The focus of our project is about using this localization information to put bounding boxes around objects. We investigate whether we could train a model that understands objects in images at a general-enough level so that it could put bounding boxes around instances of entire classes of objects which it has never seen before. This idea of generalization is just as important as raw performance in the pursuit of general artificial intelligence and understanding. For example, a toddler could easily handle an object, even if it's the first time he or she has seen it, by making assumptions about its properties from their prior experience with other objects perceived to be visually similar.

In order to train such a general model, we must abstract away from raw pixel values, and instead operate on some higher-level summarization or understanding of the image, e.g. feature activations of a convnet trained to perform image classification. Our method then, is to train a model to make use of intermediate convnet activations to make our best guess at bounding whatever the convnet believes is most relevant with respect to its task. The nature of those bounds would depend on objects in the training set, which in turn, are drawn from the universe of all objects. One inspiration behind the idea is the prior work on algorithms for generating region proposals such as Selective Search [13] and EdgeBoxes [16], which demonstrates that even low-level (and therefore cheap) image features could suggest the presence of some object with high probability. This is humorously described in the CS231n lecture as detecting "blobby" regions. If many objects tend to be "blobby", then perhaps the model would learn to recognize that fact and generalize sufficiently well.

2. Related work

There is a large body of prior work on labelling objects' locations in images, and the granularity of the annotation itself ranges from simply the absence or presence of an object, to one or multiple bounding boxes, all the way to pixel-level segmentation. Fully-supervised convolutional models that annotate bounding boxes already perform quite well; He et. al.'s winning entry [4] for ILSVRC-2015 achieved a localization error of 0.09. Here, localization error is defined as the percentage of images which achieves an intersection-over-union (IoU) of greater than or equal to 0.5 between the ground-truth and predicted bounding box areas.

On the other hand, weakly supervised learning is a term that loosely describes methods that train the model on limited information or only with partial annotation, the exact nature of which depends on context. For example, training a bounding box annotator based only on whether an object is present or not, may be considered a form of weakly-supervised learning. The advantage of weakly-supervised approaches is that they can produce the full annotation while working with partially-annotated datasets, e.g. images with no bounding boxes, which are all-too-common in ImageNet. The drawback is that they do not perform as well as fully-supervised approaches.

As an example, Ocquab, Bottou et. al.'s model [7] learns to approximate the center of an object in an image (but not its bounds) on images in VOC 07 and COCO, when trained only on labels that describe whether the object class is present or not. However, the work which is closest in spirit to ours is [10], where the authors train a model to rank a series of region proposals according to the degree of overlap with the ground truth bounding box, and then make use of the model to rank region proposals for image classes that do not have ground truth, and their model achieves 0.3213 CorLoc on the AllView dataset. CorLoc stands for correctly-localized, and it is defined as 1 - localization error. Teh et. al's work [12] is highly similar, but their model is trained in a fully-supervised setting, and they achieved 0.6459 CorLoc on Pascal VOC.

Our method takes a different approach, by not working with region proposals at all. Our original inspiration actually comes from attention mechanisms in RNNs, in particular, from Xu et. al.'s Show and Tell [14]. In Xu's work, the RNN is being used for image captioning, therefore it makes sense to make use of an attention layer that allows the RNN to select which parts of the ConvNet's feature map to focus on, when generating each individual word. In our case, because we are working with only one bounding box per image, we can work directly with the convnet's feature map. The map itself is the attention layer, so to speak, because one would expect the parts that correspond to the salient regions would be maximally-activated, assuming the convnet classified the image correctly. Visualizations in the ZFNet paper [15] provide good intuition to the idea.

Therefore, instead of learning a way to rank region proposals as in [10], we directly learn the mapping from a set of feature maps to a single set of bounding box coordinates, and we rely solely on a convnet's ability to assign the differences in each image sample as either intra or inter-class variation, in order to perform the generalization that we need.

3. Data and preprocessing

Our candidate images were drawn from ImageNet [9] synsets which contain at least 600 images, where each must also have an annotated ground-truth bounding box. At time of download, there were 265 synsets that meet this initial criteria. Of these, 38 synsets exist as WordNet synset IDs and do not have an English label, so they were discarded. This leaves us with 227 candidate synsets. All of these images were then verified to have one and only one bounding box, therefore we could safely operate on the single bounding box assumption.

Upon further inspection, a subset of images have out-of-bounds bounding boxes, so we clipped these boxes to the maximum bounds of their respective image. Furthermore, when attempting to load the images as tensors, we found that the dimensions of some small (but significant) number of images deviate from the expected shape of (h,w,3), which corresponds to a 3-channel RGB JPEG file. These are mainly black and white images with a single channel, or 4-channel CMYK JPEG files. We simply discard any image which is not formatted as 3-channel RGB. In addition, we discard any image whose height or width is smaller than 224 pixels. After this filtering, the remaining 198 synsets which still contain at least 400 images are used for the rest of our method. In total, we have N=115,064 images.

Because we want to evaluate our method's performance on out-of-sample images, we split the 198 synsets into two datasets: The *training dataset* contains the top 160 synsets ranked by the number of images contained, and the remaining 38 makes up the so-called *holdout dataset*. Each of the two datasets contains their own training and validation images. For each synset, we set aside 50 images to perform validation with.

4. Method

Figure 1 illustrates our general setup. First, we use the training dataset to train a convnet, called the *training host*, which simply performs single-label image classification. Once trained, we freeze the host's weights, and then we attach and train a separate network, called the *auxillary network*, which is responsible for outputting bounding box coordinates. The auxillary network's input consists of feature maps from specific layers of the host network; it never knows what the actual image or class label is. When we want to evaluate the model's performance on out-of-sample images, we train another host convnet called the *test host* on the holdout dataset, and then attach the trained auxillary network to it, and finally we evaluate the quality of the bounding boxes we get. As baselines, we evaluate the model's performance without transfer learning, i.e. the quality of the bounding boxes on the same dataset the aux network is trained on.

Instead of training both the host and the auxillary network end-to-end, we freeze the host's weights before training the auxillary network in order to prevent information about bounding box labels from leaking back to the host and thus alter the host's feature maps in the process. Because this leakage never happens during evaluation, the auxillary network is necessarily operating on a set of feature maps optimized without bounding box information, so we might expect a model trained in this manner to perform less well.

A second consideration is actually the question of which

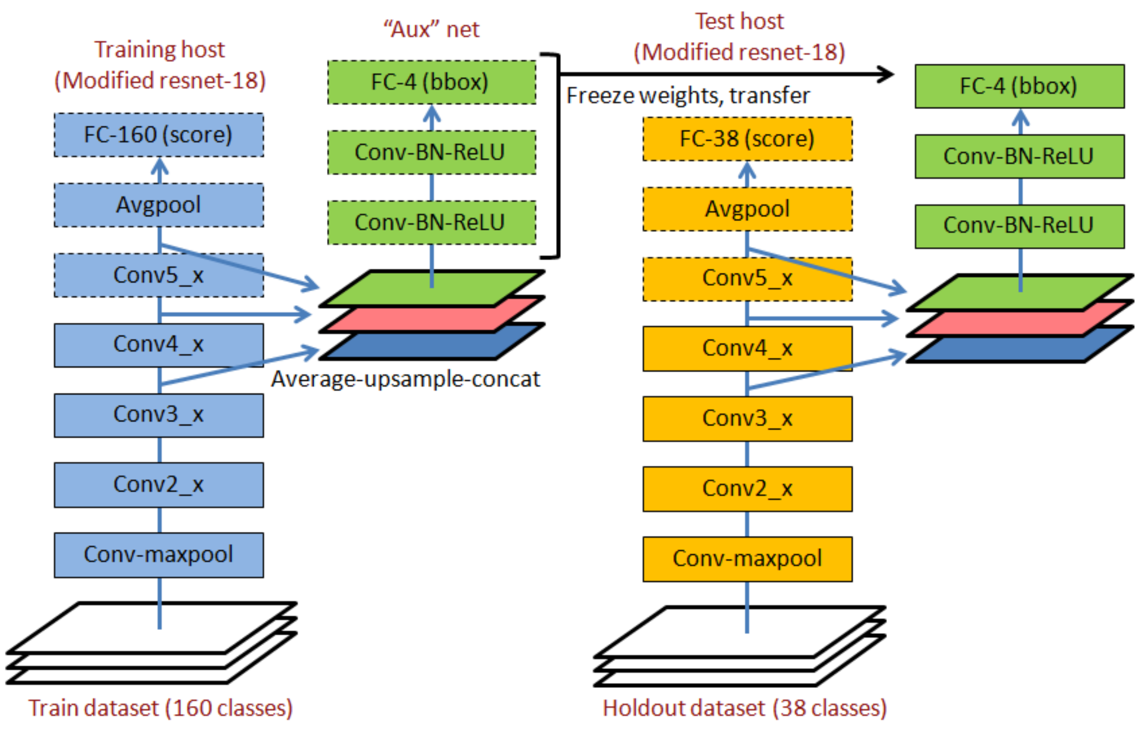

Figure 1. General architecture. A dotted outline means that layer was retrained or modified.

feature maps we should use. While lower layers in the ResNet architecture have more spatial resolution, higher level layers contain more information about the actual object class, rather than components or parts that make up the object. In order to take advantage of both, our input consists of weights from both layers at the top and lower down. Each activation volume also contains many channels. The simplest option would be to simply concatenate all the channels from all the layers we want, but that creates a very large input tensor to the auxillary network. Another idea would be to flatten individual feature volumes via a max or mean operation over the channel dimension, because what we're really interested in is the spatial information, and activations in all relevant filters, which should tell us something about the object - this could be seen in Figure 2 of He et. al.'s Spatial Pyramid Pooling paper [3]. Using the mean may result in a weaker signal due to cancellations between positive and negative values, but it would contain more information about the image (and therefore be more robust) than using only the maximally-activated channel. Here, a maximally-activated channel means the channel with the greatest sum of all activations. Despite this tradeoff, we found that the model performs better using channel-wise mean. We suspect that the optimal operation might be to perform a k-channel mean instead, to filter out noisy activations, but we did not have time to perform hyperparameter search over this variable, and its value may very well be depend on individual datasets.

4.1. Details

The hosts we used are pre-trained ResNet-18s from PyTorch's model zoo [1]. There are many variants of ResNets, but they all share the same general architecture. The first layers read in a 224x224 image with a 7x7 conv filter size. Immediately higher up are four "blocks" of layers called conv2_x to conv5_x, each with the residual connections within. The spatial size of the feature maps of these blocks also shrink by a half from one block to the next, from 112 all the way to just 7 after conv5_x. Higher up, we have the average pooling layer which connects directly to the FC, which outputs class scores. Residual nets were selected primarily for several reasons. Firstly, they are easy to train, because they converge quickly for a wide range of learning rates, thanks to the use of batch normalization and residual connections. Secondly, ResNets are relatively lightweight in terms of memory, by doing away with FC layers at the top, and they have been demonstrated to have state-of-the-art classification performance. Lastly, their feature maps have the same spatial size as VGGNet's [11], allowing us to compare the performance of the auxillary network across different hosts. However, we did not have the opportunity to run this particular experiment, because we did not manage to train the training and test hosts using a modified VGG-16 to convergence in time.

The auxillary network is a simple sequential convnet. Convolutional layers were used because the network's input

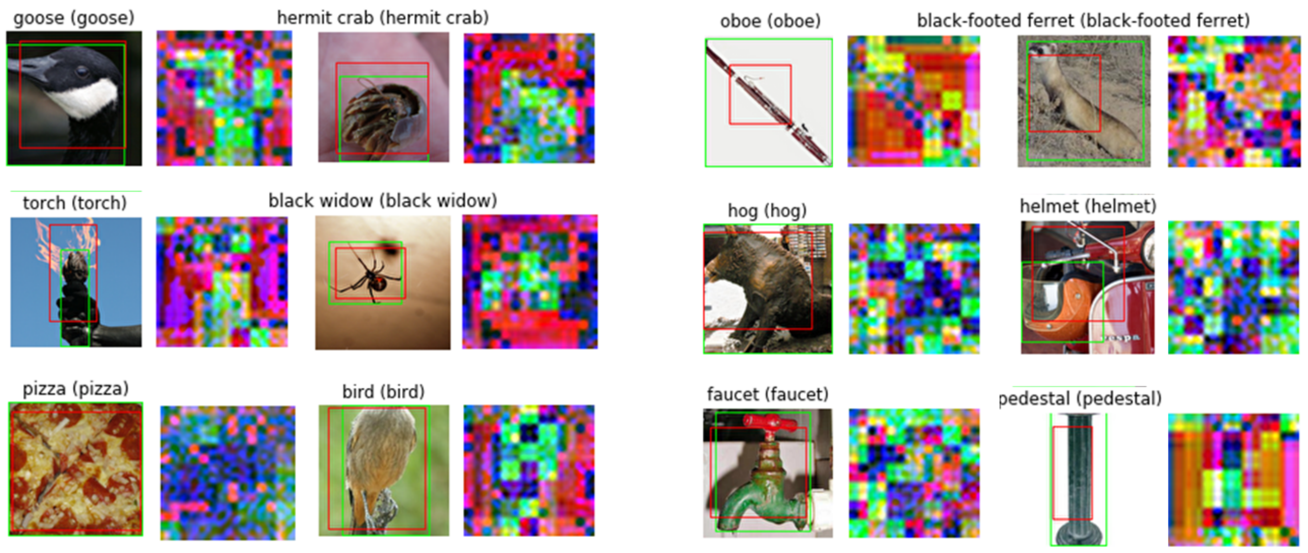

Figure 2. Model visualization. Left group is Baseline 1, right group is Baseline 2. Green box is ground truth. Red box is prediction. The color grid is the visualization of the auxillary network's input. Green, red, and blue are channel-wise averages of outputs from Conv5_x, Conv4_x, and Conv3_x respectively for the corresponding image, upscaled to a spatial size of 28 x 28, multiplied by 10 and then clamped to a max of 255.

could be interpreted as a 3-channel image, especially since the host ResNet's feature maps have been batch-normalized. Both conv layers are spatially-preserving with kernel size 3, stride 1, padding 1, with 32 filters each.

The method by which we transform the ResNet's intermediate outputs into the 3-channel input tensor of the auxillary network is thus: Starting from the output of Conv3_x, the tensor sizes are [128 x 28 x 28], [256 x 14 x 14], and [512 x 7 x 7]. We first flatten the channel dimension by taking the mean, and then we upscale all three tensors to [1 x 28 x 28] via a transpose convolution operation. For example, to upscale by k times, we would feed the tensor into a transpose convolutional layer with kernel size and stride of k, with weights locked to 1 and biases to 0. Finally, we simply concatenate all three tensors together in the channel dimension, meaning that it could be interpreted as as a false-color image.

We trained all our models with PyTorch's implementation of the Adam optimizer [5] with a minibatch size of 64, and we decayed the learning rate from 10^{-2} to 10^{-4} whenever validation error plateaus. Convergence was attained within fewer than 10 epochs. We used the cross-entropy (CE) loss function when training the hosts, and simple mean squared error (MSE) loss when training the auxillary network. Cross-entropy loss is defined as:

$$L_{CE} = -\frac{1}{n} \sum_{x_i \in X} [y_i \, ln \, \hat{y}_i \; + \; (1 - y_i) \, ln \, (1 - \hat{y}_i)]$$

where

- n is the minibatch size
- X is the set of inputs in the minibatch
- y is the image class label
- $\hat{y}$ is the predicted image class label

Mean squared error loss is defined as:

$$L_{MSE} = -\frac{1}{n} \sum_{x_i \in X} ||\vec{y_i} - \hat{\vec{y_i}}||_2$$

where $\vec{y_i}$ and $\hat{\vec{y_i}}$ are 4-dimensional vectors representing bounding box ground truth and predictions respectively.

5. Results and discussion

Quantitatively, we use the Intersection over Union (IoU) and CorLoc criteria for evaluating the quality of the bounding box with respect to the ground truth. IoU is defined as the fraction of the area of intersection between ground truth and predicted bounding boxes over their union, in terms of number of pixels. CorLoc is defined as the proportion of images having an IoU of greater than 0.5.

The performance of the auxillary network depends heavily on the host being able to correctly classify the salient object as the right class. However, due to the way we split the datasets into 160 and 38 object categories, the performance of our retrained ResNet hosts are not directly comparable to many standard results, for example, performance in the 1000-class ImageNet challenge. Instead, we created two baseline models to evaluate the quality of the transfer

	Top-1 acc	Mean IoU	CorLoc
Baseline 1	0.800	0.555	0.463
Baseline 2	0.836	0.512	0.403
Model	0.836	0.511	0.399

Table 1. Quantitative performance evaluation. Top-1 accuracy is not directly comparable as Baseline 1 is across 160 object categories, while Baseline 2 is across 38.

learning. In Table 1, Baseline 1 measures the in-sample bounding performance on the training dataset. Baseline 2 measures the same but on the test dataset. What's most surprising is that the model performs neck-to-neck with Baseline 2. This suggests that the out-of-sample bounding performance is competitive with a model trained end-to-end, at least for the test dataset, which admittedly only has 38 different object categories.

Because the performance of the auxillary net depends entirely on the quality of the feature maps of the host ResNet, we evaluate the model qualitatively by visualizing those feature maps as an RGB image, alongside the bounding box output. We shall refer to this visualization as the *color grid*. To make the grid clearer, we multiplied the individual values in the input tensor by a factor of 10, while clamping it to a maximum value of 255. Green, red and blue represent the outputs from Conv5_x, Conv4_x, and Conv3_x respectively. This produces Figure 2. We can see that the green channel from Baseline 1 quite clearly follows the outline of the salient object. For images where the host produces this kind of feature map, the auxillary network performs quite well. In the same figure, we can see that the test host in Baseline 2 doesn't produce this kind of nice feature map, and we can try to hazard a guess as to why:

Looking at Figure 2 and the first two images in Figure 3, we can see that red regions in the color grid tend to surround the green region, rather than overlapping with it, which would cause it to show up as yellow instead. The blue channel tends to be quite diffuse and tends to cover the entire image, but we believe it is important for detecting textures, or generally uniform areas, as seen in the image of the pizza and the dishrag. The green region by itself tends to correlate with only the pixels that identify the image as the correct object class, for example, it only clusters around the collar of the sweater in Figure 3. What this suggests is that the red and blue channels which correspond to lower-level features are also important for determining the bounds of the entire object. For certain object classes, we found that the red region almost serves as the complement of the green region, for example, in the image of the hook in Figure 3, and many images from Baseline 1 in Figure 2. We find this to be a very interesting property, but there is no immediate or obvious reason as to why the host network might behave this way.

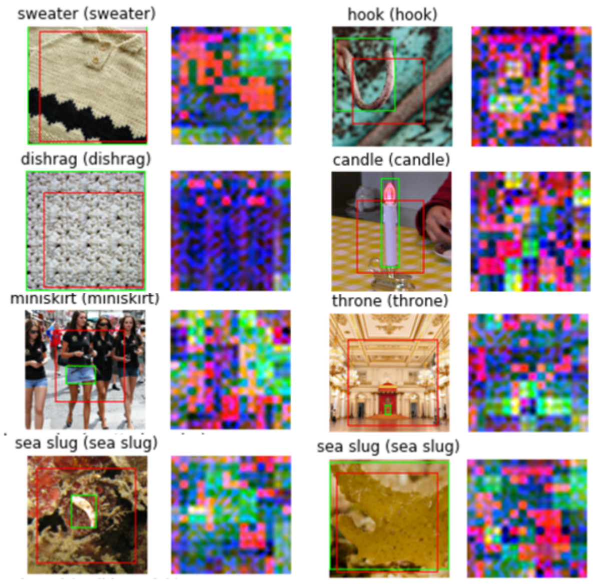

Figure 3. More example outputs from Baseline 1.

The remaining five images in Figure 3 show some problematic cases for Baseline 1. We believe these are example cases when the classifier or host network fails to generalize properly for a certain object class. We can see that the green region, which is what the network believes to be salient, does not correspond to the subject, but rather some parts of the background, similar to the example in Figure 9 of Xu et. al.'s Show, Attend, and Tell [14]. As a side note, we found that the auxillary network performs very poorly on the candle and mirror object classes. The candle tends to be classified correctly, but it may be too thin to be localized properly, as we shall explain when we look at pathological cases, but mirror is highly problematic in every case, as whatever being reflected could be classified as an object class by itself, which makes for very large intra-class variation as well as overlap with other potential class labels.

Bearing these results in mind, if we look at Baseline 2's results in 2 again, we believe that 38 classes of objects is insufficient for the network to learn the difference between inter and intra-class variation sufficiently; the vast majority of activation maps do not have any clear, visually-discernible pattern that relates it to the bounding box. Despite this visual difference, the transferred auxillary network in our main setup seems to be able to make sense of the input and give the results in Table 1. However, we feel that our evaluation, especially the model's quantitative performance, is not truly representative of what this setup can do, but we did not manage to acquire and preprocess more data in time for this report.

The model performs poorly under certain circumstances, some of which are quite obvious, as seen in Figure 4. Being a weakly-supervised method, these cases generally corre-

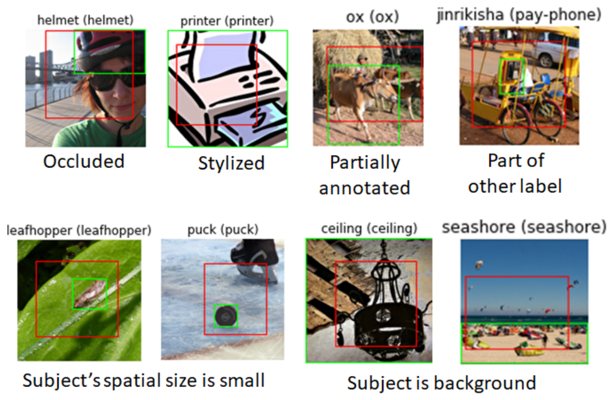

Figure 4. Pathological cases. These are outputs from Baseline 1.

spond to situations when the host cannot produce feature maps with sufficient spatial information for the auxillary network to interpret, or if the correct class label is not necessarily the subject of the image as suggested by the framing.

Because the input to the auxillary network has a spatial size of just 28 x 28, when the subject is too small, usually there isn't enough spatial information left in the feature map to localize it correctly. In addition, some images have multiple subjects, but we only have one bounding box annotation, so the auxillary network would try to bound all of the salient regions in a single bounding box, for example in the image of the ox in Figure 4. There are also cases where multiple object classes are present simultaneously in the an image, but since the host network is set up to only detect one object class, the auxillary network would only ever output one corresponding bounding box. We observed that when the auxillary network is not sure what to make of its input, it tends to output a bounding box with corners situated approximately halfway between the edge and center of the image to minimize the MSE loss, regardless of where the ground truth box might be. This corresponds to pixel coordinates (54, 54) and (168,168).

6. Further work

The model in its current form doesn't have many practical applications since its CorLoc is only about 0.4. However, the first reasonable thing to try is to scale up both the training and holdout datasets to, say, 500 object classes each, to see how far the model can improve, if at all.

Solving most of the pathological cases which we've identified is challenging. We can see no obvious solution for out-of-sample occlusion because there would be few ways to determine the true bounds of such an object. For the other cases, suppose we have a method that takes in or generates a set of region proposals as candidate bounding boxes and a convnet that recognizes the object classes, what we have is essentially Fast R-CNN [2]. What we really wanted to investigate was whether there is a good way to avoid region proposal methods (that naively generate thousands of proposals to provide high recall), and instead generate the bounding box based on some higher-level understanding of common features across different object classes. Region proposal networks in Faster R-CNN [8] seem like a good fit at first glance, but without ground-truth bounding boxes to train it in the first place, it is not a viable solution in our scenario. From what we can determine, in the case of one object class and bounding box per image, our approach is viable, but should that assumption break down, methods like Fast R-CNN, along with the comparatively slow region-proposal process [8], seem like the only way to go.

Lastly, when we set out on this project, a bonus objective was to investigate whether the auxillary network could be treated as a highly modular unit, which can be attached to a test host of a different (but compatible) architecture, e.g. a VGGNet. The use case would be that we have some classifier network already trained on a custom dataset with only image class labels, and we are interested in bounding those objects as well, but we do not have ground-truth boxes. It would be very useful if we could not, or would not need to retrain the existing host with a ResNet. A compatible architecture would need to have the following properties:

- We need to use as many feature maps as we do in the original architecture. For example, we used the top 3 feature maps from ResNet, so if we were to consider transferring the trained auxillary network to a VGGNet host, we need to do the same to it.
- Feature maps need to be of the same spatial size as the original architecture, after channel-wise average.
- Feature maps need to be outputs from batch normalization layers, so that their values are (stochastically) well-behaved, even across different architectures and images.

We did not have sufficient time to train a VGGNet modified with batchnorm layers to convergence, so we leave this as an interesting direction to pursue.

7. Conclusion

We have shown that, using our proposed setup, an auxillary network trained to annotate bounding boxes has out-of-sample performance that is competitive with its in-sample performance. We discussed the shortcomings of our research, in particular, when the dataset is too small for the host network to generalize sufficiently well in order to evaluate the auxillary network's performance fairly, and finally, we suggested alternative approaches and further work in light of our results.

8. Acknowledgements

We would like to thank the entire CS231n teaching staff. Without their assistance, this research would not have been possible.

References

[1] Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.

[2] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[5] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[10] Z. Shi, P. Siva, and T. Xiang. Transfer learning by ranking for weakly supervised object annotation.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

[12] E. W. Teh, M. Rochan, and Y. Wang. Attention networks for weakly supervised object localization.

[13] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[15] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[16] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.