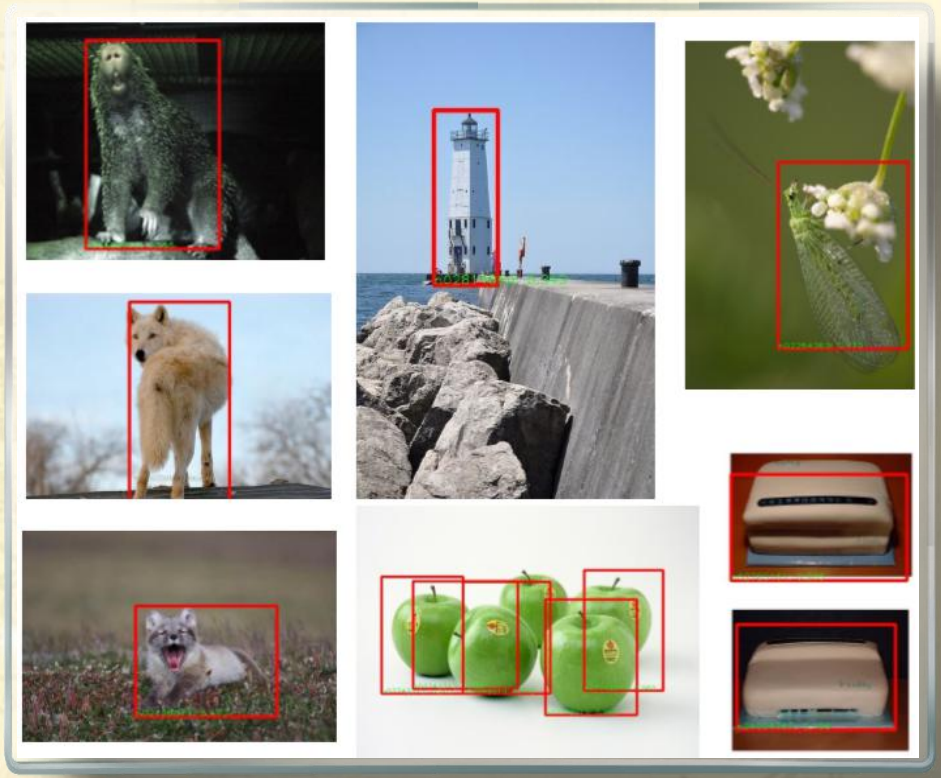
Multi Box

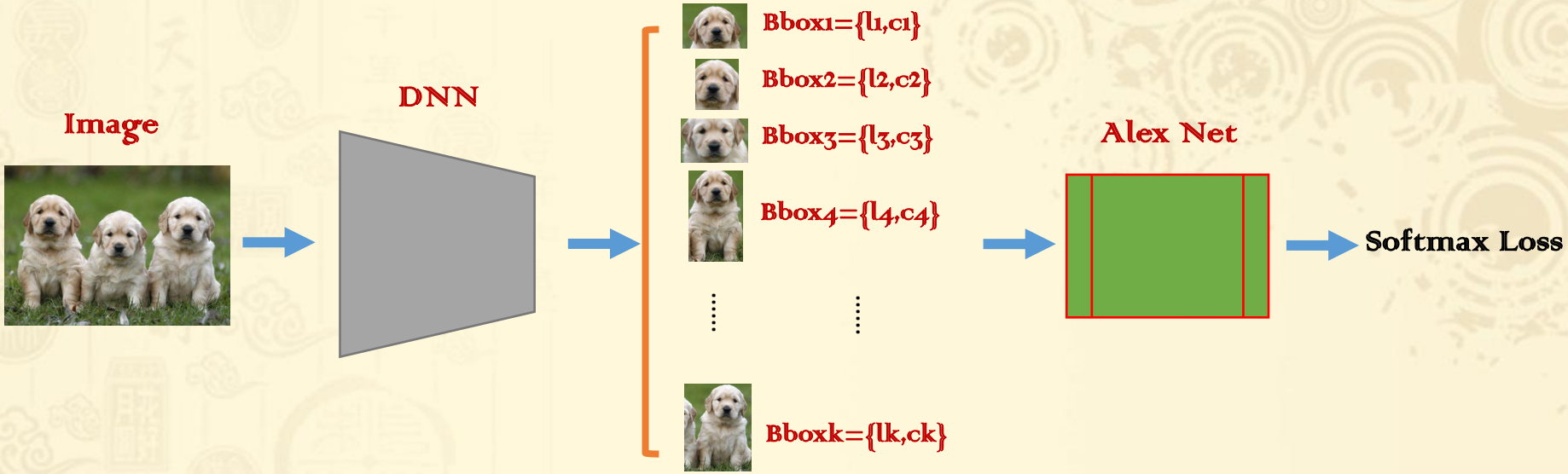
Scalable Object Detection using Deep Neural Networks

Reporter: XGH

Abstract



In this work, we propose a saliency-inspired neural network model for detection, which predicts a set of class-agnostic bounding boxes along with a single score for each box, corresponding to its likelihood of containing any object of interest. The model naturally handles a variable number of instances for each class and allows for cross class generalization at the highest levels of the network.



We aim at achieving a class-agnostic scalable object detection by predicting a set of bounding boxes, which represent potential objects. More precisely, we use a Deep Neural Network (DNN), which outputs a fixed number of bounding boxes. In addition, it outputs a score for each box expressing the network confidence of this box containing an object.

As such, they can be classified with a subsequent classifier to achieve object detection.

Deep convolutional neural networks have recently achieved state-of-the-art performance on a number of image recognition benchmarks, including the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC-2012). The winning model on the localization sub-task was a network that predicts a single bounding box and a confidence score for each object category in the image. Such a model captures the whole-image context around the objects but cannot handle multiple instances of the same object in the image without naively replicating the number of outputs for each instance.

First, we define object detection as a regression problem to the coordinates of several bounding boxes. In addition, for each predicted box the net outputs a confidence score of how likely this box contains an object.

The second major contribution is the loss, which trains the bounding box predictors as part of the network training. For each training example, we solve an assignment problem between the current predictions and the groundtruth boxes and update the matched box coordinates, their confidences and the underlying features through backpropagation.

Finally, we train our object box predictor in a class agnostic manner. We consider this as a scalable way to enable efficient detection of large number of object classes. We show in our experiments that by only post-classifying less than ten boxes, obtained by a single network application, we can achieve competitive detection results. Further, we show that our box predictor generalizes over unseen classes and as such is flexible to be re-used within other detection problems.

Mode To formalize the above idea, we encode the i-th object box and its associated confidence as node values of the last net layer:

Bounding box we encode the upper-left and lower-right coordinates of each box as four node values, which can be written as a vector $l_i \in R4$. These coordinates are normalized w.r.t. image dimensions to achieve invariance to absolute image size. Each normalized coordinate is produced by a linear transformation of the last hidden layer.

Confidence the confidence score for the box containing an object is encoded as a single node value $c_i \in [0,1]$. This value is produced through a linear transformation of the last hidden layer followed by a sigmoid.

Inference

We can combine the bounding box locations l_i, $i \in \{1,...K\}$, as one **linear layer**. Similarly, we can treat collection of all confidences c_i, $i \in \{1,...K\}$as the output as one **sigmoid layer**. Both these output layers are **connected to the last hidden layers**.

At inference time, our algorithm produces K bounding boxes. In our experiments, we use K = 100 and K = 200. If desired, we can use the **confidence scores and non-maximum suppression** to obtain a smaller number of high-confidence boxes at inference time. These boxes are supposed to represent objects. As such, they can be classified with a subsequent classifier to achieve object detection. Since the number of boxes is very small, we can afford powerful classifiers. In our experiments, we use second DNN for classification

Training

We train a DNN to predict bounding boxes and their confidence scores for each training image such that the highest scoring boxes match well the ground truth object boxes for the image. Suppose that for a particular training example, M objects were labeled by bounding boxes g_j, $j \in \{1,\ldots,M\}$. In practice, the number of predictions K is much larger than the number of groundtruth boxes M. Therefore, we try to optimize only the subset of predicted boxes which match best the ground truth ones. We optimize their locations to improve their match and maximize their confidences. At the same time we minimize the confidences of the remaining predictions, which are deemed not to localize the true objects well.

To achieve the above, we formulate an assignment problem for each training example. Let $x_{ij} \in \{0,1\}$ denote the assignment: $x_{ij} = 1$ if the i-th prediction is assigned to j-th true object. The objective of this assignment can be expressed as:

$$F_{match}(x, l) = \frac{1}{2} \sum_{i,j} x_{ij} \|l_i - gj\|_2^2$$

Additionally, we want to optimize the confidences of the boxes according to the assignment x. Maximizing the confidences of assigned predictions can be expressed as:

$$F_{conf} = -\sum_{i,j} x_{ij} \log(c_i) - \sum_i \left(1 - \sum_j x_{ij}\right) \log(1 - c_i)$$

The final loss objective combines the matching and confidence losses:

$$F(x, l, c) = \alpha F_{\text{match}}(x, l) + F_{conf}(x, c)$$

Optimization For each training example, we solve for an optimal assignment x* of predictions to true boxes by

$$x^* = \arg\min_x F(x, l, c)$$

Subject to $x_{ij} \in \{0,1\}, \sum_i x_{ij} = 1$

Then, we optimize the network parameters via back propagation. For example, the first derivatives of the back propagation algorithm are computed w. r. t. l and c:

$$\frac{\partial F}{\partial l_i} = \sum_j (l_i - g_j) x_{ij}^*$$

Training Details

HASCO

One To perform clustering of ground truth locations and find K such clusters/centroids that we can use as priors for each of the predicted locations. Thus, the learning algorithm is encouraged to learn a residual to a prior, for each of the predicted locations.

Two To using these priors in the matching process: instead of matching the N ground truth locations with the K predictions, we find the best match between the K priors and the ground truth.

Three Further, we can predict K boxes per class. Unfortunately, this model will have number of parameters growing linearly with the number of classes. Also, in a typical setting, where the number of objects for a given class is relatively small, most of these parameters will see very few training examples with a corresponding gradient contribution.

Experiment Details

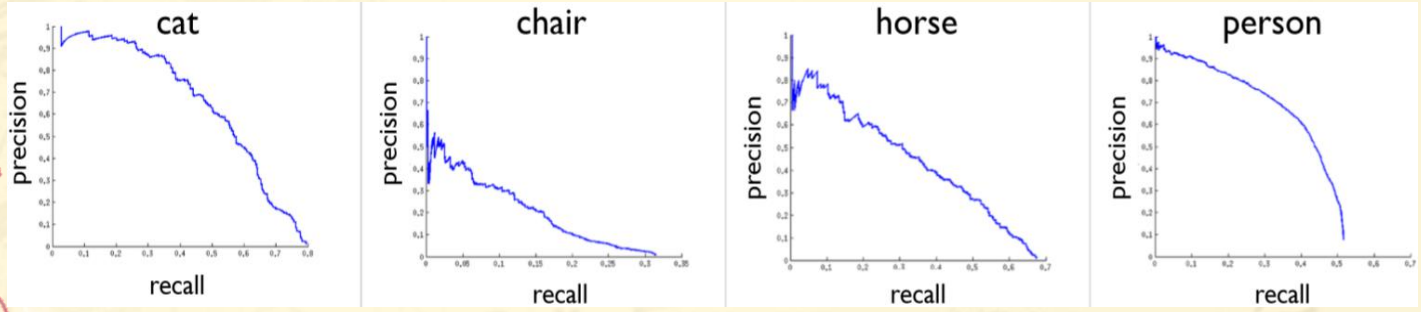
HASCO

The network architecture for the localization and classification models that we use is the same as the one used by. We use Adagrad for controlling the learning rate decay, mini-batches of size 128, and parallel distributed training with multiple identical replicas of the network, which achieves faster convergence. As mentioned previously, we use priors in the localization loss—these are computed using k-means on the training set. We also use an α of 0.3 to balance the localization and confidence losses.

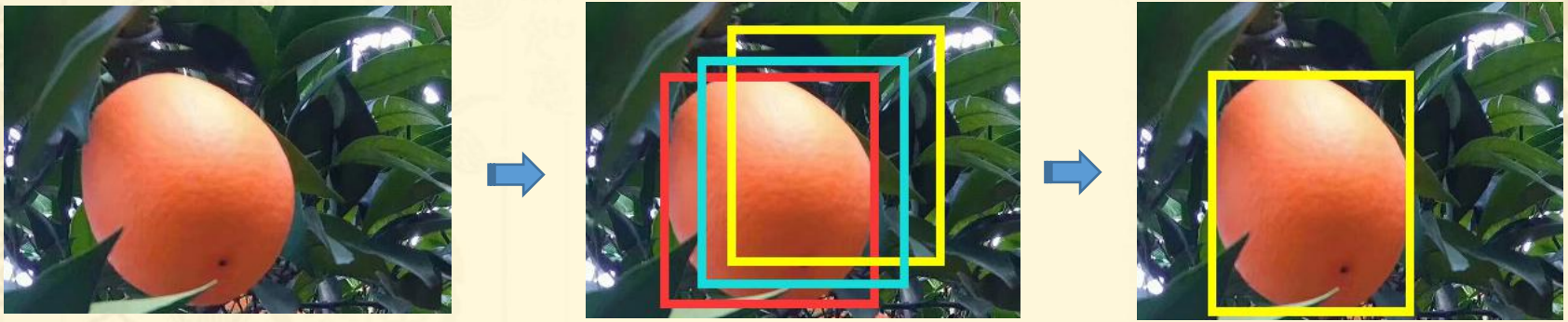
The localizer might output coordinates outside the crop area used for the inference. The coordinates are mapped and truncated to the final image area, at the end. Boxes are additionally pruned using non-maximum-suppression with a Jaccard similarity threshold of 0.5.

class	aero	bicycle	bird	boat	bottle	bus	car	cat	chair	cow
DeepMultiBox	**.413**	.277	**.305**	**.176**	.032	.454	.362	**.535**	.069	.256
3-layer model [18]	.294	.558	.094	.143	.286	.440	.513	.213	.200	.193
Felz. et al. [6]	.328	.568	.025	.168	**.285**	.397	.516	.213	.179	.185
Girshick et al. [9]	.324	**.577**	.107	.157	.253	.513	**.542**	.179	**.210**	.240
Szegedy et al. [15]	.292	.352	.194	.167	.037	**.532**	.502	.272	.102	**.348**

class	table	dog	horse	m-bike	person	plant	sheep	sofa	train	tv
DeepMultiBox	.273	**.464**	.312	.297	.375	.074	.298	.211	.436	.225
3-layer model [18]	.252	.125	.504	.384	.366	**.151**	.197	.251	.368	.393
Felz. et al. [6]	.259	.088	.492	.412	.368	.146	.162	.244	.392	.391
Girshick et al. [9]	.257	.116	**.556**	**.475**	**.435**	.145	.226	**.342**	**.442**	.413
Szegedy et al .[15]	**.302**	.282	.466	.417	.262	.103	**.328**	.268	.398	**.47**



Bounding box regression



The window is generally represented by a four-dimensional vector (x, y, w, h) ,which represents the center point coordinates and the width and height of the window, respectively. For above Figure, the yellow box P represents the original Proposal, the red box G represents the target's Ground Truth G, and our goal is to find a relationship that causes the input original window P to be mapped to a regression window that is closer to the real window G,which G^ like the green one.

Bounding box regression

The purpose of the bounding box regression is to find a mapping f for a given (P_x, P_y, P_w, P_h) so that $f(P_x, P_y, P_w, P_h) = (Gx^, Gy^, Gw^, Gh^)$ and $(Gx^,Gy^,Gw^,Gh^) \approx (Gx,Gy,Gw,Gh)$

How to do it?

First Translation (Δx, Δy) -> $\Delta x = Pw\ dx(P)$, $\Delta y = Ph\ dy(P)$

$$G^x = Pw\ dx(P) + Px$$

$$G^y = Ph\ dy(P) + Py$$

Second Scale scaling (Sw, Sh)-> $Sw = \exp(dw(P))$, $Sh = \exp(dh(P))$

$$G^w = Pw\ \exp(dw(P))$$

$$G^h = Ph\ \exp(dh(P))$$

Bounding box regression

Now ,we need to get the value of dx(P),dy(P),dw(P),dh(P) by linear regression.

What is linear regression?

Given a characteristic vector X of the input, learn a set of parameters W such that the value after linear regression is very close to the true value Y (Ground Truth). That is, $Y \approx WX$.

Input : P=(Px,Py,Pw,Ph)? No, Is the CNN Feature vector $P_{feature}$

Output : Translation transformation and scale scaling dx(P),dy(P),dw(P),dh(P) , So:

$$tx=(Gx-Px)/Pw$$

$$ty=(Gy-Py)/Ph$$

$$tw=\log(Gw/Pw)$$

$$th=\log(Gh/Ph)$$

Then the objective function can be expressed as: $d_*(P)=w_*P_{feature}$, $P_{feature}$ is the input feature vector, and w_* is the parameter to be learned (* means x, y, w, h, that is, each transform corresponds to an objective function), d*(P) is the predicted value obtained. We want the prediction value to be the smallest difference from the true value t*=(tx, ty, tw, th),so the loss function is:

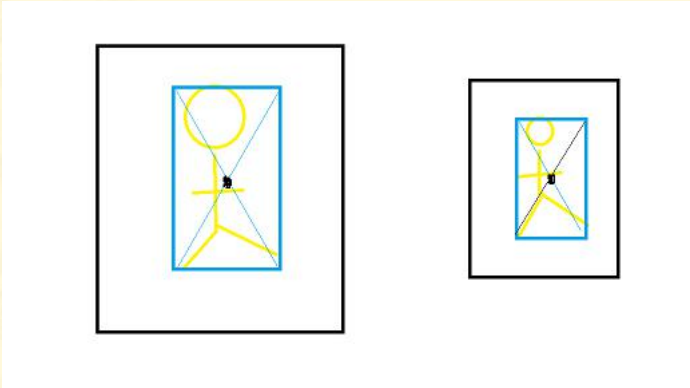
$$Loss = \sum_{i}^{N}\left(t_*^i - w_*^T P^i\right)^2$$

The function optimization goal is:

$$W_* = argmin_* \sum_i^N \left(t_*^i - w_*^T P^i\right)^2 + \lambda \|w_*\|^2$$

Some questions

One: Why does the wide and high scale design this form?



Some questions

HASCO

Two: Why do we think it is a linear transformation when IoU is large ?

$$t_w = \log(G_w/P_w) = log(\frac{G_w + P_w - P_w}{P_w}) = log(1 + \frac{G_w - P_w}{P_w})$$

G^x=Pw dx(P)+Px

G^x=(Gw-Pw)+Px

Over

THANKS

HASCO