R-CNN

Bryan Anenberg
Stanford University
anenberg@stanford.edu

Michela Meister
Stanford University
mmeister@stanford.edu

1. Introduction

Object detection is a challenging task for many reasons. For example, objects of one class may come in different shapes, colors, positions and poses. A successful system must be able to detect objects of the same class in spite of this variation, while at the same time ignoring objects outside the class. One solution to this are deformable part-based models, as described by Felzenszwalb et al [1]. Recently CNNs have become more popular for describing objects as the variation among a class can be learnt through training and does not have to be decided by a specific model.

2. Algorithm

The goal of this algorithm is to implement a detector for three different classes - cats, cars, and people. We begin with selective search bounding box proposals for each image and use a CNN to extract feature vectors from these regions. We use the features from our training set to train an SVM for each class. Then we give each feature vector from our test set a score for each class and the corresponding bounding box proposal is assigned the class for which it has the highest positive score. We augment these proposals using bounding box regression and then choose one bounding box for each object instance using bounding box regression. [2]

3. Implementation Details

3.1. Feature Extraction

We begin with a set of selective search bounding box proposals for each image, along with labeled ground truth bounding boxes for each class instance. We then crop and warp each sample to a 227x227 square sample.1 2 We perform cropping such that the output square sample has 16 pixels of context on each edge. For our training set, we classify each selective search bounding box as either a positive sample or negative sample with respect to each class. A positive sample is any bounding box with an IoU overlap close to 1 with a ground truth bounding box for that class. A negative sample is any bounding box with an IoU overlap less than .3 with a ground truth bounding box for that class. We then use a pre-trained AlexNet CNN to extract 512-dimensional feature vectors from each square sample.

3.2. SVM Training

We use the sklearn wrapper for the liblinear SVM. Determining the best parameters for the SVM is vital, and we experiment with our regularization, normalization, and bias parameters, as described later on. However we do hold some parameters constant. For each SVM we set the class weight automatically to be the inverse of the class size. This is important, because there is such a large difference in the sizes of our positive and negative sample sets. Because of the large size of our training set, we also experiment with iterative SVM training. To do this we divide our training samples randomly into batches. We begin by training the SVM on our first batch. After this initial training we use the SVM to score samples in both the first and second batches. We examine these scores and keep only the positive samples and the hard negative samples to update the SVM, where hard negative samples are samples that are either misclassified or very close to the decision boundary. We choose our hard negatives by ranking our negative samples by decreasing score and choosing the top samples. (We experiment with the best number of hard negatives to choose.) We continue by updating the SVM, then scoring the first, second, and third batches, and so on. In addition to this iterative method, we also train some SVMs on the entire dataset in one batch, without considering hard negatives.

3.3. Bounding Box Regression

The idea behind bounding box regression is that there is some class-specific transformation between the high-scoring bounding box proposals for an object instance and the ground truth for that instance. We find this transformation through training on the proposed and ground truth bounding boxes from our training set. We select pairs (P,G) of proposals and their corresponding ground truths, selecting for proposals with an IoU overlap of .6 or higher with a ground truth box of their class. From this set of pairs (P,G) we determine a target t, which describes the relation-

ship between a proposal and its ground truth. We then use ridge regression with a regularization of 1000 to find a set of vectors W 4 that maps the feature vector for the region corresponding to a proposed bounding box to its target t. During detection we then apply W to calculate a target t 3 for each proposed region and augment the proposed bounding box using the transformation given in 5.

3.4. Non-Maximal Suppression

After performing detection, there may be many bounding box proposals for each class instance in an image. We employ non-maximal suppression to choose one bounding box for each instance, using Felzenszwalbs strategy [1]. For each class in an image, we start out by ranking our proposed bounding boxes by their detection score. We choose the box with the top score, and then only choose other boxes that have an overlap of less than 0.3 with their predecessors. This ensures that only one box is chosen per instance and that multiple instances will still be recognized.

4. Experiments

We originally train our SVMs iteratively, using 16 batches and hard negative mining. We compare this method to training our SVMs on the entire data set in one batch, without considering hard negatives. We also experiment with different regularization parameters, normalizations, biases, and number of epochs run. All of our SVMs are evaluated on a validation set of 75 samples.

The following tables describe our SVM evaluations. nb indicates the number of batches used, hn indicates the number of hard negatives chosen at each update, and C gives our regularization parameter. Accuracy is determined as follows: Samples are split into positive and negative groups based on overlap with ground truth bounding boxes. If a sample overlaps with a ground truth bounding box for class x by more than 0.6 IoU, for the sake of this accuracy measurement it is considered a positive sample for class x. Negative samples for class x are those that have less than 0.6 IoU overlap with a ground truth bounding box for class x. pos-acc-x gives the systems accuracy at labeling these positive samples for class x.

5. Results

5.1. SVM Training

Figures 6-9 demonstrate SVMs that did not work well at all. From Figure 7 we see that it is clearly important to normalize features. Even when an SVM for one class worked well, accuracies in other classes was low. This is probably because one SVM is assigning high scores to everything, which means that objects of that class are detected well, but objects of all the other classes are misclassified. However, training SVMs on the full data set (10-12) without hard negative mining gives decent results, and these results are even improved when we loosen our requirements for positive bounding boxes. Specifically, we allow the positive bounding boxes to be those boxes with greater than 0.8 Intersection of Union overlap with the ground truth bounding boxes of that class.

5.2. Final results

After training the relaxed SVM on the full training data set and evaluating the performance on the test data set we achieved performance of: average precision$_c lass1$: $0.0089 average_p recision_c lass2$: $0.0096 average_p recision_c lass3 : 0.0070$

The poor performance of the model could be due to the difficulties in training the SVMs.

6. Conclusion

With more time, we would try to further optimize our SVM training, as it seems that the SVM is the crux of the system. We would also integrate the entire pipeline in order to achieve detection results. It was very difficult in this project to unit test any one portion of the system, and in the future we plan to test our individual sections early and often.

Figure 1

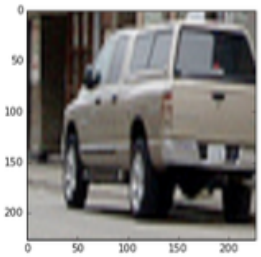

Figure 2

$$
\begin{aligned}
t_x &= (G_x - P_x)/P_w \\
t_y &= (G_y - P_y)/P_h \\
t_w &= \log(G_w/P_w) \\
t_h &= \log(G_h/P_h).
\end{aligned}
$$

Figure 3

$$
\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2 .
$$

Figure 4

$$
\begin{aligned}
\hat{G}_x &= P_w d_x(P) + P_x \\
\hat{G}_y &= P_h d_y(P) + P_y \\
\hat{G}_w &= P_w \exp(d_w(P)) \\
\hat{G}_h &= P_h \exp(d_h(P)).
\end{aligned}
$$

Figure 5

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	16	10000	0.001	1	True	0.000000	0.240848	0.504098	0.009670
1	1	16	10000	0.010	1	True	0.000000	0.231214	0.508197	0.003979
2	1	16	10000	0.100	1	True	0.000000	0.229287	0.508197	0.003586
3	1	16	10000	1.000	1	True	0.003802	0.233141	0.504098	0.003744
4	1	16	10000	10.000	1	True	0.370722	0.208092	0.225410	0.005884
5	1	16	10000	100.000	1	True	0.669202	0.177264	0.032787	0.020985
6	1	16	10000	1000.000	1	True	0.545627	0.144509	0.016393	0.033311

Figure 6: Varied the SVM regularization parameter C

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	16	10000	0.1	1	True	0	0	1	0
1	1	16	10000	1.0	1	True	0	1	0	0
2	1	16	10000	10.0	1	True	0	0	1	0
3	1	16	10000	100.0	1	True	0	0	1	0

Figure 7: Trained SVM without normalizing the features

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	16	10000	1	1	True	0.003802	0.233141	0.504098	0.003744
1	1	16	10000	1	10	True	0.007605	0.231214	0.508197	0.005637
2	1	16	10000	1	100	True	0.000000	0.000000	0.000000	1.000000
3	1	16	10000	1	10000	True	0.000000	0.000000	0.000000	1.000000

Figure 8: Trained SVMs varying the bias

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	16	10000	1	1	True	0.003802	0.233141	0.504098	0.003744
1	3	16	10000	1	1	True	0.315589	0.171484	0.479508	0.000370

Figure 9: Trained SVM over multiple epochs of the data set.

	epochs	nb	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	1	1	1	True	0.315589	0.905588	0.532787	0.489602
1	1	1	10	1	True	0.300380	0.936416	0.516393	0.530349
2	1	1	1	10	True	0.313688	0.905588	0.532787	0.512779
3	1	1	50	10	True	0.230038	0.023121	0.200820	0.993710

Figure 10: Trained the SVM on the full data set without hard negative mining.

	epochs	nb	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	1	1	1	True	0.315589	0.905588	0.532787	0.489602
1	1	1	10	1	True	0.300380	0.936416	0.516393	0.530349
2	1	1	1	10	True	0.313688	0.905588	0.532787	0.512779
3	1	1	50	10	True	0.230038	0.023121	0.200820	0.993710

Figure 11: Trained the SVM on the full data set without hard negative mining.

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	1	10000	50	1	True	0.395437	0.934489	0.516393	0.606218
1	1	1	10000	1	10	True	0.298479	0.942197	0.491803	0.555901

Figure 12: Trained the SVM on the full data and consider positives as those bounding boxes with greater than or equal to 0.8 IoU with ground truth bounding box of that class.

	epochs	nb	hn	C	bias	L2-norm	pos_acc_1	pos_acc_2	pos_acc_3	neg_acc
0	1	1	10000	50	1	True	0.395437	0.934489	0.516393	0.606218
1	1	1	10000	1	10	True	0.298479	0.942197	0.491803	0.555901

Figure 13: Trained the SVM on the full data and consider positives as those bounding boxes with greater than or equal to 0.8 IoU with ground truth bounding box of that class.

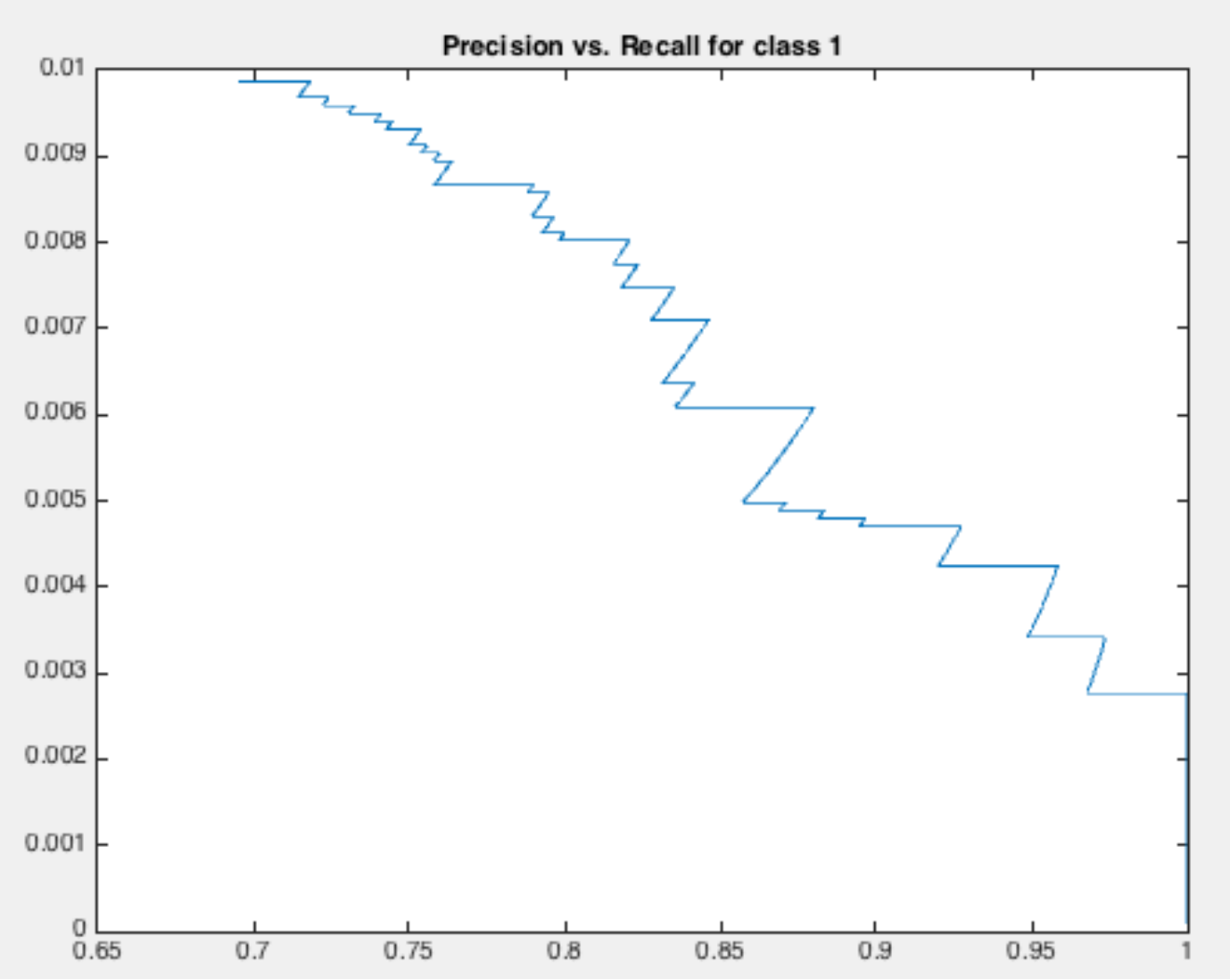

Figure 14: Precision Recall curve for class 1.

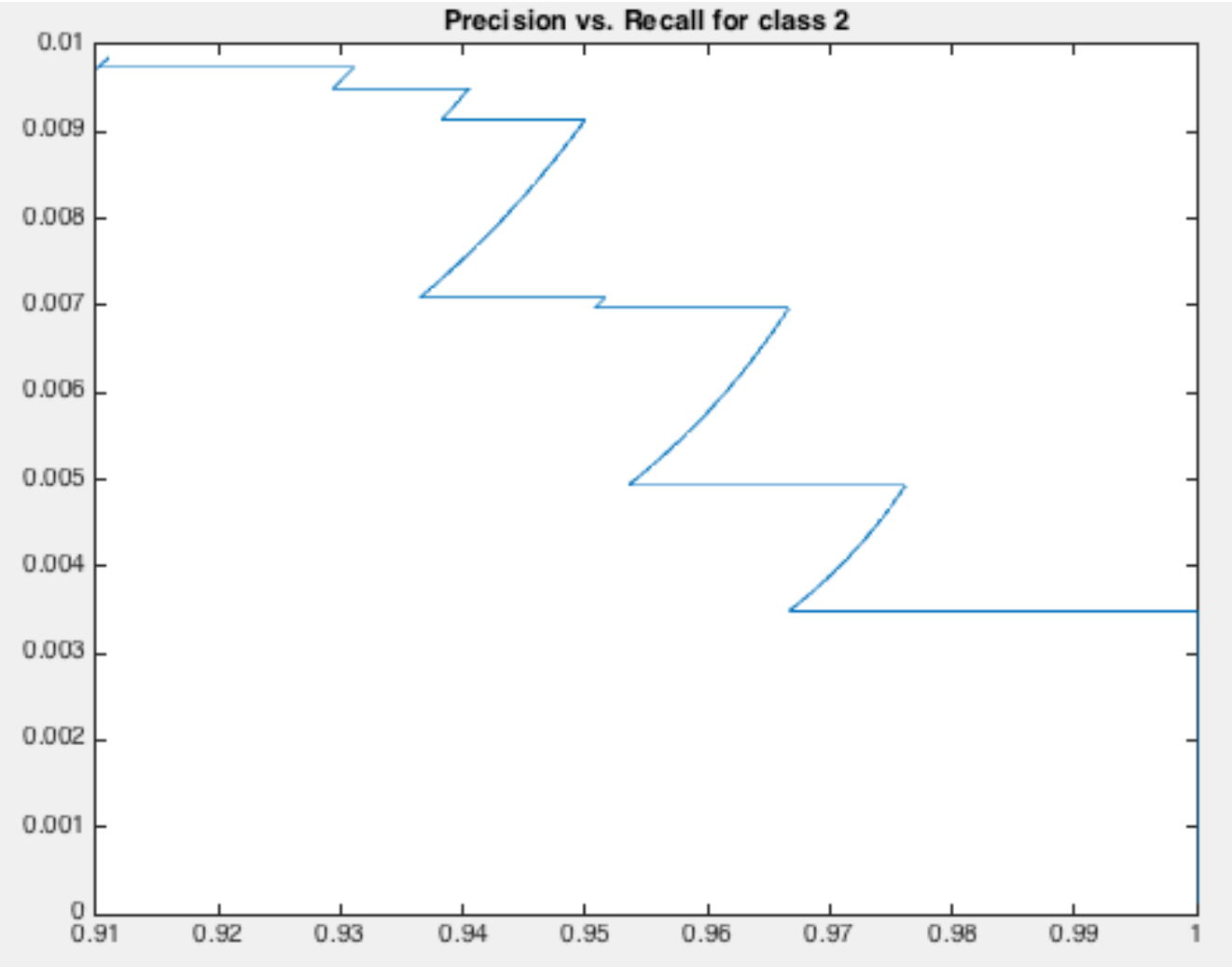

Figure 15: Precision recall curve for class 2.

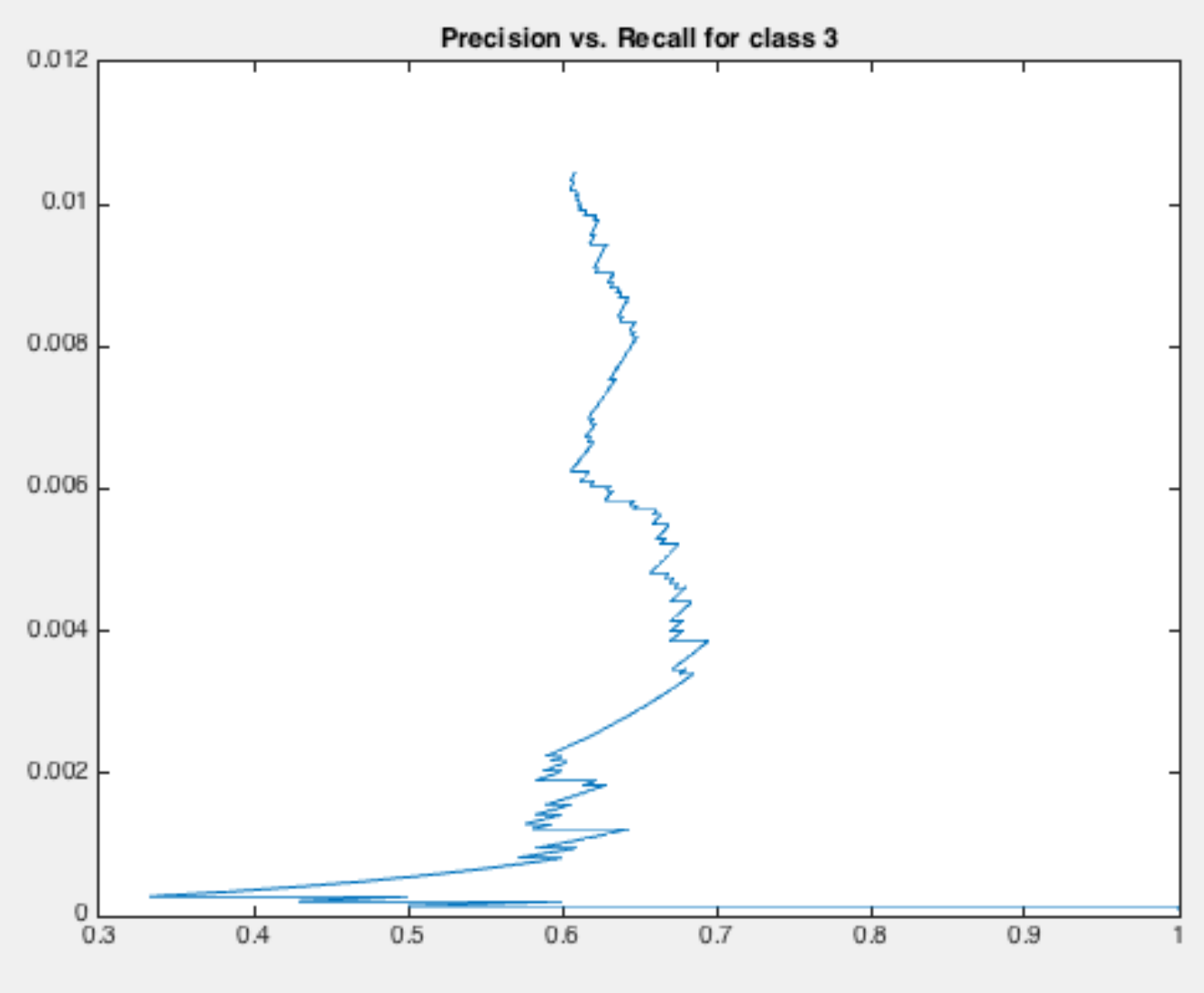

Figure 16: Precision recall curve for class 3.

References

[1] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

[2] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.