A Sparse Deep Feature Representation for Object Detection from Wearable Cameras

Quanfu Fan[1]
qfan@us.ibm.com
Chun-Fu (Richard) Chen[1]
chenrich@us.ibm.com
Gwo Giun (Chris) Lee[2]
clee@mail.ncku.edu.tw

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2] National Cheng Kung University, No. 1, University Rd, Tainan, Taiwan

Abstract

We propose a novel sparse feature representation for the faster RCNN framework and apply it for object detection from wearable cameras. Two main ideas, *sparse convolution* and *sparse ROI pooling*, are developed to reduce model complexity as well as computational cost. Sparse convolution approximates a full kernel by skipping weights in the kernel while sparse ROI pooling performs feature dimensionality reduction on the ROI pooling layer by skipping odd-indexed or even-indexed features. We demonstrate the effectiveness of our approach on two challenging body camera datasets including realistic police-generated clips. Our approach achieves a significant reduction of model size by a factor of over $10\times$ as well as a computational speedup of about $2\times$, yet without compromising much detection accuracy compared to a VGG16-based baseline detector.

1 Introduction

There is a growing interest in the use of wearable cameras by major urban police departments throughout the U.S. A body camera, typically attached to an officer's shoulders or glasses, can faithfully record the activities of the officer from his own perspective. It thus provides a more transparent relationship between police and public. Moreover, body cameras generate tremendous data, which enable useful video analytics applications such as automatic redaction and suspect search [1] for improving policing and public safety. For those applications to be effective, one of the fundamental technologies needed is object detection, to find either faces or individuals for further visual analysis.

Over the past few years, CNN-based approaches [2, 5, 13, 16, 17, 18] have been paramount for generic object detection. Among them, region-based CNNs (RCNNs) [2, 5, 18], especially faster RCNN [18], have significantly advanced this field, achieving state-of-the-art performance on challenging datasets such as PASCAL VOC [4] and COCO [14]. Nonetheless, to yield competitive performance on large-scale datasets, RCNNs rely on deep CNN models with substantial complexities such as VGG16 [19] and ResNet152 [8]. These models require a lot of memory to run and are inefficient in computation. This greatly limits their applications to body-worn cameras. In addition, CNN-based models have not been much evaluated in surveillance settings. Hence, it is not clear whether or not their performance

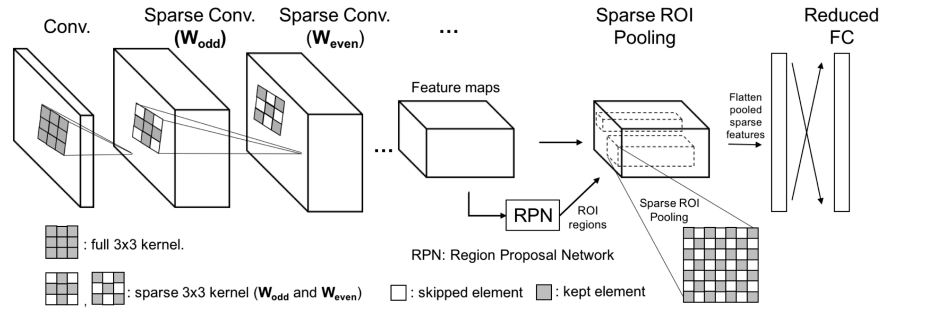


Figure 1: A sparse representation for the faster RCNN detection framework. Two main ideas, *sparse convolution* and *sparse ROI pooling*, are proposed to reduce model complexity as well as computational cost. *Sparse convolution* uses manually-designed sparse kernels to approximate the full kernels in a model while *sparse ROI pooling* skips features at consecutive locations in ROI pooling for dimensionality reduction. Illustrated is an example of a 3×3 full kernel and its corresponding sparse versions, which spatially complement to each other.

is still satisfactory under the challenges posed by body cameras such as severe motion blur, heavy occlusions and camera distortion et al. [1]

Inspired by recent works on model compression through kernel sparsity learning [6, 20], we develop a novel sparse feature representation under the faster RCNN framework and apply it for object detection on wearable cameras. Our essential idea is to reduce parameter redundancy at different layers in a detection model by using different techniques. For convolutional layers, we exploit two manually-designed sparse kernels to approximate full convolutions directly. We specifically make the sparse kernels spatially complementary, with non-zero weights either at the even or odd indices of a full kernel. The two kernels alternate in sequence to substitute the full kernels in a CNN model, as illustrated in Fig. 1. Our approach uses deterministic sparse kernels that allows for training a model from scratch. This distinguishes itself from previous works such as [6, 20] that depend on pre-trained models.

Features from the ROI pooling layer are high dimensional. We propose *sparse ROI pooling* to reduce their dimensionality. Similar to the *sparse convolution* described above, *sparse ROI pooling* simply skips either the odd-indexed or the even-indexed features to avoid pooling a same neuron response on the output feature maps. This halves the number of parameters at the first fully connected (FC) layers, yet without compromising the capability of feature representation. Finally, we show that significantly reduced FC layers with low capacity are sufficient to provide good recognition capabilities for face and person objects from body cameras.

To sum up, our main contributions are:

- a sparse feature representation that enables VGG-based faster RCNN to achieve a) a $10\times$ reduction of model parameters and b) a computational speedup by a factor of nearly $2\times$
- a comprehensive performance evaluation of object detection with faster RCNN on body camera data
- demonstration that significantly reduced FC layers with low capacity do not compromise recognition capabilities of faster RCNN on body camera data

2 Related Works

In recent years, there has been significant progress made in object detection using CNNs [2, 5, 13, 16, 17, 18]. These models demonstrate impressive results on large-scale datasets, and some of them such as SSD [16] and YOLO [17] are even able to run in real time. However, most of them use very deep CNNs with substantial complexities as feature extractors. While newly proposed network architectures such as SqueezeNet [11] and MobileNet [9] make it possible to build efficient and small-size detectors on top of them, the performance of such detectors is still not satisfactory [10].

On the other hand, how to compress CNNs has been a hot research topic given the fact that these models are overly parameterized. For example, low-rank approximation is one of the main ideas for model compression [3, 12]. Kernel sparsity learning has gained a lot of attention [6, 15, 20, 21] recently. These approaches perform kernel approximation by sparse representations, either through pruning small weights [6] or learning to sparsify kernels by group regularization [15, 20, 21]. Nevertheless, there are very few attempts that apply compressed CNN models for object detection, which is our primary focus of this work.

3 Our Approach

3.1 Sparse CNNs

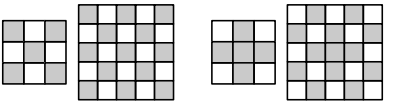
While approaches based on kernel sparsification [15, 20, 21] have demonstrated significant reductions in model size as well as computational cost, one of the limitations in these approaches is that sparsity penalties often lead to irregular patterns in kernels. This makes the computational gain in practice either too small or highly dependent on dedicated software or hardware handlings [7, 15]. In addition, sparsity learning needs a pre-trained model to start with and fine tuning is required afterwards.

To overcome the limitations aforementioned in sparsity learning, we manually design two sparse kernels to approximate full convolutions in CNN models. The two kernels, denoted by $\mathbf{W_{even}}$ and $\mathbf{W_{odd}}$ respectively, are mathematically expressed by,

$$\begin{aligned} W_{even_{i,j,c,n}} &= 0, \text{ if } (j \times k + i) \bmod 2 \neq 0 \\ W_{odd_{i,j,c,n}} &= 0, \text{ if } (j \times k + i) \bmod 2 \neq 1 \\ &\quad \text{and } (j \neq \lfloor k/2 \rfloor \text{ and } i \neq \lfloor k/2 \rfloor), \end{aligned} \tag{1}$$

where $(i,\ j)$ specifies the spatial location of a cell in a kernel, k is the kernel size, c is the channel index and n denotes the kernel index. We keep the center point nonzero for both $\mathbf{W_{odd}}$ and $\mathbf{W_{even}}$ as our analysis suggests that this location, which often carries a large weight in the kernel, is of significant importance for feature representation. Figure 2 shows the sparse patterns for 3×3 and 5×5 kernels, respectively. When the kernel size is 3×3, $\mathbf{W_{even}}$ becomes a $\times$ shape and $\mathbf{W_{odd}}$ is a $+$ shape.

Note that $\mathbf{W_{odd}}$ and $\mathbf{W_{even}}$ complement to each other as a join of them in the spatial domain gives rise to a full coverage of the receptive field of the kernel. We hope that such a design can compensate for some local details that are missing in sparse convolution but critical for object detection. In addition, our sparse kernels are based on deterministic patterns,



(a) Sparse kernel ($\mathbf{W_{even}}$) (b) Sparse kernel ($\mathbf{W_{odd}}$)

Figure 2: Examples of our proposed sparse kernels in the case of 3×3 and 5×5 ($\mathbf{W_{even}}$ and $\mathbf{W_{odd}}$). White color indicates skipped weights. For a 3×3 kernel, $\mathbf{W_{even}}$ is a '$\times$' shape, and $\mathbf{W_{odd}}$ a '$+$' shape.

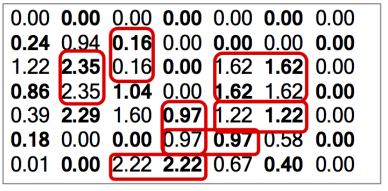


Figure 3: Sparse ROI pooling. ROI pooling leads to sparse and redundant features, especially at horizontally or vertically consecutive locations (highlighted by red boxes). *Sparse ROI pooling* skips either the odd-indexed or even-indexed features for dimensionality reduction. By doing so, it halves the number of parameters at FC6 without compromising the capability of feature representation.

meaning that there is no need to use an index table to store their patterns in memory. Therefore, as opposed to other techniques such as [6, 20], our approach provides more consistent empirical running time with regards to the theoretical FLOPs analysis.

3.2 Sparse ROI Pooling

ROI pooling in faster RCNN converts a CNN feature map inside a region proposal by max-pooling into a smaller feature map with a fixed spatial extent (*e.g.* 7×7 in VGG16). The pooled features are subsequently connected to the first fully connected layer (i.e. FC6). FC6 is overly parameterized, which is largely attributes to the high dimensionality of the ROI features. For example, in the case of VGG16, there is a total of $25,088$ ROI features, yielding 103 million parameters at FC6 that accounts for 70% of the total model size. It is thus highly desirable to perform dimensionality reduction on the ROI features for a compact detector.

Indeed, ROI features are not only sparse but redundant. In ROI pooling, a region proposal is first projected to the feature map, downsampled by a significant factor α (*e.g.* $\alpha = 16$ in VGG16). If an either side of the projected bounding box is smaller than the pooling size, then a same neuron activation is pulled multiple times for consecutive locations in the output, either in the horizontal or vertical direction. In our case, persons and faces are often small (see their size distributions in Fig. 5), thus resulting in many identical features from ROI pooling, as shown by the example in Fig. 3.

Motivated by this observation, we propose *sparse ROI pooling*, an idea similar to *sparse convolution*, to reduce the dimension of the ROI features. To avoid pooling identical features at consecutive locations, *sparse ROI pooling* simply takes either the odd-indexed or even-indexed features, but not both, as the output of the ROI pooling layer. We would like to point out that this idea is principally different from using a smaller pooling size, which results in

Figure 4: Two evaluation datasets recorded by GoPro and Indigo cameras respectively. The set *GoPro* are self-collected data while *Indigo* are realistic police-generated clips. *GoPro*: a) a farmers market b) an urban side walk and c) a metro train station; *Indigo*: d) an arrest scene e) a domestic violence scene and f) a traffic stop.

a coarser feature representation that likely hurts detection accuracy. Despite its simplicity, *sparse ROI pooling* halves the number of parameters at FC6, yet still performing similarly to the baseline models, as shown later in our experiments.

3.3 Reduced Fully Connected Layers

FC layers in a CNN account for most of the model parameters. It has been shown that FC layers present high parameter redundancy and this redundancy can be significantly pruned with sophisticated techniques such as low-rank factorization with only a minimal increase of error rate for classification [3, 12].

We apply a simple technique here to reduce the capacity of a network in the FC layers. We experimented with different numbers of neurons in the FC layers, and found that although FC layers are necessary for faster RCNN to achieve good performance, the number of their neurons can be significantly reduced without leading to much accuracy loss in person and face detection on body camera data.

4 Experimental Results

4.1 Datasets

Test Dataset. Two body-worn camera datasets were used to evaluate our proposed approach. The first set *GoPro* was recorded using a GoPro Hero4 camera with ultra HD resolution 1920×1080 at 29 FPS, This dataset includes three crowded scenes: an indoor farmers market, a metro train station and an urban sidewalk. The second set *Indigo* has six realistic police clips captured with Indigo Vision HD 1280×720 at 30 FPS, including two traffic stops, an interview scene, a domestic violence scene and an arrest scene in a parking lot. *GoPro* contains a lot of high-level activity and presents significant challenges for object detection such as crowdedness, heavy occlusions and frequent background changes. There are also many small and profile faces in *GoPro*, as shown in Fig. 5. The second set, although mostly involving one or two persons in the view, captures many difficult objects like blurred or partially visible faces/persons. Figure 4 shows a few sample images from each dataset.

We annotated persons and faces at one frame every other second for the *GoPro* data and at every 5^{th} frame for the *Indigo* data. This leads to a total of 1,089 *GoPro* images and 5,049

	ref. (VGG16)	$sc_{+\times}123$	$sc_{+\times}345$	$sc_{+\times}12345$	$sc_{+\times}\overline{12345}^{\dagger}$	$sc_{+}12345$
Top-5 Acc. (%)	90.01	**89.09**	88.78	88.70	88.90	88.22
Top-1 Acc. (%)	71.29	**69.46**	69.29	68.62	69.20	68.03
FLOPs ↓	–	1.37×	1.44×	1.79×	1.56×	1.79×
Params (Conv.) ↓	–	1.06×	1.77×	1.80×	1.66×	1.80×
Params (Total) ↓	–	1.01×	1.05×	1.05×	1.04×	1.05×

†: Keep the first convolution of each layer intact.

Bold numbers indicate the best performance of all the sparse models.

Table 1: Classification performance of different sparse VGG16 models on ImageNet. A model is denoted by $sc_x y$ where $x \in \{+\times,+\}$ indicates how sparse kernels are applied and y specifies which conv. layers are sparsified. $sc_{+\times}$ means the two complementary sparse kernels alternate while sc_{+} indicates only the sparse kernel '+' is used.

Indigo images. A person or face is marked as difficult if it's heavily occluded or too small for the annotator to discern. Difficult objects are excluded in our evaluation.

Training Dataset. *GoPro* and *Indigo* are used for evaluation purpose only in our experiments. We annotated the VOC2007 and VOC2012 datasets [4] separately for training. For each annotated person in the datasets, we labeled 5 additional body parts including face, head, head-shoulders, torso and legs. We used all the training and validation images in both datasets for training our detectors.

4.2 Experimental Setup

The primary focus of our work is how to make a faster RCNN detector compact and efficient. Although most of the experiments and analysis in this section are based on VGG16, our approach is applicable to any CNN under the faster RCNN framework.

We trained all the sparse CNN models from scratch using the identical hyper-parameters of a baseline network such as momentum and weight decay, initial learning rate, batch size, etc. The learning rate is reduced by 10 each time the validation error reaches a plateau. We augmented data by flipping horizontally and pre-processed data with mean subtraction.

We adopted the multiple-phase training scheme for faster RCNN, and fine tuned all the detectors under the default settings. The only exception is that we added two more scales (i.e. 32 and 64) in RPN in order to better detect small objects like face.

In what follows, we conduct extensive experiments to evaluate our approach on both the classification and detection tasks using the ImageNet dataset and our own body camera data.

4.3 Performance Evaluation of Classification

We first look at the classification performance of our proposed *sparse convolution* method on ImageNet. To better understand how the sparsity of a model affects its performance, we explored different combinations of the two sparse kernels (see Section 3.1 for details) based on the VGG16 network structure. This includes three nets with sparse convolution applied to the first three conv. layers ($sc_{+\times}123$), the last three conv. layers ($sc_{+\times}345$) and all the conv. layers ($sc_{+\times}12345$) in VGG16 [1]. We also tested another model ($sc_{+\times}\overline{12345}$), which

[1] VGG16 has 5 conv. layers, each of which contains $2 \sim 3$ convolutions. When we say that *sparse convolution* is applied to a conv. layer, it means all the full kernels in that layer is substituted by sparse kernels.

data	detection	*ref. (VGG16)*	$sc_{+\times}123$	$sc_{+\times}345$	$sc_{+\times}12345$	$sc_{+\times}\overline{12345}^{\dagger}$	$sc_{+}12345$
GoPro	person	77.36	71.41	73.77	72.56	**74.91**	74.69
	face	54.18	53.21	52.21	51.91	**55.30**	52.92
Indigo	person	57.60	55.69	**56.29**	53.98	55.43	54.87
	face	81.11	81.16	81.83	82.72	81.47	**83.54**

†: Keep the first convolution of each layer intact.

Bold numbers indicate the best performance of all the sparse models.

Table 2: Detection Performance (in AP) of sparse-VGG16-based faster-RCNNs. Please refer to Table 1 for the naming convention for the models.

is similar to $sc_{+\times}12345$, but keeps the first convolution of each layer intact. In all of these models, we alternate the two sparse kernels, $\mathbf{W}_{\mathbf{odd}}$ (i.e. $+$) and $\mathbf{W}_{\mathbf{even}}$ (i.e. $\times$), to replace the convolutions in VGG16.

Table 1 lists the top-1 and top-5 accuracies as well as the reductions in model parameters and FLOPs of different sparse models. First of all, all the sparse models are observed to perform comparably to the referenced model, suggesting that *sparse convolution* provides strong approximation of the VGG16 model. With full convolutions kept at the higher layers, $sc_{+\times}123$ achieves the best top-1 and top-5 classification results, but it only gains moderate savings in model size and computation. On the other hand, sparsifying all the convolutional layers seems to hurt classification performance, as indicated by the 2.5% point loss in the top-1 accuracy of $sc_{+\times}12345$. However, by increasing only 8% more parameters, $sc_{+\times}\overline{12345}$ yields better accuracy than $sc_{+\times}12345$.

To validate the effectiveness of complementary kernels, we trained a model solely based on $\mathbf{W}_{\mathbf{odd}}$ ($sc_{+}12345$). As expected, compared with the other models in Table 1, $sc_{+}12345$ is inferior, especially on the top-1 accuracy.

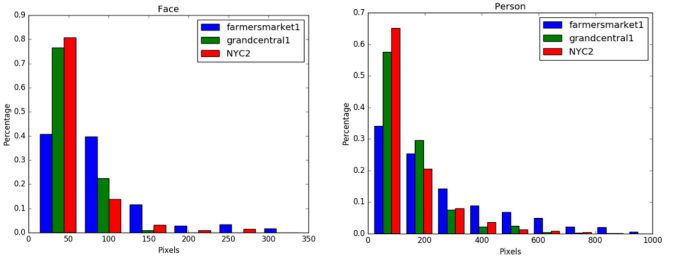
4.4 Performance Evaluation of Detection

Below we analyze the contribution to detection performance of each component in our proposed sparse feature representation. The performance is measured by the widely accepted metric for object detection, i.e. Average Precision (AP). We focus on face and person only in our evaluation.

Sparse Convolution. We first compared the performance of different sparse VGG16 models discussed above when they are used as feature extractors for faster RCNN. At this stage, the ROI pooling layer and FC layers remain the same as those of the baseline detector. The results are provided in Table 2.

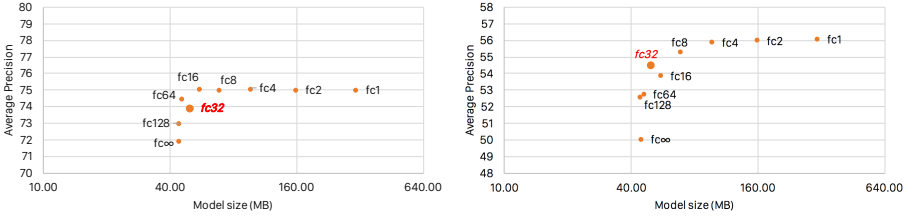
From Table 2, it's noted that the sparse models behave differently on the detection task from classification. For example, $sc_{+\times}123$ is the best performed classifier; however it produces less competitive results on the *GoPro* dataset, seeing a drop of almost 6% point in person detection. Among all the detectors, $sc_{+\times}\overline{12345}$ is the only one that performs consistently well on both datasets, achieving comparable results on person detection and even slightly better face results than the referenced detector.

The face performance on the *GoPro* data and the person performance on the *Indigo* data are beyond satisfactory. This is largely due to the limitations of faster RCNN, which is not robust in handling small objects (*e.g.* faces on *GoPro*) and heavily occluded or partial objects (*e.g.* persons on *Indigo*).



a) Face b) Person

Figure 5: Size distributions of *Face* and *Person* on the *GoPro* data by ***width*** of a bounding box. Bin size: a) *56* pixels and b) *112* pixels.



(a) Person (b) Face

Figure 6: Detection performance v.s. model size based on the *GoPro* data. Sparse ROI pooling is applied. Here fc_k denotes a model which has the number of neurons at each FC layer reduced to $4096/k$. Note that $k = \infty$ is a special case where no FC layer is used in a detector.

Sparse ROI Pooling. We implemented *sparse ROI pooling* based on $sc_{+\times}\overline{12345}$ due to its superiority to other models on detection. *Sparse ROI pooling* can skip either the odd-indexed (*sp1*) or even-indexed *sp2* features. Table 3 provides the slightly better results from *sp2* only, indicated here by $sc_{+\times}$-*sp2*-fc_1.

Based on Table 3, it can be observed that *sparse ROI pooling* demonstrates clear advantages over the regular max-pooling ($sc_{+\times}$). It does face detection better than $sc_{+\times}$ on both datasets while performing similarly on person detection. The results are justifiable, since the majority of the *GoPro* faces are less than 100 pixels in width (see Fig. 5 for details). With a close look at the results at each scene in the *GoPro* data, the performance improvement of *sparse ROI pooling* mostly comes from grandcentral1 and NYC2, on which small faces dominate.

In addition to improving accuracy on small objects, *sparse ROI pooling* alone leads to about a 10% computational saving over the baseline ($sc_{+\times}$) (from 1.39 to 1.53) and a 40% reduction in model size.

Effects of reduced FC layers. Low-capacity FC layers are highly desired for making a small-size detector. We experimented with different numbers of neurons in the FC layers and analyzed their effects on detection accuracy. Let N_{fc} be the number of neurons at a FC layer. For convenience, we assume that N_{fc} is configured to be the same for each FC layer. A model is denoted by fc_k if N_{fc} is reduced by a factor of k, i.e. $N_{fc} = 4096/k$.

Figure 6 shows that the detection accuracy is not very sensitive to N_{fc} till $k = 32$ for person and $k = 8$ for face, where there are an accuracy drop more than 1% point. We chose $k = 32$ as the sweet spot for the capacity of FC layers, and provided more results in Table 3 for

data	detection	*ref. (VGG16)*	$sc_{+\times}$	$sc_{+\times}$-*sp2-fc*$_1$	$sc_{+\times}$-*sp2-fc*$_{32}$	$sc_{+\times}$-*sp2-fc*$_\infty$
GoPro	*person*	77.36	74.91	**74.97**	73.92	71.86
	face	54.18	55.30	**56.05**	54.52	50.02
Indigo	*person*	57.60	**55.43**	54.86	54.29	51.78
	face	81.11	81.47	**82.16**	81.13	80.23
Model Size (MB)		522	500 (1.04×)	308 (1.70×)	49 (10.58×)	45 (11.67×)
FLOPs ↓		–	1.39×	1.53×	1.78×	1.79×

Bold numbers indicate the best performance of all the sparse models.

Table 3: Detection performance of sparse-VGG16-based faster RCNNs using sparse ROI pooling and reduced FC layers. Here for clarity, $sc_{+\times}$ is a short name for $sc_{+\times}$*12345*. *sp2* represents the case that the odd-indexed features are skipped in ROI pooling. fc_k indicates that the number of neurons at each FC layer is reduced to $4096/k$. $k = \infty$ is a special case where no FC layers are used in a detector.

data	detection	*ref. (AlexNet)*	*Song* [6]	*SSL1* [20]	*SSL2* [20]	$sc_{+\times}$	$sc_{+\times}$-*sp2-fc*$_8$	$sc_{+\times}$-*sp2-fc*$_{32}$
GoPro	*person*	65.28	61.04	57.74	63.70	**65.09**	64.29	63.58
	face	44.30	41.35	38.37	42.79	41.96	**43.61**	41.79
Indigo	*person*	52.84	48.30	48.56	**52.95**	52.59	51.43	51.71
	face	73.93	75.21	**76.23**	75.00	75.58	73.80	74.89
Model Size ↓		–	8.86×*	2.28×*	1.91×	1.02×	12.70×	18.60×
FLOPs ↓		–	–†	–†	–†	1.18×	3.49×	4.11×

*: An additional table to record locations of zeros is not included.

†: FLOPs is not measurable since those sparse models need specialized implementations.

Bold numbers indicate the best performance of all the sparse models.

Table 4: Comparisons of our proposed detectors with other approaches. The baseline detector is based on AlexNet. Please refer to Table 3 for the naming convention for the detectors.

analysis. The gain from the reduced FC layers is substantially rewarding. When combined with sparse ROI pooling, $sc_{+\times}$-*sp2-fc*$_{32}$ is only 49MB, 10× smaller than the baseline model (500MB). In addition, the detector achieves a speedup by a factor of 1.8×.

Now one question arises: are the FC layers actually needed for object detection? To answer this question, we completely remove the FC layers, and connect the ROI pooling layer directly to the classification layer in faster RCNN. By doing so, the classifier becomes a linear regressor, denoted by $sc_{+\times}$-*sp2-fc*$_\infty$ in Table 3. As shown in Table 3, the linear regressor is notably less powerful than fc_{32}, suggesting that the FC layers are critical for robust detection, though a large capacity may not be needed.

4.5 Performance Comparisons With Other Approaches

We compared our approaches with two recently developed techniques of sparse representation. The first one [6] reduces the size of a model by directly pruning small weights while the second one [20] learns to sparsify groups of weights by structural regularization. Both of them achieve high model compression rates without losing much accuracy on the ImageNet classification task. Here AlexNet was used as the baseline for comparison as only this model is publicly available from these approaches. We trained faster RCNN with the sparse AlexNet models from [6], [20] and our own approach, similar to what's done in Section 4.4. Nonetheless, we disabled updating of any zero weight when fine tuning the models from [6]

data	VGG16$sc_{+\times}$-*sp2*-fc_{32}	YOLO	SSD
GoPro	75.21	61.22	73.24
Indigo	54.29	55.77	59.26
Model Size (MB)	49	194	104

Table 5: Performance comparisons with YOLO and SSD.

and [20] in order to maintain their original sparsity.

Table 4 provides the results of different faster RCNN detectors based on the sparse AlexNet models. Our approach is better than both Song *et al*. [6] and SSL1 [20] on the *GoPro* data. While performing slightly worse than SSL2 on the *Indigo* data, our detector is $6 \sim 9$ times smaller than SSL2, and should run more efficiently as well. Overall, our approach demonstrates clear advantages over [6] and [20] in terms of model size and efficiency.

We further compared our approach with two real-time object detectors, YOLO [17] and SSD [16], on pedestrian detection. To make the comparisons fair, we used their best models trained with the union of VOC2007 and VOC2012 *trainval*, the same training set for our model. As shown in Table 5, our approach achieves better *GoPro* results than YOLO and SSD, but worse on *Indigo*. However, our model is $2\times$ smaller than SSD and $5\times$ smaller than YOLO.

5 Conclusion

We have presented a novel sparse feature representation for faster RCNN and applied it for object detection on wearable cameras. We demonstrate the effectiveness of our approach on person and face detection using two challenging body camera datasets collected at various scenarios. Our approach reduces a VGG16-based faster RCNN detector by a factor of over $10\times$ while still performing comparably against the baseline detector. In addition, our detector achieves a computational speedup by nearly $2\times$.

References

[1] Lisa Brown and Quanfu Fan. Enhanced face detection using body part detections for wearable cameras. In *International Conference on Pattern Recognition (ICPR)*, 2016.

[2] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision (ECCV)*, pages 354–370. Springer, 2016.

[3] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[4] 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 (IJCV)*, 111(1):98–136, January 2015.

[5] Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[6] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*, 2015.

[7] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *International Symposium on Computer Architecture (ISCA)*, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[10] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*, 2016.

[11] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, Song Han, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. In *International Conference on Learning Representations (ICLR)*, 2016.

[12] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*, 2014.

[13] Yi Li, Kaiming He, Jian Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[19] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[20] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning Structured Sparsity in Deep Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[21] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision (ECCV)*, 2016.