Pedestrian Detection with a Large-Field-Of-View Deep Network

Anelia Angelova Alex Krizhevsky Vincent Vanhoucke

Abstract

Pedestrian detection is of crucial importance to autonomous driving applications. Methods based on deep learning have shown significant improvements in accuracy, which makes them particularly suitable for applications, such as pedestrian detection, where reducing miss rate is very important. Although they are accurate, their runtime has been at best in seconds per image, which makes them not practical for onboard applications. We present here a Large-Field-Of-View (LFOV) deep network for pedestrian detection, that can achieve high accuracy and is designed to make deep networks work faster for detection problems.

The idea of the proposed Large-Field-of-View deep network is to learn to make classification decisions simultaneously and accurately at multiple locations. The LFOV network processes larger image areas at much faster speeds than typical deep networks have been able to do, and can intrinsically reuse computations. Our pedestrian detection solution, which is a combination of a LFOV network and a standard deep network, works at 280 ms per image on GPU and achieves 35.85 average miss rate on the Caltech Pedestrian Detection Benchmark.

I. Introduction

Pedestrian detection is naturally very important in applications such as driving assistance or autonomous driving. These applications are becoming more and more mainstream with several auto makers already offering pedestrian detection warning systems [1], [2], [3], others planning to have fully integrated such systems within a couple of years [4], or developing technologies for level 4, or fully autonomous driving [5].

Pedestrian detection has also seen intense development [6], [7], [8], [9], [10], incorporating various classification algorithms to recognize pedestrians: Adaboost, SVM, Decision Forests [8], [9], [10]. Recently, deep learning methods have become the top performing approaches for pedestrian detection [11], [12], [13], and have also shown overwhelmingly good results in other computer vision applications [14], [15], [16], [17], and similarly in other domains [18], [19], [20], [21]. Deep networks are also versatile, as they do not need task-specific or hand-crafted features and can perform equally well on both rigid (e.g. traffic signs, cars) and deformable object categories without having to explicitly model parts and their relationships. Furthermore, deep networks readily transfer knowledge between domains by training on one domain and improving results by fine-tuning in another [16]. However, their main drawback has been the speed of classification.

The task of detection is more complex than classification alone, as it is required to also localize where the object is, including its extents and scale. One of the most common approaches for detection has been the sliding window approach [22], [6], in which a classifier is applied to a window of pre-specified size, then the same task is repeated for the neighbouring window and so on. A cascade-of-classifiers [22] is typically employed to save time, by applying simpler models first and complex models on only the hardest patches. Irrespective of the of use of a cascade, a sliding-window style detection requires thousands of classifications per image, which increases the computational time of the individual classifier by the number of locations tested.

We propose an alternative way to perform detection that is based on deep networks. Instead of exhaustively sliding a classifier, we deploy a large-field-of-view deep neural network that can understand larger areas of the image, and thus examines fewer positions in the image. The LFOV deep network is *trained* so that it can do multiple detections *simultaneously at multiple locations*, Figure 1. As a result, much fewer of those classifiers can be distributed in the image to fully cover it for detection. Secondly, because of its deep architecture, the LFOV network can reuse computations, that is, the computations for detecting pedestrians at multiple locations are shared within the network. It also can take advantage of context, due to the large field of view of operation. Thirdly, the LFOV deep network is designed so that it is faster than if we would have deployed separate classifiers with the same input dimensions and same capacity (Section III-E). In combination with a standard deep network in a shallow cascade, the LFOV classifier is much faster than any prior deep learning techniques and achieves competitive performance.

The contributions of this work are as follows: The main objective of the proposed LFOV deep network is to provide an algorithmic speedup for the very powerful but otherwise extremely slow deep networks. Our end-to-end pedestrian detection solution, which is entirely based on deep learning, provides more than 60x speedup over the single deep network approach, for both CPU and GPU implementations. This is done with a very small loss in accuracy compared to a standard deep learning method. Secondly, we show that a deep network can be successful at simultaneously making decisions at multiple locations. Lastly, we demonstrate that the LFOV deep network can be made faster than individual smaller networks. This realization is very important to be able to achieve a speed advantage. We also show that our solution is very competitive with the state-of-the-art in both accuracy and computational time, by running the whole detection at 280 ms per image on an NVIDIA Tesla K20 GPU, whereas prior such methods are on the order of seconds [13]. In the context of more general object detection algorithms that use deep-network implementations, the fastest known runs at about 13 seconds per image on GPU [23].

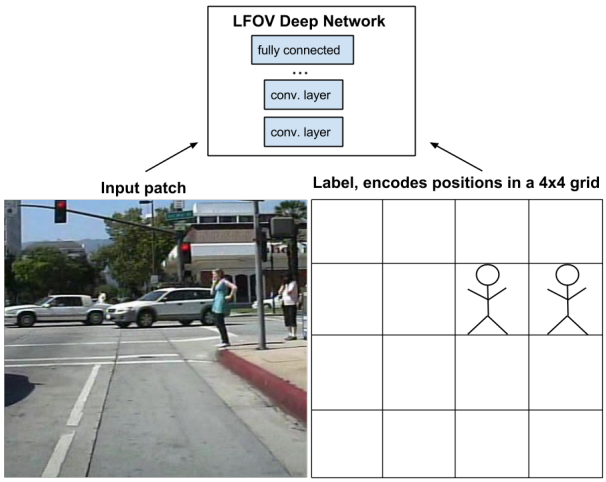

Fig. 1. Large-field-of-view deep classifier is a deep network that is trained to detect simultaneously multiple objects on a regular grid. At a single scale, a larger portion of the image is taken as input. The output label is here 16 dimensional, where each position encodes whether there is a pedestrian in the corresponding 4x4 grid cell of the input image. For example, here two pedestrians can be discovered by the application of a single LFOV classifier, while all possible 16 locations are tested simultaneously. The LFOV is run at multiple scales, similar to other detectors.

The LFOV deep network alone can also be viewed as a new way to generate proposals for a detection task that requires precise localization, such as pedestrian or traffic sign detection. The LFOV network, when used as a mechanism for generating proposal boxes, takes about 100ms per image on GPU, and also has very high recall for pedestrian detection (Section IV-B), which is hard to achieve through other standard bounding-box proposal methods.

We implemented our LFOV classifier using the open source system [14], and provided all details to the best of our knowledge (Section III-A), so that they can be easily reproduced. We also plan to submit our results to the open pedestrian detection benchmark [24], so they can be directly comparable with others.

II. Previous work

Pedestrian detection has been a topic of many recent works, many of which have had significant impact [25], [6], [26], [7], [27], [8], [9], [28], [10]. The algorithm of Viola and Jones for face detection, which proposed fast features and cascade of Adaboost classifiers [22], has been applied to pedestrian detection [29], generating interest in this domain too. Dalal and Triggs [6] developed the HOG features, again for the purposes of pedestrian detection. This work has been influential to many other computer vision problems.

An important recent work has been of Dollar et al [27], who developed a publicly available toolbox [30] and a benchmarking dataset [7], [31]. As a result, many present and future methods can be evaluated in the same setting. Another important work has been on increasing the speed of pedestrian detection with proposed methods reaching speeds of 100 to 135 fps [8]. These methods are the fastest reported for pedestrian detection. A lot of methods have introduced or experimented with a large variety of features, thus pushing the progress of pedestrian detection and steadily improving the state-of-the-art [10]. Some methods, based on the Deformable Parts Models, have also been successfully applied in the context of pedestrian detection [32], [33]. Other methods have explored multi-scale solutions, or employed context or motion features [32].

Deep learning methods have been applied to pedestrian detection, improving on the detection accuracy over many prior methods [11], [12], [13]. However, their run-time has been somewhat slow, i.e. about 1-1.5 seconds per image [13], or even reaching runtimes in the order of minutes [34]. Our method is also based on deep networks, and because of that is very accurate, and at the same time is several times faster than other deep network methods. Prior deep networks have used edge filters to initialize weights rather than learning from raw pixels values, or have used deep nets in combination in a deformable-parts model style to model parts [11]. We train our deep network LFOV classifier from raw pixel values and without hand-made or any special features, and we show that it can achieve competitive results. In addition, our framework allows for simultaneous processing at multiple locations, and thus reuses computation and takes advantage of context, whereas prior methods take advantage of context by running separate classifiers, e.g. for cars [33], or other pedestrians [35].

In the domain of object detection, the sliding window detection has been the most common. It essentially samples the image densely and evaluates all patches and is often applied in a cascade-style, i.e. progressively employes harder classifiers. Our approach is closer to the traditional sliding window cascades as it will de facto examine all locations. It differs by offering a larger field of view to make its decision, and thus speed up the detection.

An alternative to a cascade approach is to first identify proposal boxes [36], [23]. Previous methods that apply proposal windows are much slower (in tens of seconds) and have not been applied to applications that demand accurate localization.

Fig. 2. Procedure for generating examples for training for the LFOV classifier: each of these examples is generated by positioning a pedestrian in the center of each cell of a 4x4 grid, in as many cell positions as possible.

For example, in the most recent work of Girshick et al [23] detection with proposal boxes takes 53 seconds per frame for a CPU implementation and 13 seconds per frame on GPU. When using the LFOV classifier as a mechanism for generating proposal boxes, it runs about 100 ms per image, and it can recall a large portions of pedestrians, which typically take very small areas of the image.

Recent work has focused on reducing the computational time needed for detection of deep nets [37], [38]. These methods optimize detection by reusing computation or make computations more efficient. These approaches are still relatively slow, more than one second per image and not suitable for time-sensitive problems. Instead of speeding up the computations after the models are learned, our LFOV classifier is designed to reuse computations at the time of training.

III. Large-field-of-view deep network

We here describe our proposed method. The Large-Field-of-View (LFOV) classifier is a deep neural network which works on larger areas of the image as input. For the input it takes, e.g. the bottom left image of Figure 1, it is trained to output whether there is a pedestrian at multiple locations. To be specific, we subdivide the input image into a 4x4 grid and for each grid cell, the network learns whether there is a pedestrian that is centered within this cell or not (Figure 1). A convolutional deep neural network is applied to the input (details provided in Section III-A) and the output that the network is going to learn is a 16 dimensional output, in which each output dimension corresponds to one cell in the grid and its interpretation is whether this cell has a pedestrian or not (in fact, for convenience we used a 17-dimensional output, details are below).

A. LFOV architecture

The Large-Field-of-View (LFOV) classifier is going to be used as a first stage in generating proposal boxes for pedestrian detection. We implemented it here as a very simple convolutional network. We have also designed it so that its computational time is very fast. The architecture is inspired by the original work of Krizhevsky et al [14] but is much simpler.

More specifically the LFOV classifier works on 64x64 RGB image input and has the following layers:

- A 5x5 convolutional layer with a filter depth of 32, with a stride of 2. It is followed by a normalization layer, as defined by [14] and a pooling layer (3x3, stride of 2).

- A 1x1 convolutional layer of depth 32. This type of layer was first introduced here [39] and is added as it essentially adds depth to the full network at very little computational cost.

- A fully connected layer of depth 512. It is followed by a standard dropout layer [40] of 0.5 dropout rate.

- Finally, a fully connected layer of 17 outputs is attached at the end, to which a cross-entropy cost is applied. Each output is responsible for one of the cells in the 4x4 grid, plus an additional output for detecting a pedestrian that spans the full image. Although we did not have to add the pedestrian that takes the full image, we found it useful to seamlessly detect pedestrians at this higher resolution as well.

All layers, except the 1x1 convolutional one, are followed by a rectified linear unit. Figure 3 visualizes the architecture.

The main idea of LFOV classifier is that the deep network classifier, through several layers of inference, is capable of learning about the presence of pedestrians at different smaller scales. Indeed we observed that it can learn very successfully the presence of pedestrians at these lower scales, but at the same time the pedestrian at 1x1 grid is learned with higher success rate than the ones at the 4x4 grid. Furthermore, in our implementation, to save time, we use an input image size of 64x64, so we are effectively detecting pedestrians of very low resolution, 16x16, in each grid cell. We note here that larger and more complex deep network can also be directly trained over the 4x4 grid inputs. We did not pursue that approach,

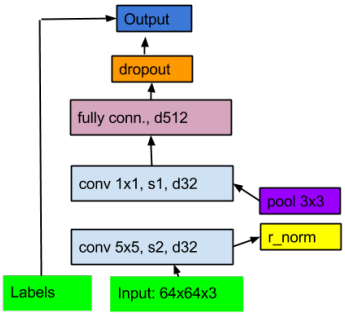

Fig. 3. Architecture of the Large-field-of-view (LFOV) deep network which runs as a first stage of our pedestrian detection. We use 's' to denote the step size, and 'd' to denote the filter depth per layer.

because applying a LFOV classifier at full resolution would be too slow for applications such as pedestrian detection, which strive for at least several frames per second in processing time.

Note that, as our model uses fully-connected layers in the design, the decision about a pedestrian in a cell will be dependent on the information that is available in the neighbouring cells and the whole grid, which will provide context. That said, the design of our classifier is block-wise, so we will not always have full context available e.g. the most top-right pedestrian may not have full context. We could avoid that by testing overlapping decisions, but we chose not to, in the interest of speed.

B. Training the LFOV network

Figure 2 visualizes how we generated examples for training the LFOV network. More specifically, we focused on a positive example, and generated all possible square boxes around the pedestrian, so that the person falls into all of the cells of the 4x4 grid, where possible. Pedestrians that are not centered in a cell are not considered as positive examples. Figure 2 shows examples that are used as inputs to our network. Note that although the pedestrians may look quite hard to spot in those images at a first glance, our LFOV network can successfully learn to detect them (even when rescaled so that each pedestrian is approximately 16x16 pixels in size). Section IV-B has details about the successful recall of the trained LFOV classifier, i.e. what percentage of the available pedestrians in all images have been found by LFOV classifier on the test data. As the pedestrians came at different sizes, we generate data at different scales, which is later rescaled to the same size network. Similarly, at detection time, the network is applied at multiple scales.

To generate negative examples, we sampled randomly across the image and included examples that do not have pedestrians in any of the boxes. Pedestrians that are in-between grid cells are not included as either positive or negative. Because of this, for the last stage of the classifier, we harvested hard negative examples, by running the trained classifier and including additional negative examples that are not classified correctly. This is necessary because the random sampling of negative examples will generate too many and 'too easy' examples, which do not benefit the classifier much.

Pre-training. As is standard with deep neural networks, we used pre-training from an equivalent network that was trained on Imagenet dataset [14]. Pre-training was applied only to our baseline (last-stage) classifier. While that is not a strict requirement, because of the richness of the features obtained from a more diverse dataset, pre-training can be helpful for tasks for which training data may not be fully sufficient. We observed a very small improvement with pre-training.

C. Detection with LFOV

When detecting with a regular sliding window classifier the step size at which each classifier is tested is very important. For too large step sizes, one can miss important detections, conversely, small sizes, e.g. 1 or 2 pixels will make the processing extremely slow. We focused on a step size of 4 pixels for our baseline algorithm, as a good tradeoff for all our evaluations. It is likely that better results are achieved with more dense sampling, both in terms of final accuracy and in terms of increase in pedestrian recall.

The LFOV classifier was implemented to use the same step size as dense sampling would. Let us assume that the LFOV network is implemented with a 4x4 grid and is considering an input of size 64x64. This means that within each grid cell we are considering detecting a pedestrian of size 16x16. When applying the LFOV classifier, for this example, we can see that at positions 0, 16, 32, and 48 in both horizontal and vertical directions, we would detect simultaneously a pedestrian if there is one. However, for the desired step size of 4 pixels, in this example, we would not very successfully detect a pedestrian, where its top left corner starts at positions 4, 8, 12, 20, etc. To do that we actually apply a local sliding window to cover all displacements, so as to test all locations at the desired step size. Note however, because of the LFOV trick, we *only* need to test these displacements to cover the top left grid cell and the LFOV classifier will automatically be testing all step sizes for all the grid cells necessary. After doing the local dense sampling of steps 0, 4, 8, 12, the next step size

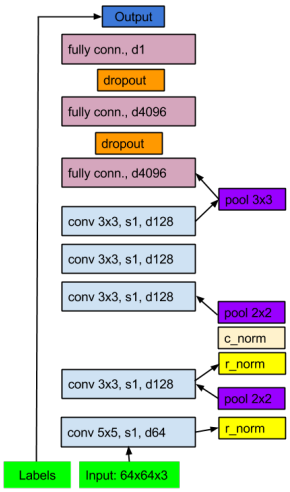

Fig. 4. Architecture of the deep network classifier which runs as a third and final stage of our detection algorithm.

we need to test with the LFOV classifier will be directly 64 rather than 16.[1] In effect, for this example, the step sizes the LFOV classifier needs to take are as follows: 0, 4, 8, 12, 64, 68, 72, 76, 140, etc.

D. LFOV Cascade

Since we need the final pedestrian detector to work fast, we are going to use a shallow cascade of deep networks, where the LFOV classifier is the first stage.

Our algorithm consists essentially of 3 stages: 1) A LFOV classifier, Figure 3, which works at larger blocks of the image and is specifically designed to be fast (as just described in Section III-A). Its purpose is to generate proposal boxes where pedestrians can be detected. 2) A small deep network that uses the same deep network model architecture as the LFOV (as shown in Figure 3), but works on 16x16 image inputs instead of the large-field 64x64 inputs. This classifier is applied to all candidate boxes that the LFOV produces as positive classifications. It also has a single 1-dimensional output, which determines whether the input image contains a pedestrian or not. 3) The last stage is a standard deep network as in Krizhevsky et al [14]. Naturally it is applied to all candidates that the previous stages have generated. The middle classifier is only needed for speed, and naturally can lose a bit of precision (as shown later in our experiments). The architecture of the last stage is described in Figure 4.

The first two stages are trained to have high recall.

E. Design of the Large-Field-of-View Network

The most important component of the LFOV deep network is that it is several times faster than the cumulative work of standard deep networks of the same capacity. This means that the LFOV network is beneficial in terms of speed compared to other deep network models. Here are the details.

Among the models at the first stage, out fastest model at this resolution (of 1 convolutional and 1 fully connected layer) which is trained to have sufficiently high recall, works at 0.5 milliseconds per 128 patches on GPU (the model we actually used had an additional 1x1 convolution and is slower, taking 0.67 milliseconds). An image of this size typically has about 80K candidates. This makes the processing of this image at least 80000*0.0005=40 seconds. When we divide by 128, we get 40/128 = 0.3125 seconds, which means it needs at least 300 ms to apply the fastest possible individual network (it will take more than 400 ms for the model with 1x1 convolutions). The LFOV classifier that can simultaneously localize 16 pedestrians and has 4x4 larger field of view, with the architecture as in Figure 3, works at 3 msec per 128 input images. Per image we evaluate 4500 large-field-of-view patches, so 4500/128*3=105 msec. Compare this to the standard classifier, we can see that we have 3x speedup compared the best timing we can achieve with the same architecture, but with a standard deep network classifier. Thus the LFOV classifier has the advantage of working on the same input (with the same capacity) but doing that several times faster. This is a key observation in the design of the LFOV classifier. We also note that more speedups are possible from considering larger fields of view.

[1]In theory the next step can be at 64+12=76 since the last grid cell would have covered these additional step sizes, but our implementation is more straightforward and did not take advantage of that possible speedup.

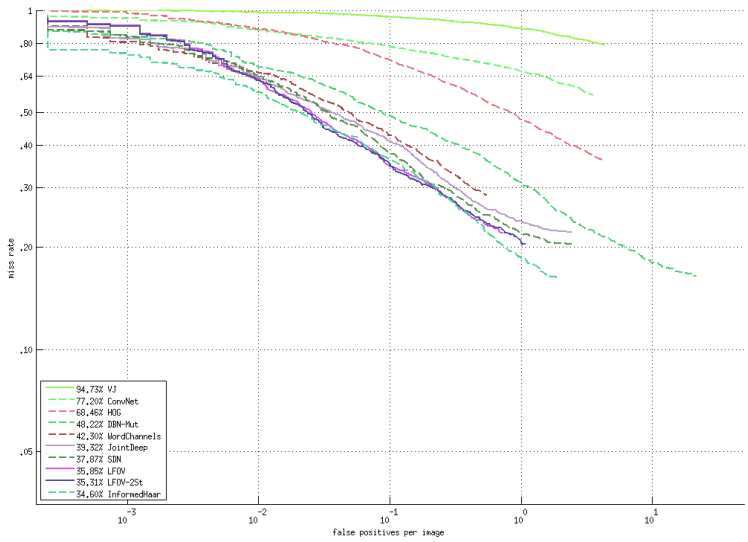

Fig. 5. Results on Caltech test data compared to state-of-the-art results in pedestrian detection. Plots are ordered by their average miss rate.

IV. Experimental evaluation

A. Caltech pedestrian detection dataset

We evaluated our results on the Caltech Pedestrian detection dataset [7], which has been the main benchmark for pedestrian detection and a large number of methods have been evaluated on it [24].

We use the standard training and test protocols established in the Caltech benchmark and report the results by measuring the mean miss rate, as had prior methods. We use the code provided in the toolbox of this benchmark [30] to do the evaluation. The results below are obtained when training with the standard Caltech-Training dataset for training. Other works have included additional Inria dataset, but we chose not to because the Inria dataset is less relevant to pedestrian detection for autonomous driving. Our baseline classifier has started training from a model pretrained on Imagenet, as in [14].

Figure 5 shows the performance of our method in comparison to the state-of-the-art methods, using Piotr Dollar's evaluation toolbox and results provided therein [30]. We tested on the Caltech Test set in the 'reasonable' setting, as is standard for all methods. Figure 5 shows the average miss rate and the false positive rate per image vs. the miss rate, which follows the standard evaluation protocol. In the interest of showing less cluttered plot, we visualize only the best methods and also a number of common methods, as is suggested by [31]. Table I shows additional details, including the most recent successful approaches.

As seen in Figure 5, the LFOV pedestrian detector achieves mean average precision of 35.85%, and is among the best performing results. When the middle stage of the cascade is ignored (LFOV-2St), our method performs at 35.31%. The InformedHaar [41] classifier is the best so far with average mean miss rate at 34.6%. We also note that there are recent unpublished results that have achieved 29% on Caltech data, and 22% when using motion features.

Table I provides additional information regarding runtime and training data and additional approaches. There we focused on the best, most recent methods that have used primarily Caltech data for training. The runtime of our method is very good on GPU, especially considering that deep network methods have been notoriously slow to run on CPU, or to run detection on GPU. We can notice that most methods have not considered runtime as very important and did not report it. Prior deep-learning based methods work at more than one second per frame. Fast methods, such as WordChannels [42] or VeryFast [8], which run at more than 16 frames per second of GPU or CPU, are at the same time not among the best performing. We also note that prior methods that have used deep learning techniques [11], [13], typically use other fast features for early elimination of most patches.

Although we do not use explicitly context to detect pedestrians it is implicit in our model since the large network can in principle observe larger areas of the image (and make use of context). We can see that our method outperforms others that have used context. For example, Ouyang and Wang [35] use two-pedestrian detection to improve on single-pedestrian detection, Ouyang et al [43] jointly detect two neighbouring pedestrians. Yan et al [33] use a vehicle detector to help remove false alarms and improve on the pedestrian detection. Considering that adding context in prior methods is likely to increase computational time and slow down the detection, the LFOV classifier has an advantage, as the context is built-in in the large field of view and no extra computation is needed. We believe, however, that some additional sources of context, such as the

TABLE I

COMPARISON OF OUR METHOD TO STATE-OF-THE-ART RESULTS FOR PEDESTRIAN DETECTION, INCLUDING TIME FOR TESTING. WE FOCUS ON RECENT METHODS THAT USED CALTECH TRAINING DATA AS THEIR MAIN DATA SOURCE.

Method	Mean miss rate (%)	Timing (seconds per image)	Training dataset
MultiResC [32] (multires)	48.5		Caltech
DBN-Mut [43] (deep)	48.2		Caltech, Inria
Roerei [10]	46.13	1	Inria
MOCO[44]	45.5		Caltech
MultiSDP [12] (deep, w. context)	45.4		Caltech, Inria + Context
WordChannels [42] (multires)	42.3	0.06 (GPU)	Caltech, Inria
MT-DPM [33]	40.5	1	Caltech
JointDeep[11] (deep)	39.3		Caltech, Inria
SDN[13] (deep)	37.9	1-1.5 (GPU)	Caltech, Inria
MT-DPM+Context [33] (w. context)	37.64		Caltech + Context
ACF+SDt [45] (w. motion)	37.3		Caltech + Motion
InformedHaar [41]	34.6	1.6	Caltech, Inria
Ours (LFOV, deep)	35.85	0.28 (GPU)	Caltech, Pretr.
Ours (LFOV-2St, deep)	35.31	0.55 (GPU)	Caltech, Pretr.

TABLE II

RECALL OF EACH STAGE OF OUR CLASSIFIER MEASURED ON THE CALTECH TEST SET. RESULTS AFTER NON-MAXIMUM SUPPRESSION (NMS) ARE SHOWN AT THE BOTTOM.

Classifier	Pedestrian Recall
Baseline (Stage 3)	93.49
LFOV (Stages 1,3)	89.55
LFOV (Stages 1,2,3)	87.08
LFOV w. NMS (Stages 1,3)	82.29
LFOV w. NMS (Stages 1,2,3)	78.01

presence of a vehicle as in Yan et al [33] can further improve our proposed method too. Additionally, motion information can be quite helpful, e.g. Park et al [45] utilize motion stabilization to improve the baseline classifier. Since some pedestrians are extremely hard to identify as individual examples, without any motion data, we believe that it can greatly improve our classifier too.

Figure 6 shows some example detections. Here we can see both successful detections, but also some missed and false alarms.

B. LFOV network recall

We here compute the average recall each of the stages our classifier can provide. The first stage (LFOV) can be viewed as a mechanism for generating proposal boxes, and we believe it can be suitable for detecting other objects, e.g. traffic signs. Table II shows the recall of each network (prior to NMS). The recall measures what percentage of the available pedestrians in all images have been found by the classifier on the test data.

As seen, the recall of LFOV deep network is reduced by adding more stages of the cascade, which is as expected. At the same time, the shallow cascade is needed for obtaining speedup. We can also observe that the NMS algorithm is also contributing to a considerable loss in recall.

C. Timing

We measured the timing of each of the stages on the GPU and estimated the final runtime[2]. The LFOV classifier which is applied at the first stage takes about 105 ms, The second stage takes about 20-25 milliseconds, depending how many boxes are sampled. The last stage takes about 150ms, for a baseline model that runs at about 50 milliseconds per 128 images. Overall the full LFOV-based detection algorithm works at 280 ms per frame on the GPU. If, on the other hand, we apply

[2]Timings are approximate, since only individual stages' performance is measured on the GPU.

TABLE III

SPEEDUP OF OUR LFOV DEEP NEURAL NETWORK DETECTOR COMPARED TO THE BASELINE DEEP NEURAL NETWORK.

Classifier	Speed CPU (seconds)	Speed GPU (seconds)
LFOV (Stages 1,2,3)	122	0.22-0.28
Baseline	7738	18.2-30
Speedup	**63**	**83-108**

Fig. 6. Example detections.

the baseline model to all input candidate windows, it will need 30.47 seconds, which is a 108.8 speedup of the LFOV classifier. For a faster baseline algorithm, e.g. 30ms, we will have 18.2 seconds for the baseline and 220 milliseconds for LFOV, which is 83x speedup. We also measured the corresponding CPU runtime in both cases. Table III shows the speedups we can achieve for our classifier when running on both CPU and GPU (where our CPU implementation is not optimized and somewhat slow). As seen the speedups for both are larger than 60x, which is a very good algorithmic speedup of the LFOV classifier.

We note here that the final runtime is not real-time, as is desired at e.g. 10 fps. For simplicity sake we kept our algorithm entirely based on deep networks, but like all other previous deep network solutions [13], [11], it is conceivable to incorporate other faster features and use them to gain additional speed advantage.

V. Conclusion and Future Work

This paper proposed a Large-Field-of-View classifier, which is a deep network which processes larger areas of the image and simultaneously makes decisions of the presence of pedestrians at multiple locations. The LFOV network is designed in such a way that it is faster than the cumulative time of standard deep networks. As a result it can process the full image much faster. Our method can be applied at 3.6 fps on GPU for online pedestrian detection, and can also be valuable as and offline speedup for detection algorithms: it is a way to reliably detect small objects (such as pedestrians) without losing much recall and keeping original precision, and at the same time increasing the speed by more than 60 times.

Deep networks have been shown to be ”large capacity” classifiers, so future improvements in deeper and more complex networks are bound to yield even more accuracy gains. So we believe the proposed combination of employing deep networks for both improved accuracy and for doing more work, e.g. as in this case for detecting multiple pedestrians simultaneously, is a novel direction which can allow for new practical applications.

We demonstrated end-to-end detection solution that is based entirely on deep neural networks, but the proposed solution can become even more interesting when the capacity of the neural networks becomes bigger, i.e. when they can handle larger inputs. For example, right now we can consider a 4x larger field of view, and thus getting the neural network do 16x detections simultaneously. But for networks with even larger fields of view, one can consider even more speedups.

Acknowledgements We thank Abhijit Ogale for providing the CPU implementation. We also thank Dave Ferguson and Abhijit Ogale for their comments and discussions.

REFERENCES

[1] "Mobileye Pedestrian Collision Warning System www.mobileye.com/technology/applications/pedestrian-detection/pedestrian-collision-warning/."
[2] E. Coelingh, A. Eidehall, and M. Bengtsson, "Collision warning with full auto brake and pedestrian detection - a practical example of automatic emergency braking," *13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2010.
[3] "BMV Driving Assistance Package http://www.bmw.com/."
[4] "VW Emergency Assistance System http://safecarnews.com/next-gen-vw-passat-to-feature-emergency-assistance-systems/."
[5] "https://plus.google.com/+googleselfdrivingcars/ Google Self-Driving Car Blog."
[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *CVPR*, 2005.
[7] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," *CVPR*, 2009.
[8] R. Benenson, M. Matthias, R. Tomofte, and L. V. Gool, "Pedestrian detection at 100 frames per second," *CVPR*, 2012.
[9] P. Dollar, R. Appel, and P. Perona, "Crosstalk cascades for frame-rate pedestrian detector," *ECCV*, 2010.
[10] R. Benenson, M. Matthias, T. Tuytrlaars, and L. V. Gool, "Seeking the strongest rigid detector," *CVPR*, 2013.
[11] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," *ICCV*, 2013.
[12] X. Zeng, W. Ouyang, and X. Wang, "Multi-stage contextual deep learning for pedestrian detection," *ICCV*, 2013.
[13] P. Luo, X. Zeng, X. Wang, and X. Tang, "Switchable deep network for pedestrian detection," *CVPR*, 2014.
[14] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *NIPS*, 2012.
[15] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," *ICML*, 2014.
[16] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," *CVPR Workshop on Deep Learning in Computer Vision*, 2014.
[17] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Localization and detection using convolutional networks," *ICLR*, 2014.
[18] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, 2003.
[19] A. Mohamed, G. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. on Audio, Speech and Language Processing.*, 2010.
[20] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," *IJCNN*, 2011.
[21] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.
[22] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *CVPR*, 2001.
[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *http://arxiv.org/pdf/1311.2524v4.pdf*, 2013.
[24] "P. Dollar, Caltech Pedestrians Benchmark www.vision.caltech.edu/image_datasets/caltechpedestrians/."
[25] C. Papageorgiou, T. Evgeniou, and T. Poggio, "A trainable pedestrian detection system," *Proceedings of Intelligent Vehicles, Stuttgart, Germany*, 1998.
[26] C. Wojek, S. Walk, and B. Schiele, "Multi-cue onboard pedestrian detection," *CVPR*, 2009.
[27] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," *BMVC*, 2010.
[28] Y. Ding and J. Xiao, "Contextual boost for pedestrian detection," *CVPR*, 2012.
[29] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *ICCV*, 2003.
[30] "www.vision.ucsd.edu/ pdollar/toolbox/doc/index.html P. Dollar Toolbox."
[31] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *PAMI*, 2012.
[32] D. Park, D. Ramanan, and C. Folwkes, "Multiresolution models for object detection," *ECCV*, 2010.
[33] J. Yan, X. Zhang, Z. Lei, S.Liao, and S. Li, "Robust multi-resolution pedestrian detection in trafc scenes," *CVPR*, 2013.
[34] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," *CVPR*, 2013.
[35] W. Ouyang and X. Wang, "Single pedestrian detection aided by multi-pedestrian detection," *CVPR*, 2013.
[36] K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders, "Segmentation as selective search for object recognition," *ICCV*, 2011.
[37] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv technical report*, 2014.
[38] A. Giusti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," *ICIP*, 2013.
[39] M. Lin, Q. Chen, and S. Yan, "Network in network," *http://arxiv.org/abs/1312.4400*, 2014.
[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
[41] S. Zhang, C. Bauckhage, and A. Cremers, "Informed haar-like features improve pedestrian detection," *CVPR*, 2014.
[42] A. Costea and S. Nedevschi, "Word channel based multiscale pedestrian detection without image resizing and using only one classifier," *CVPR14*, 2014.
[43] W. Ouyang, X. Zeng, and X. Wang, "Modeling mutual visibility relationship in pedestrian detection," *CVPR*, 2013.
[44] G. Chen, Y. Ding, J. Xiao, and T. Han, "Detection evolution with multi-order contextual co-occurrence," *CVPR*, 2013.
[45] D. Park, L. Zitnick, D. Ramanan, and P. Dollar, "Exploring weak stabilization for motion feature extraction," *CVPR*, 2013.