
End-to-end people detection in crowded scenes

Russell Stewart¹ Mykhaylo Andriluka^{1,2}

¹Department of Computer Science, Stanford University, USA

²Max Planck Institute for Informatics, Saarbrücken, Germany
{stewartr, andriluk}@stanford.edu

Abstract

Current people detectors operate either by scanning an image in a sliding window fashion or by classifying a discrete set of proposals. We propose a model that is based on *decoding* an image into a set of people detections. Our system takes an image as input and directly outputs a set of distinct detection hypotheses. Because we generate predictions jointly, common post-processing steps such as non-maximum suppression are unnecessary. We use a recurrent LSTM layer for sequence generation and train our model end-to-end with a new loss function that operates on sets of detections. We demonstrate the effectiveness of our approach on the challenging task of detecting people in crowded scenes.

1 Introduction

In this paper we propose a new architecture for detecting objects in images. We strive for an end-to-end approach that accepts images as input and directly generates a set of object bounding boxes as output. This task is challenging because it demands both distinguishing objects from the background and correctly estimating the number of distinct objects and their locations. Such an end-to-end approach capable of directly outputting predictions would be advantageous over methods that first generate a set of bounding boxes, evaluate them with a classifier, and then perform some form of merging or non-maximum suppression on an overcomplete set of detections.

Generating a set of detections from an integrated process has an important advantage in that multiple detections on the same object can be avoided by remembering the previously generated output. To control this generation process, we use a recurrent neural network with LSTM units. To produce intermediate representations, we use expressive image features from GoogLeNet that are further fine-tuned as part of our system. Our architecture can thus be seen as a “decoding” process that converts an intermediate representation of an image into a set of predicted objects. The LSTM can be seen as a “controller” that propagates information between decoding steps and controls the location of the next output (see Fig. 2 for an overview). Importantly, our trainable end-to-end system allows joint tuning of all components via back-propagation.

One of the key limitations of merging and non-maximum suppression utilized in [5, 15] is that these methods typically don’t have access to image information, and instead must perform inference solely based on properties of bounding boxes (e.g. distance and overlap). This usually works for isolated objects, but often fails when object instances overlap. In the case of overlapping instances, image information is necessary to decide where to place boxes and how many of them to output. As a workaround, several approaches proposed specialized solutions that specifically address pre-defined constellations of objects (e.g. pairs of pedestrians) [4, 19]. Here, we propose a generic architecture that does not require a specialized definition of object constellations, is not limited to pairs of objects, and is fully trainable.

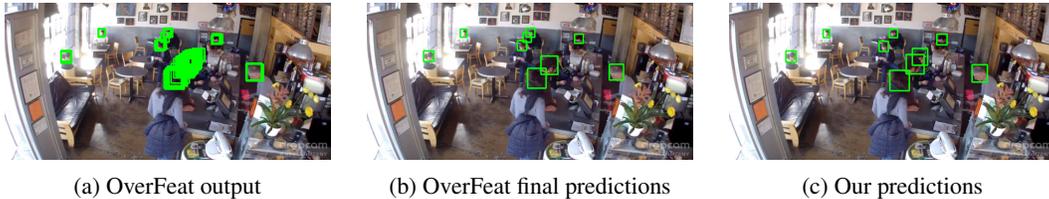


Figure 1: Initial over-complete set of detections of OverFeat (a) and output of post-processing (b). Note the failure to detect the third person in the center. Detection results obtained with our method (c).

We specifically focus on the task of people detection as an important example of this problem. In crowded scenes such as the one shown in Fig. 1, multiple people often occur in close proximity, making it particularly challenging to distinguish between nearby individuals.

The key contribution of this paper is a trainable, end-to-end approach that jointly predicts the objects in an image. This lies in contrast to existing methods that treat prediction or classification of each bounding box as an independent problem and require post-processing on the set of detections. We demonstrate that our approach is superior to existing architectures on a challenging dataset of crowded scenes with large numbers of people. A technical contribution of this paper is a novel loss function for sets of objects that combines elements of localization and detection. Another technical contribution is to show that a chain of LSTM units can be successfully utilized to decode image content into a coherent real-valued output of variable length. We envision this technique to be valuable in other structured computer vision prediction tasks such as multi-person tracking and articulated pose estimation of multiple people.

1.1 Related work

Detection of multiple objects in the presence of occlusions has been a notorious problem in computer vision. Early work employed a codebook of local features and Hough voting [12, 2], but still required complex tuning and multi-stage pipelines. Importantly, these models utilized weak representations based on local features that are outperformed by modern deep representations.

To overcome the difficulties of predicting multiple objects in close proximity, several attempts have been made to jointly predict constellations of objects [4, 19, 14]. Our work is more general, as we do not explicitly define these groups, and instead let the model learn any features that are necessary for finding occluded instances.

Currently, the best performing object detectors operate either by densely scanning the image in a sliding window fashion [15, 5, 21], or by using a proposal mechanism such as [20, 18], and leveraging CNNs to classify a sparsified set of proposals [5]. These approaches work well for images with few object instances that do not overlap, but often fail in the presence of strong occlusions. For example, MultiBox [18] learns class independent proposals that are subsequently classified with CNN. Like MultiBox, we propose a set of bounding boxes from images, but these proposals directly correspond to object instances and do not require post-processing. The MultiBox outputs are necessarily sparse, whereas our system is able to generate predictions in arbitrarily close proximity.

Our approach is related to the OverFeat model [15]. We rely on a regression module to generate boxes from a CNN encoding. However, in our case boxes are generated as part of an integrated process, and not independently as in OverFeat. As a result, each output box corresponds directly to an object detected in the image, and we do not require merging or non maximum suppression. Another important advantage of our approach is that it outputs a confidence corresponding to each output that is trained end-to-end. In the case of OverFeat, an end-to-end trained confidence prediction is not available, as the output is the result of a heuristic merging procedure.

Our work is related to recent neural network models for predicting sequences [10, 16]. As in [16], we rely on an LSTM to predict variable length outputs. Unlike in language generation, detection requires that a system reason over a 2D output space, which lacks a natural linear ordering. MultiBox [18] addresses this challenge by introducing a loss function that allows unordered predictions to be

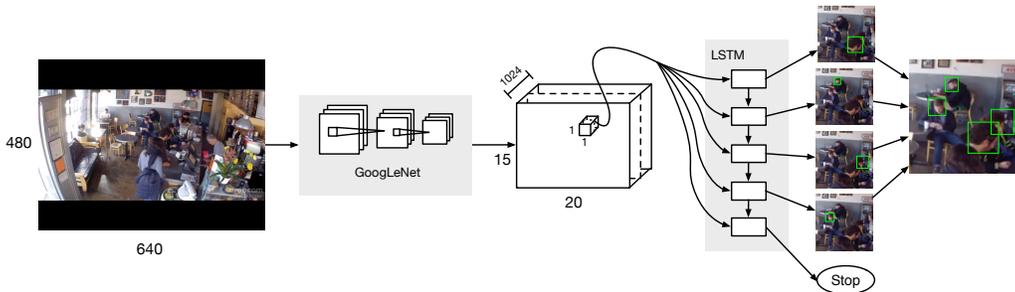


Figure 2: Our system first encodes an image into a block of high level features. An LSTM then acts as a controller, decoding this information into a set of detections.

permuted to match ground-truth instances during training. We build on this contribution by leveraging the capacity of our recurrent decoder to make joint predictions in sequence. In addition to computing an optimal matching of predictions to ground-truth, our loss function encourages the model to make predictions in order of descending confidence. Suitable loss functions have previously been proposed in structured speech recognition and natural language processing [6]. Here we propose such a loss function for object detection.

2 Model

2.1 Overview

Deep convolutional architectures such as [11, 17] construct image representations that are effective for a variety of tasks. These architectures have been leveraged for detection, albeit primarily by adapting them into a classification or regression framework. Deep representations have sufficient power to jointly encode the appearance of multiple instances, but one must augment them with a component for multiple instance prediction to realize this potential. In this paper, we consider recurrent neural networks (RNN), and in particular LSTM units [7] as a candidate for such a component. The key properties that make the combination of deep CNN’s with RNN-based decoders appealing are (1) the ability to directly tap into powerful deep convolutional representations and (2) the ability to generate coherent sets of predictions of variable length. These properties have been leveraged successfully in [10] to generate image captions, and in [16] for machine translation. The ability to generate coherent sets is particularly important in our case because our system needs to remember previously generated predictions and avoid multiple predictions of the same target.

We construct a model that first encodes an image into high level descriptors via a convolutional architecture (e.g. [17]), and then decodes that representation into a set of bounding boxes. As a core machinery for predicting variable length output, we build on a recurring network of LSTM units. An overview of our model is shown on Fig. 2. We transform each image into a grid of 1024 dimensional feature descriptors at strided regions throughout the image. The 1024 dimensional vector summarizes the contents of the region and carries rich information regarding the positions of objects. The LSTM draws from this information source and acts as a controller in the decoding of a region. At each step, the LSTM outputs a new bounding box and a corresponding confidence that a previously undetected person will be found at that location. Boxes are encouraged to be produced in order of descending confidence. When the LSTM is unable to find another box in the region with a confidence above a prespecified threshold, a stop symbol is produced. The sequence of outputs is collected and presented as a final description of all object instances in the region.

2.2 Loss function

The architecture introduced in Sec. 2.1 predicts a set of candidate bounding boxes along with a confidence score corresponding to each box. Hypotheses are generated in sequence and later predictions depend on previous ones via the memory states of the LSTM. At each recurrence, the LSTM outputs an object bounding box $\mathbf{b} = \{\mathbf{b}_{pos}, b_c\}$, where $\mathbf{b}_{pos} = (b_x, b_y, b_w, b_h) \in \mathbb{R}^4$ is a relative

position, width and height of the bounding box, and $b_c \in [0, 1]$ is a real-valued confidence. Confidence values lower than a pre-specified threshold (e.g. 0.5) will be interpreted as a stop symbol at test time. Higher values of the bounding box confidence b_c should indicate that the box is more likely to correspond to a true positive. We denote the corresponding set of ground truth bounding boxes as $G = \{\mathbf{b}^i | i = 1, \dots, M\}$, and the set of candidate bounding boxes generated by the model as $C = \{\tilde{\mathbf{b}}^j | j = 1, \dots, N\}$. In the following we introduce a loss function suitable for guiding the learning process towards the desired output.

Consider the example in Fig. 3, which schematically shows a detector with four generated hypotheses, each numbered by its prediction step, which we denote as rank. Note the typical detection mistakes such as false positives (hypothesis 3), imprecise localizations (hypothesis 1), and multiple predictions of the same ground-truth instance (hypotheses 1 and 2). Different mistakes require different kinds of feedback. In the case of hypothesis 1, the box location must be fine-tuned. Conversely, hypothesis 3 is a false positive, and the model should instead abandon the prediction by assigning a low confidence score. Hypothesis 2 is a second prediction on the target already reported by hypothesis 1, and should be abandoned as well. To capture these relationships, we introduce a matching algorithm that assigns a unique candidate hypothesis to each ground-truth. The algorithm returns an injective function $f : G \rightarrow C$, i.e. $f(i)$ is the index of candidate hypothesis assigned to ground-truth hypothesis i .

Given f , we define a loss function on pairs of sets G and C as

$$L(G, C, f) = \alpha \sum_{i=1}^{|G|} l_{pos}(\mathbf{b}_{pos}^i, \tilde{\mathbf{b}}_{pos}^{f(i)}) + \sum_{j=1}^{|C|} l_c(\tilde{\mathbf{b}}^j, y_j) \quad (1)$$

where $l_{pos} = \|\mathbf{b}_{pos}^i - \tilde{\mathbf{b}}_{pos}^{f(i)}\|_1$ is a displacement between the position of ground-truth and candidate hypotheses, and l_c is a cross-entropy loss on a candidate’s confidence that it would be matched to a ground-truth. The label for this cross-entropy loss is provided by y_j . It is defined from the matching function as $y_j = \mathbb{1}\{f^{-1}(j) \neq \emptyset\}$. α is a term trading off between confidence errors and localization errors. We set $\alpha = 0.03$ with cross validation. Note that for a fixed matching, we can update the network by backpropagating the gradient of this loss function.

As an naïve baseline, we consider a simple matching strategy based on the fixed ordering of the ground-truth bounding boxes. We sort ground-truth boxes by image position from top to bottom and from left to right. This fixed order matching sequentially assigns candidates to the sorted ground-truth. We refer to this matching function as “fixed order” matching, denoting it as f_{fix} , and the corresponding loss function as L_{fix} .

Hungarian loss: The limitation of the fixed order matching is that it might incorrectly assign candidate hypotheses to ground-truth instances when the decoding process produces false positives or false negatives. This issue persists for any specific ordering chosen by f_{fix} . We thus explore loss functions that consider all possible one-to-one assignments between elements in C and G .

Recall that one of the principled objectives of our model is to output a coherent sequence of predictions on multiple objects. We define the stopping criterion for the generation process to be when a prediction score falls below a specified threshold. For such a score threshold to make sense, we must encourage the model to generate correct hypotheses early in the sequence, and to avoid generating low-confidence predictions before high-confidence ones. Therefore, when two hypotheses both significantly overlap the same ground-truth (e.g. hypotheses 1 and 2 in Fig. 3), we prefer to match the hypothesis that appears earlier in the predicted sequence.

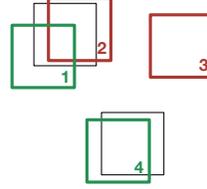


Figure 3: Illustration of the matching of ground-truth instances (black) to accepted (green) and rejected (red) candidates. Matching should respect both precedence (1 vs 2) and localization (4 vs 3).

To formalize this notion, we introduce the following comparison function between hypotheses and ground-truth:

$$\Delta(\mathbf{b}_i, \tilde{\mathbf{b}}_j) = (o_{ij}, r_i, d_{ij}) \quad (2)$$

The function $\Delta : G \times C \rightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{R}$ returns a tuple where d_{ij} is the L_1 distance between bounding box locations, r_j is the rank or index of $\tilde{\mathbf{b}}_j$ in the prediction sequence output by the LSTM, and $o_{ij} \in \{0, 1\}$ is a variable penalizing hypotheses that do not sufficiently overlap a ground-truth instance. Here, the overlapping criterion requires that a candidate’s center lie within the extent of the ground-truth bounding box. The o_{ij} variable makes an explicit distinction between localization and detection errors. We define a lexicographic ordering on tuples produced by Δ . That is, when evaluating which of two hypotheses will be assigned to a ground-truth, overlap is paramount, followed by rank and then fine-grained localization.

Given the definition of the comparison function Δ in Eq.2, we find the minimal cost bipartite matching between C and G in polynomial time via the Hungarian algorithm. Note that the Hungarian algorithm is applicable to any graph with edge weights that have well-defined addition and pairwise comparison operations. To that end, we define $(+)$ as element-wise addition and $(<)$ as lexicographic comparison. For the example in Fig. 3, correctly matching hypotheses 1 and 4 would cost $(0, 5, 0.4)$, whereas matching 1 and 3 would cost $(1, 4, 2.3)$, and matching 2 and 4 would cost $(0, 6, 0.2)$. Note how the first term, used for detecting overlap, properly handles the case where a hypothesis has low rank, but is too far from the ground-truth to be a sensible match (as is the case for hypothesis 3 in Fig. 3). We refer to the corresponding loss for this matching as the Hungarian loss and denote it as L_{hung} .

We also consider a simplified version of L_{hung} where only the top $k = |G|$ ranked predictions from C are considered for matching. Note that this is equivalent to removing or zeroing out the pairwise matching terms o_{ij} in Eq. 2. We denote this loss as L_{firstk} . We experimentally compare L_{fix} , L_{firstk} , and L_{hung} in Sec. 4, showing that L_{hung} leads to best results.

3 Implementation details

We constructed our model to encode an image into a 15x20 grid of 1024-dimensional top level GoogLeNet features. Each cell in the grid has a receptive field of size 139x139, and is trained to produce a set of distinct bounding boxes in the center 64x64 region. The 64x64 size was chosen to be large enough to capture challenging local occlusion interactions. Larger regions may also be used, but provide little additional on our scenes, where few occlusion interactions span that scale. 300 distinct LSTM controllers are run in parallel, one for each 1x1x1024 cell of the grid.

Our LSTM units have 250 memory states, no bias terms, and no output nonlinearities. At each step, we concatenate the GoogLeNet features with the output of the previous LSTM unit, and feed the result into the next LSTM unit. We have produced comparable results by only feeding the image into the first LSTM unit, indicating that multiple presentations of the image may not be necessary. Producing each region of the full 480x640 image in parallel gives an efficient batching of the decoding process.

Our model must learn to regress on bounding box locations through the LSTM decoder. During training, the decoder outputs an overcomplete set of bounding boxes, each with a corresponding confidence. For simplicity and batching efficiency, the cardinality of the overcomplete set is fixed, regardless of the number of ground-truth boxes. This trains the LSTM to output high confidence scores and correct localizations for boxes corresponding to the ground truth, and low confidence scores elsewhere. Because early outputs are preferred during matching, the model learns to output high confidence, easy boxes first. In our dataset, few regions have more than 4 instances, and we limit the overcomplete set to 5 predictions. Larger numbers of predictions neither improved nor degraded performance.

After sequence generation, ground-truth instances are matched to predictions using the bipartite matching function described in 2.2, which favors candidates that are output earlier and closer to the ground-truth targets. With an optimal matching determined, we backpropagate the Hungarian loss described in equation 1 through the full network.



Figure 4: Example of stitching in a new region’s predictions (red) with accepted predictions (green).

Model training: We use the Caffe open source deep learning framework [9] for training and evaluation. The decoder portion of our model is a custom LSTM implementation derived from NLPCaffe [1]. We train with learning rate $\epsilon = 0.2$ and momentum 0.5. Gradients are clipped to have maximum 2-norm of 0.1 across the network. We decreased the learning rate by a multiple of 0.8 every 100,000 iterations. Convergence is reached at 500,000 iterations.

Training proceeds on all subregions of one image at each iteration. Parallelism of the LSTM decoders across regions mitigates efficiency gains for larger batch sizes. All weights are tied between regions and LSTM steps. However, we were surprised to find slight performance gains when using separate weights connecting LSTM outputs to predicted candidates at each step. These weights remain tied across regions. Tying these weights reduced average precision (AP) from .85 to .82. All hyperparameter AP analysis is performed on the validation set.

Initialization: GoogLeNet weights are initialized to a snapshot after 10,000 iterations of training from our baseline OverFeat model (which was in turn initialized with Imagenet pretraining). Fine tuning of GoogLeNet features to meet the new demands of the decoder is critical. Training without fine tuning GoogLeNet reduced AP by .29.

All weights in the decoder are initialized from a uniform distribution in $[-0.1, 0.1]$. However, these initializations differ drastically from our pretrained GoogLeNet, which has activations in the range $[-80, 80]$. To compensate for this mismatch, we use a scale layer to decrease GoogLeNet activations by a factor of 100 before feeding them into the LSTM. Likewise, the initial standard deviation of the fully connected layers output is on the order of 0.3, but bounding box pixel locations and sizes vary in $[-64, 64]$. Thus, we scale up the regression predictions by a factor of 100 before comparing them with ground-truth. Note that these modifications are the same as changing weight initializations only if one also introduces proportional learning rate multipliers.

Regularization: We use dropout with probability 0.15 on the output of each LSTM. Removing dropout reduces AP by .011. Images are jittered by up to 32 pixels in the vertical and horizontal directions, and are scaled at random by a factor in $[0.9, 1.1]$. We found it important to remove L_2 regularization entirely. When applying the original L_2 regularization multiplier of $2e-4$ to only the GoogLeNet section of our network, we were unable to train. Multipliers as small as $1e-6$ reduced AP by .03.

Stitching: While we trained our algorithm to predict bounding boxes on 64×64 pixel regions, we apply our algorithm to full 480×640 images at test time. To this end, we generate predictions from each region in a 15×20 grid of the image. We then use a stitching algorithm to recursively merge in predictions from successive cells on the grid. Each iteration of the algorithm performs the process illustrated in Figure 4. At a given iteration, let A denote the current set of all accepted bounding box predictions. We process a new region, evaluating the decoder until a stop symbol is produced and collect a set C of newly proposed bounding boxes. Some of these new bounding boxes may correspond to previous predictions. Thus to avoid adding false positives, we destroy any new boxes having nonzero intersection with accepted boxes, conditioned on the constraint that previously accepted boxes may destroy at most one new box.

To this end, we define a bipartite matching problem related to that in section 2.2 with a pairwise loss term $\Delta' : A \times C \rightarrow \mathbb{N} \times \mathbb{R}$ given as $\Delta'(\mathbf{b}_i, \tilde{\mathbf{b}}_j) = (m_{ij}, d_{ij})$. Here, m_{ij} states whether two boxes do not intersect, and d_{ij} is a local disambiguation term given by the L_1 distance between boxes. Minimizing this matching cost will maximize the number of destroyed candidates. As before, we

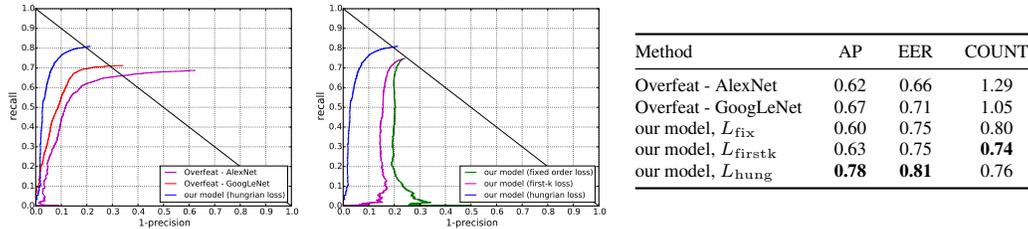


Figure 5: Performance evaluation.

leverage the Hungarian algorithm to find a minimum cost matching in polynomial time. We examine each match pair, $(\mathbf{b}, \hat{\mathbf{b}})$, and add any candidate $\hat{\mathbf{b}}$ that does not overlap with its match \mathbf{b} to the set of accepted boxes.

4 Experimental results

Dataset and evaluation metrics: To evaluate the performance of our approach, we collected a large dataset of images from busy scenes using video footage available from public webcams. In total, we collect 11917 images with 91146 labeled people. We extract images from video footage at a fixed interval of 100 seconds to ensure a large variation in images. We allocate 1000 images for testing and validation, and leave the remaining images for training, making sure that no temporal overlaps exist between training and test splits. The resulting training set contains 82906 instances. Test and validation sets contain 4922 and 3318 people instances respectively. Images were labeled using Amazon Mechanical Turk by a handful of workers pre-selected through their performance on an example task. We label each person’s head to avoid ambiguity in bounding box locations. The annotator labels any person she is able to recognize, even if a substantial part of the person is not visible. Images and annotations will be made available¹.

Examples of collected images are shown in Fig. 6, and in the video included in the supplemental material. Images in our dataset include challenges such as people at small scales, strong partial occlusions, and a large variability in clothing and appearance. Our evaluation uses the standard protocol defined in [3]. A hypothesis is considered correct if its intersection-over-union score with a ground-truth bounding box is larger than 0.5. We plot recall-precision curves and summarize results in each experiment with average precision (AP) and equal error rate (EER) in Fig. 5. We also analyze how well each model predicts the total count of people in an image. As in [13], we measure count error by computing the average absolute difference between the number of predicted and ground-truth detections in test set images. For each model, an optimal detection threshold is selected on the validation set, and we report the results as COUNT in Fig. 5.

Baseline methods: We experimented with R-CNN [5] and OverFeat [15] models to define a strong baseline in our comparison. We use the implementation of OverFeat for the Caffe framework [9] that has been made publicly available by [8], and R-CNN provided by [5]. In the initial experiments, R-CNN did not appear suitable for our setting primarily because the bottom-up proposals generated with selective search [20] achieved poor recall. We therefore use OverFeat in the rest of the evaluation. The original version of OverFeat provided by [8] relied on the image representation based on AlexNet [11]. We hence refer to the original version as OverFeat-AlexNet. Since both OverFeat and our model are implemented in Caffe, we were able to directly substitute the GoogLeNet architecture into the OverFeat model. We denote the new model as OverFeat-GoogLeNet. The comparison of the two OverFeat variants is shown in Fig. 5. We observe that OverFeat-GoogLeNet performs significantly better than OverFeat-AlexNet.

Performance evaluation: Note that the image representation used in our model and in OverFeat is exactly the same. Both are implemented using the same code, parameters, filter dimensions, and number of filters. This gives us the interesting possibility of directly comparing the hypothesis generating components of both models. In the case of OverFeat [15], this component corresponds

¹<http://d2.mpi-inf.mpg.de/datasets>

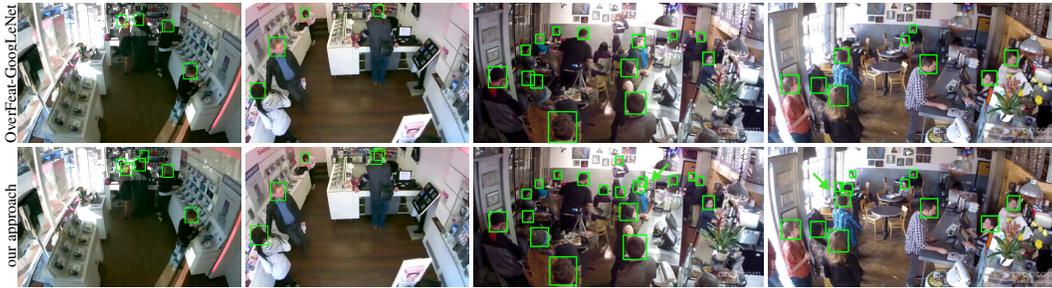


Figure 6: Example detection results obtained with OverFeat-GoogLeNet (top row) and our approach (bottom row). We show each model’s output at 90% precision. See the text for a description and the supplemental material for a video² of our model’s output.



Figure 7: Example failure cases of our method.

to a bounding box regression from each cell followed by a round of non-maximum suppression. In our model this component corresponds to decoding with an LSTM layer that produces a variable length output. The performance of our best model is shown Fig. 5 and compared to both versions of OverFeat. Our approach delivers a substantial improvement over OverFeat, improving recall from 71% to 81%. We also achieve considerable improvement in AP (0.78 for our model vs. 0.67 for OverFeat-GoogLeNet), and people counting error (0.76 vs. 1.05). Fig. 6 shows several examples of detections obtained by our model and OverFeat-GoogLeNet. The green arrows highlight cases where our model can detect people even in the presence of strong occlusions. Examples of typical failure cases are indicated by red arrows in Fig. 7.

Comparison of loss functions. We now evaluate the loss functions introduced in Sec. 2.2. The model trained with L_{fix} achieves only 0.60 AP. This suggests that allowing the LSTM to output detections from easy to hard during training, rather than in some fixed spatial ordering, was essential for performance. To explore the importance of overlap terms in our loss function, we evaluate the L_{firstk} loss, which matches the k ground-truth instances in each region to the first k output predictions. We observe that L_{firstk} outperforms L_{fix} at test time by allowing permutations of LSTM outputs during training. However, we found that L_{firstk} struggled to attach confidences to specific box locations. With L_{firstk} , early confidence predictions are often too high, and late predictions too low. It appears that instead of learning the probability that the corresponding box is correct, the model learns on the i^{th} recurrent step to predict the confidence that there are at least i people in a region. These confidences are inappropriate for detection thresholding, and underscore the importance of including the overlap terms, o_{ij} , in our matching function. Precision recall curves for each loss function are shown in Fig. 5.

5 Conclusion

In this paper, we introduced a new method for object detection and demonstrated its performance on a large dataset of images of crowded scenes. Our system addresses the challenge of detecting multiple partially occluded instances by decoding a variable number of outputs from rich intermediate representations of an image. To teach our model to produce coherent sets of predictions, we defined a loss function suitable for training our system end-to-end. We envision that this approach may also prove effective in other prediction tasks with structured outputs, such as people tracking and articulated pose estimation.

²<http://www.youtube.com/watch?v=QeW10h3kQ24>

Acknowledgements. This work has been supported by the Max Planck Center for Visual Computing and Communication. The authors would like to thank NVIDIA Corporation for providing a K40 GPU and Angelcam³ for providing some of the images used in this work. The authors would also like to thank Will Song and Brody Huval for helpful discussions.

References

- [1] NLPcaffe. <http://github.com/Russell91/NLPcaffe>. Accessed: 2015-05-23.
- [2] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using Hough transform. In *CVPR 2010*.
- [3] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.
- [4] A. Farhadi and M.A. Sadeghi. Recognition using visual phrases. In *CVPR 2011*.
- [5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR'14*.
- [6] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML 2006*.
- [7] Sepp Hochreiter and Juergen Schmidhuber. Long short-term memory. In *Neural Computation* 9, 1997.
- [8] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, Fernando Mujica, Adam Coates, and Andrew Y. Ng. An empirical evaluation of deep learning on highway driving. *CoRR*, abs/1504.01716, 2015.
- [9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [10] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR'15*.
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS'12*.
- [12] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR 2005*.
- [13] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *NIPS'10*.
- [14] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. In *CVPR'13*.
- [15] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR'14*.
- [16] Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to sequence learning with neural networks. In *NIPS*2014*.
- [17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [18] Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014.
- [19] Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele. Detection and tracking of occluded people. In *BMVC 2012*.
- [20] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.
- [21] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. In *CVPR*, 2015.

³www.angelcam.com