Improving a real-time object detector with compact temporal information

Martin Ahrnbom
Lund University
martin.ahrnbom@math.lth.se

Morten Bornø Jensen
Aalborg University
mboj@create.aau.dk

Kalle Åström
Lund Univeristy
kalle@maths.lth.se

Mikael Nilsson
Lund University
micken@maths.lth.se

Håkan Ardö
Lund University
ardo@maths.lth.se

Thomas Moeslund
Aalborg University
tbm@create.aau.dk

Abstract

Neural networks designed for real-time object detection have recently improved significantly, but in practice, looking at only a single RGB image at the time may not be ideal. For example, when detecting objects in videos, a foreground detection algorithm can be used to obtain compact temporal data, which can be fed into a neural network alongside RGB images. We propose an approach for doing this, based on an existing object detector, that re-uses pretrained weights for the processing of RGB images. The neural network was tested on the VIRAT dataset with annotations for object detection, a problem this approach is well suited for. The accuracy was found to improve significantly (up to 66%), with a roughly 40% increase in computational time.

1. Introduction

Neural networks designed for real-time object detection using a single image as their input have recently improved significantly. Detectors like SSD [14], SqueezeDet [26] and YOLOv2 [18] outperform previous real-time detectors while approaching the accuracy of slower methods like those based on Faster R-CNN [19]. It might thus be tempting to use real-time detectors directly, but in practical problems there is often more information available than these networks take advantage of. For example, when detecting objects in videos, looking at only a single frame at the time is bound to make detection more difficult; humans have access to all we have seen before a given moment to help us detect various objects, and this information could be particularly helpful for occluded or small objects, hard to distinguish in a single frame. Commonly used datasets for object detection like COCO [13] and PASCAL VOC [5] only contain stand-alone images which algorithms are supposed to find objects in. This has led to strong development of algorithms and networks designed for this particular task. The use of temporal information in neural networks for object detection is not as well explored.

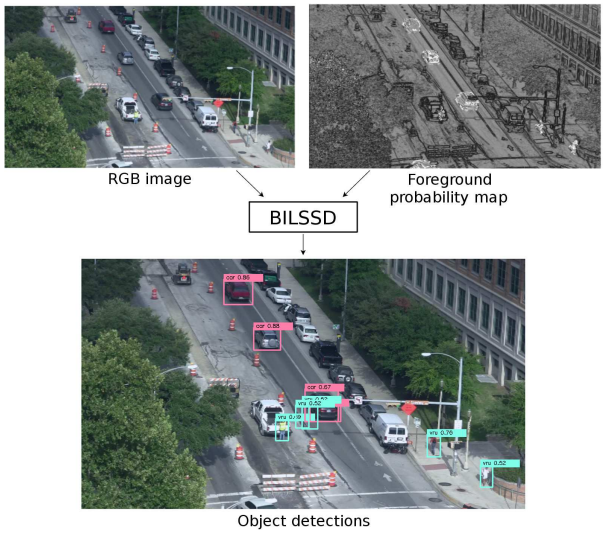

Figure 1. The BILSSD network takes an RGB image and a corresponding foreground probability map as input to produce object detections.

Taking advantage of temporal information in object detectors is not trivial. End-to-end learning is currently often the preferred way of solving computer vision and deep learning problems, but in the case of videos, that approach is not ideal. Feeding multiple frames directly into a Convolutional Neural Network (CNN) is problematic, as the amount of data to be processed by the network grows large if more than a few frames are to be considered. Recurrent Neural Networks (RNN's) can learn to process videos, but this only solves part of the problem; in order to properly train the RNN, it should be unrolled to allow backpropagation "through time" which also uses a large amount of memory during training if a large number of frames are to be consid-

ered. If there was a compact way to represent temporal information gathered from a large number of frames, a faster and simpler approach would be to feed that data alongside standard RGB pixels into a single-frame object detector.

In the case where videos are filmed by a static camera, a foreground detector like the one by Ardö and Svärm [1] can be used to compute a per-pixel foreground probability map. Going from RGB to Red-Green-Blue-Foreground (RGBF) adds only a single input layer, increasing the amount of data to feed into the network only by $1/3$ while providing useful temporal information. Compared to using an RNN, some generality is lost, as any temporal information other than what is considered foreground and background cannot be learned, and the videos have to be filmed by a static camera. What is gained is the simplicity and speed of being able to re-use existing and optimised single-frame object detectors as a starting point. Compared to using a single-frame object detector directly, temporal information is gained without sacrificing real-time performance.

Using foreground detection or background subtraction for object detection is a well-established concept, and used to be a popular approach for object detection. With the recent improvements to object detection CNN's, methods relying on foreground detection are no longer considered state of the art. However, this does not necessarily imply that these kinds of data cannot improve the performance of object detectors.

In other problems, other kinds of data might be available that can be expressed as an additional input layer. For example depth information, which has become more commonly available thanks to products like the Kinect, and thermal cameras that are sometimes used as a complement to RGB. The network design for including additional input layers does not need to make strict assumptions on the type of data it will process, as long as it somewhat resembles an image and is spatially correlated to the RGB layers.

For a network using RGB and additional modalities to be practically viable, unless a very large and varied annotated multimodal dataset is available for pretraining, it is necessary to be able to reuse RGB pretraining on the part of the network that is to process RGB data. It is also beneficial if the network can easily be constructed from any RGB-only object detector, such that if a better single-frame object detector is designed in the future, a corresponding improved mutlimodal version is easy to construct.

For practical object detection problems, before using a standard single-frame RGB object detector, one should ask if any additional data is available that could significantly help the detector perform its task, like temporal information. If so, a network design is needed that allows the use of this additional information, preferably without sacrificing the recent improvements of fast single-frame RGB object detectors. This paper proposes such a network.

The main contributions of this paper are:

- Bonus Input Layer Single-Shot multibox Detector (BILSSD), a novel neural network design based on SSD which can utilise both RGB and and additional data, like a foreground probability map, for object detection (Section 3)
- A set of annotations designed for object detection for some frames in the VIRAT [16] video dataset (Section 6.1)

2. Related work

Multimodal object detection has been attempted before. For example, Viola *et al*. [24] and Jones and Snow [10] propose object detectors for videos using both spatial and temporal information that are not based on deep learning, distancing themselves from today's state-of-the-art approaches. Similarly, Gould *et al*. [6] used RGB along with depth images for detecting household objects. Like BILSSD, features were calculated from both modalities separately, allowing some pretraining to be done on larger RGB-only datasets, but it was also not based on deep learning. Further from BILSSD's approach, Javed *et al*. [9] and Bang *et al*. [2] propose methods not based on deep learning using only temporal data (recurrent motion images and adaptive background subtraction images, respectively) for object detection.

One way of using temporal information for object detection is via recurrent neural networks. Ning *et al*. [15] suggests a method for adding recurrent layers to an existing single-frame object detector to do simultaneous object detection and tracking. The high level features and detections from the single-frame object detector are fed into LSTMs that are trained to make spatially and temporally consistent detections. Because the recurrent layers operate on high level features, there is no ability to learn low-level motion features like separating the foreground from background. Such low-level features will not be brought up to high-level layers, as the single-frame object detector is first trained on its own, before the training of the recurrent layers.

Using temporal information for generating object proposals has been done in a few ways. Tripathi *et al*. [23], Sharir *et al*. [21] and Oneata *et al*. [17] propose different methods for creating object proposals in videos, not only spatially but also temporally. Those object proposals can then be evaluated by a CNN to do full object detection in videos. These approaches differ significantly from BILSSD, as it does not rely on separate object proposals, which by necessity is computationally redundant as the tasks of finding and classifying objects are intimately connected.

Many neural networks use temporal information in videos for various other computer vision tasks. Yeung *et*

al. [27] propose a method for finding the times for certain actions in short videos by feeding multiple frames into an RNN. Karpathy *et al*. [11] explore multiple ways of utilising temporal information for classifying entire videos, by comparing early, late and "slow" fusing strategies. The "slow" strategy fuses features in multiple steps, including some fusion in the middle of the network, somewhat similarly to BILSSD. Donahue *et al*. [4] propose an RNN for image retrieval and caption generation in videos. Closer to BILSSD's approach, Simonyan *et al*. [22] propose a neural network that process both RGB frames and optical flow differences between frames separately and classify videos by a late fusion of features from both modalities.

There have been deep neural networks that tackle the problem of foreground segmentation in videos, like Caelles *et al*. [3]. It differs from traditional foreground segmentation algorithms in that it only segments a single foreground object, which has to be annotated manually in one frame. Another recent attempt at foreground segmentation is a neural network proposed by Jain *et al*. [8], which does not utilise temporal information.

Gupta *et al*. [7] train neural networks with depth data alone and in addition to RGB. They also take advantage of existing RGB networks by splitting the depth channel into three channels in an attempt to mimic the structure of RGB, and then retraining an existing RGB R-CNN detector on this new input data. This allows the re-use of an existing network design and pretrained weights, but they were not able to improve the results by fusing the modalities inside the network; instead they propose running two separate detectors and fusing their output.

In conclusion, many research approaches have tried to use temporal or otherwise multimodal input data for various vision tasks, including object detection, but none of them have made an object detector based on modern real-time neural networks, that combine RGB and temporal data in a "deep fusion" way, while being able to largely re-use the network design, and pretraining for the RGB processing layers.

3. BILSSD

This section describes a deep neural network design called Bonus Input Layer Single-Shot multibox Detector (BILSSD) based on Single-Shot multibox Detector [14]. The main difference is that BILSSD takes four input layers instead of three (RGBF instead of RGB, in our experiments). In order to be able to re-use initial layers pretrained for RGB images, the fourth input layer is processed separately by similar convolutional and pooling layers. The only difference in these layers is that the number of output features per layer is reduced by half, a design choice made on the assumption that the additional data can be represented by fewer features compared to RGB images. All features are then merged by three convolutional layers before being fed into the detection part of the SSD network. See Figure 1 for a basic overview, and the top part of Figure 2 for a more detailed description of the network. The design can be described as a "deep fusion", which differs from both "early fusion" and "late fusion" as the network processes the data both before and after the fusing of modalities.

Since the primary purpose of this network is to show the usefulness of providing additional input data, no other redesigns of the network, compared to standard SSD, are made.

BILSSD's concept of "deep fusion" is not inherently tied to SSD's design. Any similar deep neural network designed for object detection, for example YOLOv2 [18] and SqueezeDet [26], should be possible to modify in a similar way. The detection part of the SSD network is in BILSSD's implementation completely unchanged, and the processing of additional data is very similar to the processing of RGB, so making similar changes to any similar detector should be straightforward.

BILSSD's design is not inherently bound to some specific type of additional data; as long as the data can be expressed as a single layered image that is spatially consistent with the RGB data, it can be used with BILSSD, although minor changes like the number of output features from the layers processing the additional data may improve results, depending on the type of data.

4. Pretraining foreground feature extraction

While the first few layers that process RGB images based on VGG-16 can utilise existing pretrained weights to initialise the training process, no corresponding weights exist for the layers that process the foreground probability maps. It was initially tested to train the BILSSD network with pretraining for the RGB layers, and randomly initialised weights for the others layers. The network was then found to prefer only using RGB features. To work around this, a simple neural network was designed, which shares the initial layers with BILSSD's initial layers that extract features from foreground probability maps. The task of this simple network is to, given a foreground probability map, produce a 4×4 grid of values between 0 and 1, where high values indicate high confidence that an annotated object (of any class) exist in the corresponding 16th of the image, and low values indicate the opposite. An example of what output from the simple network can look like can be seen in Figure 3.

Converting existing ground truth to this format is straight-forward, by marking the cell in the 4×4 grid containing the center coordinates of each annotation's bounding box as a 1, while all others are set to 0. The simple network is designed to learn to find objects rather than to classify them. The idea behind this design is that the fore-

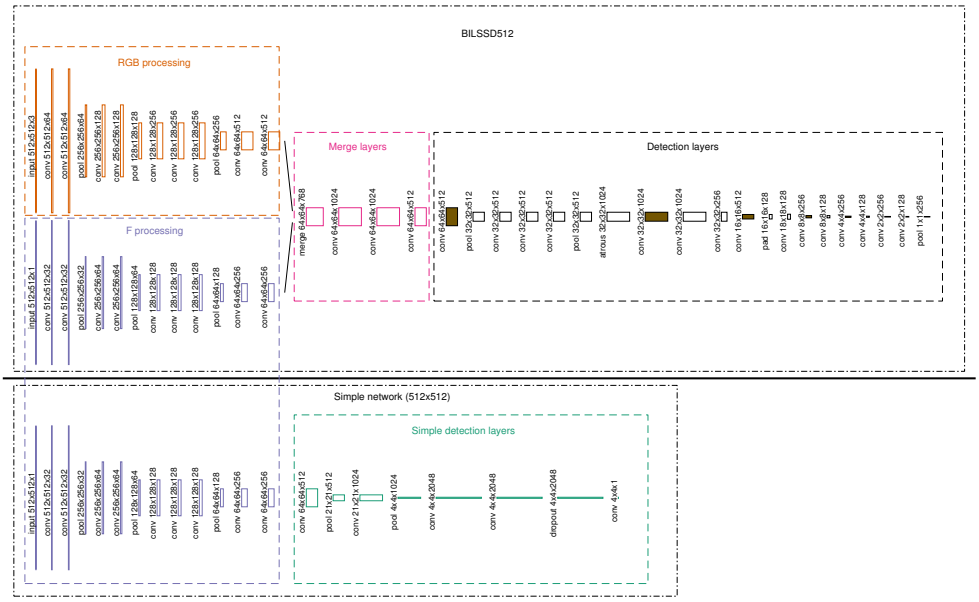

Figure 2. In this schematic, data flows from left to right. For each layer in the network, drawn as a box, the output dimensions are drawn on the left. Above horizontal black line is the network design of BILSSD512. RGB images are processed by the RGB processing layers (orange), and F are in parallel processed by the F processing layers (purple). Features from both are merged and processed by the merge layers (magenta). These are followed by the detection layers (black), which are identical to those in standard SSD. The boxes used as input into the SSD detectors (not shown in this visualisation) are filled brown. If one were to remove the F processing and merge parts, the result would be the standard SSD network. Note that the input and output of the merging layers are identical, meaning that no change to the detection layers was necessary. Below the horizontal black line is the "simple" network, which shares its initial layers with the F processing layers of BILSSD. The simple detection layers (green) do simplified object localisation and bring the resolution down to 4×4, which is the output of the simple network.

ground probability maps may be better suited for localisation rather than classification.

After training this simple network, the weights for the layers that overlap with the processing of the additional input data in BILSSD are used as pretrained weights. This approach should help BILSSD utilise the additional input data. For a detailed description of the simple network for processing 512×512 images, see the bottom part of Figure 2. When processing 300×300 images, the only difference to when processing 512×512 images is the last pooling layer which pools a 3×3 region rather than 5×5, to bring the resolution down to the same 4×4.

5. BGGRAD foreground detection

The foreground detection algorithm used in this paper is BGGRAD, as described in Ardö *et al*. [1]. This algorithm generates a single-layer probability map where dark pixels indicate a high probability of background, white pixels indicate a high probability of foreground and grey areas are regions where the algorithm is not certain. Because the algorithm is based on matching gradients directions in different frames, areas with little or no gradients, like flat surfaces (generally in the interior of objects) will appear grey. This means that, in general, foreground objects appear grey with white outlines while background objects appear grey with black outlines. A few examples can be seen in Figure 4. This allows the shapes of objects to remain visible, and could help the network in separating the different objects when looking at only the foreground probability maps.

The algorithm's main limitation, like most foreground detection methods, is that it relies on the camera being stationary. When the camera shakes, background objects will appear as foreground. It is also somewhat sensitive to heavy compression artifacts, as edges between blocks of pixels compressed separately may appear as foreground.

6. Experiments on VIRAT

A Keras implementation[1] of BILSSD was trained on the VIRAT dataset using annotations designed for object detection (see Section 6.1). The output from the BGGRAD

[1]The implementation is based on a port of SSD to Keras available here: https://github.com/rykov8/ssd_keras

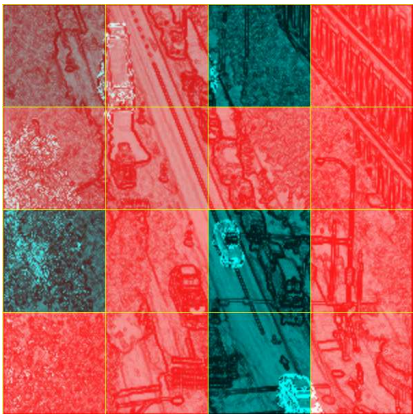

Figure 3. An example of output from the simple network. Blue regions (dark regions, if viewed in monochrome) means high confidence for annotated objects appearing somewhere in the region, while red (brighter, if viewed in monochrome) means a low confidence. These colours are drawn over the 300×300 foreground probability map that the simple network receives as input. The image is rescaled to this size before being processed by the foreground detection algorithm. In this example, the simple network is able to correctly detect two cars and a pedestrian, but also believes the moving tree to be an annotated object.

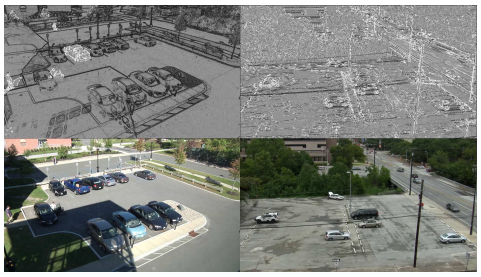

Figure 4. Two examples of the BGGRAD algorithm after running on videos from the VIRAT dataset. At the bottom are the RGB inputs, and above them are the corresponding foreground probability maps. On the right, is an example of what the output looks like when the algorithm runs on a shaky video, where all edges appears as foreground.

foreground detection algorithm [1] was used as the fourth input layer. For this task, BILSSD was trained and evaluated using RGBF, only RGB and only F. This allows some analysis of how much the different modalities help in object detections. When only using one modality, this was implemented by feeding only zeroes as input to the other modality's processing layers. In the case where only RGB is used, BILSSD should behave nearly identical to the standard SSD network in terms of accuracy, as the only difference is the additional "merging" layers that are assumed to affect the end result at most marginally, as they should quickly learn to only include RGB features.

In these experiments, both the 300×300 and 512×512 versions of SSD were used as the base for BILSSD, and the two versions are labelled "BILSSD300" and "BILSSD512". Images were scaled down to 300×300 and 512×512 respectively before going through the background detection algorithm. To generate the foreground probability maps, all frames in the videos were fed into the foreground detection algorithm. The frames where annotations exist were saved, and used in training.

6.1. VIRAT annotations for object detection

The VIRAT dataset [16] is designed for event recognition, and thus its official annotations only mark certain pedestrians and vehicles that are part of the annotated events. The dataset is however a large collection of surveillance videos filmed with, for the most part, stable cameras, making it a good benchmark for object detections that work in such a context. We have made third-party annotations for the task of object detection, where most visible objects of the following two classes are annotated by bounding boxes:

- "vulnerable road users" ("VRU's", such as pedestrians, bicyclists)
- "vehicles" (four-wheeled vehicles like cars, buses, trucks)

A total of 1240 frames have been annotated, from 62 different videos in the VIRAT dataset. Half of those videos make up the training set, while the other half is used for evaluation. In total, there are 2733 VRU's and 721 vehicles annotated, with 1368 VRU's and 339 vehicles in the training set, and 1365 VRU's and 382 vehicles in the test set. The annotations are made to resemble data used in traffic surveillance analysis, for example only vehicles that are not parked are annotated. This, along with a large number of small pedestrians that likely appear more clearly in foreground probability maps, makes utilising temporal information a promising approach for this challenge. On the other hand, there are frames in the dataset where camera shake cause the foreground detection algorithm to produce bad results. These annotations are available here: https://github.com/ahrnbom/ViratAnnotationObjectDetection.

6.2. Training

First, the simple network was trained on foreground probability maps using randomly initialised weights, for 30 epochs. This procedure was repeated until a good initialisation allowed convergence. The network was tested with these weights on some samples from the dataset and its output was inspected visually to make sure the simple network had learnt to detect objects. This was done for both 300×300 and 512×512 foreground probability maps.

For all three variants (RGBF, RGB and F) the BILSSD300 and BILSSD512 networks were trained for 100

epochs using an Adam optimiser [12] with a base learning rate of 3×10^{-4}. The batch size was 16 for BILSSD300 and 8 for BILSSD512. The BGGRAD foreground detection algorithm was set to look at the previous 100 frames for computing the foreground probabilities, and it processes blocks of 8×8 pixels at the time.

For the RGB processing layers, pretrained weights from ImageNet [20] were used, while the F processing layers used weights from the simple network as described in Section 4. The merging and detection layers had random weight initialisation.

During training, data augmentation was performed by horizontally flipping the images with a probability 0.5, varying saturation, brightness, contrast and lighting over RGB images, while adding random noise with an amplitude of 10% of the value range to the foreground probability maps. Additionally, random cropping was performed with an aspect ratio between $3/4$ and $4/3$ and the area of the cropped section was between 75% and 100% of the original images.

All variants (RGBF, RGB and F) were trained for 100 epochs each, which took around 3 hours for SSD300 and 6 hours for SSD512.

6.3. Results

6.3.1 Accuracy

The mAP scores for BILSSD300 and BILSSD512 for the VIRAT dataset can be seen in Table 1, and corresponding precision-recall curves for the two classes can be seen in Figure 5. In short, using RGBF outperforms using only RGB (which should behave similarly to standard SSD) or only F in terms of accuracy. The accuracy improves for both the tested input resolutions, by 66% and 31% respectively.

	RGBF	RGB	F
BILSSD300	0.272	0.208	0.154
BILSSD512	**0.400**	0.241	0.125

Table 1. mAP scores for different input resolutions and modalities of BILSSD on the VIRAT dataset. Bold number indicates best result.

6.3.2 Qualitative analysis

Some output from the different BILSSD networks trained on RGBF, RGB and F were inspected manually. It was found that when trained with only F or only RGB, correct detections were given only marginally better confidence values than a large number of incorrect detections. They all had problems with giving false positives relatively high confidences, around 0.40 for RGBF and around 0.44 for F and RGB, while true detections vary between 0.4 and 0.9 for RGBF but for F and RGB they only rarely get above 0.5. Using only RGB, confidences above 0.5 were more common than when using only F, explaining the low mAP scores of the latter. True positives of the VRU class generally got lower confidences than of the vehicle class, likely due to the smaller objects being harder to detect.

Comparing 300×300 and 512×512 versions of RGBF, the higher resolution was found to help detecting small objects. They both had problems with outputting more than one box near real objects, not quite close enough to be caught by the non-maximum suppression, and this issue is more noticeable in the lower resolution. Failing to localise objects did occur, as well as some false positives, but incorrect classifications were uncommon.

6.3.3 Execution speed

For the computer used in these experiments, which is equipped with a NVIDIA Titan X GPU and an Intel Core i7-6800K CPU, the execution times for a batch size of 1 can be seen in Table 2. These times can be compared to the original SSD's reported execution speeds on the same GPU model and batch size, which were 46 FPS and 19 FPS for SSD300 and SSD512 respectively. One should note that the original SSD implementation was done in Caffe, while BILSSD's implementation was done in Keras with a Tensorflow backend and the computers' other differences may have some impact, so the numbers are not perfectly comparable. However, using these numbers, BILSSD (including the BGGRAD preprocessing) has roughly 40% more processing time compared to SSD.

	BILSSD300	BILSSD512
Time BILSSD	0.029 s	0.055 s
Time BGGRAD	0.0023 s	0.019 s
Time both	0.031 s	0.074 s
FPS BILSSD	34 FPS	18 FPS
FPS BGGRAD	430 FPS	52 FPS
FPS both	32 FPS	14 FPS

Table 2. Execution times and frame rates for BILSSD and BGGRAD on 300×300 and 512×512 resolutions. These times were computed as an average over more than 100 frames.

7. Conclusions

We have introduced the BILSSD network, which is based on SSD while adding the ability to utilize multimodal spatially aligned input. We have tested it on the VIRAT dataset, using foreground probability maps computed by the fast BGGRAD foreground detection algorithm as the additional input, along with RGB images.

On this dataset, accuracy increases when RGB and F are used together, compared to using only RGB and using only

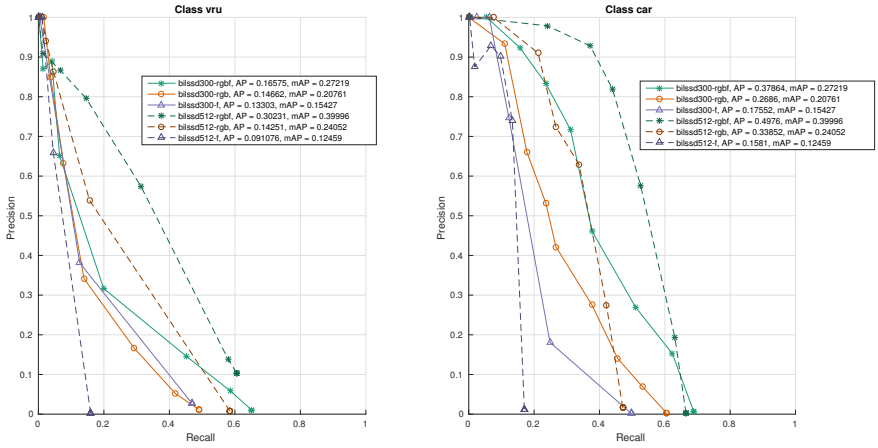

Figure 5. Precision-recall curves for the VIRAT dataset. Using RGBF is better than using only RGB or only F for both resolutions, and the improvement is significant in all cases except for the VRU class in lower resolution, where the improvement is marginal. Higher resolution is better than lower resolution for RGBF and RGB, but surprisingly not for only F, which performs poorly overall.

F, for both the VRU and vehicle classes. The improvements are expected, as it is difficult to tell the difference between a parked and non-parked car without temporal data, and small pedestrians appear more clearly in the foreground probability maps. For example, BILSSD300 using RGBF outperforms BILSSD512 using only RGB (similar to SSD512) in terms of mAP while running much faster, so for this problem, adding temporal data is a more efficient way to improve performance than increasing the resolution. Accuracies for the VRU class are generally low for BILSSD300, which makes sense as most instances of this class are small, making them more difficult to detect in low resolution images.

Using only F performs poorly overall. Because the features learned from the F input improves RGBF significantly compared to RGB, it is obvious that these features are helpful, but on their own they do not seem to provide enough confidence for separating true from false positives. Another reason why using only F performs so poorly could be the lack of training data; while the RGB layers use the large and varied ImageNet as a starting point, the F layers have only ever looked at the limited number of images in the VIRAT annotations. Finding a better pretraining strategy for the F layers could improve not only the accuracy when using only F, but also when using RGBF.

It should be noted that the relatively low number of annotated frames used in these experiments means that one should be careful drawing too general conclusions about how much using foreground probability maps in addition to RGB improves accuracy on other datasets. What can be concluded is that the BILSSD network is capable of utilising multiple modalities to increase the accuracy of object detections, and this improvement is independent of increasing the spatial resolution of the input data.

8. Future work

There are other ways in which BILSSD could be evaluated. Most notably, it would be interesting to try the approach on more datasets. There is currently no large scale dataset of videos filmed with stationary cameras in varied environments in decent video quality. The DETRAC dataset [25] is a large scale surveillance dataset, but the camera shake in most videos prevent the BGGRAD algorithm to work properly. Developing a fast yet shake resistant foreground detection algorithm and using it with BILSSD on the DETRAC dataset could be an interesting direction for future evaluation.

The pretraining of the foreground processing layers could likely be improved. As Gupta *et al*. [7] showed, pretraining on RGB images from ImageNet can be used as an initialisation for non-RGB images with improved results compared to starting from scratch, if the data is formatted to partially mimic the structure of RGB. Perhaps a foreground detection algorithm could be developed which produces three output layers, somewhat mimicking the structure of RGB, to take full advantage of such an approach. The probability map already shares some properties with RGB, like the concept of edges around objects, so such an approach is probably feasible.

It would be interesting to try the BILSSD network using other modalities than foreground probabilities alongside RGB, like depth data or thermal images, perhaps using more than one non-RGB modality at the time. Different pretraining strategies and minor network changes may be necessary, depending on the data.

Implementing similar multimodal versions of other

object detectors, like Faster R-CNN, SqueezeDet and YOLOv2, would allow further analysis of how well the concept of deep fusion generalises. When a new and better single-frame object detector is made in the future, as long as this network can be split into a feature extraction part and a detection part, implementing a deep fusion of modalities in the style of BILSSD should be easy.

References

[1] H. Ardö and L. Svärm. Bayesian formulation of gradient orientation matching. In *Lecture Notes in Computer Science*, volume 9163, pages 91–103. Springer, 2015.

[2] J. Bang, D. Kim, and H. Eom. Motion object and regional detection method using block-based background difference video frames. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012 IEEE 18th International Conference on*, pages 350–357. IEEE, 2012.

[3] S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool. One-shot video object segmentation. *CoRR*, abs/1611.05198, 2016.

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[6] S. Gould, P. Baumstarck, M. Quigley, A. Y. Ng, and D. Koller. Integrating Visual and Range Data for Robotic Object Detection. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*, Marseille, France, Oct. 2008. Andrea Cavallaro and Hamid Aghajan.

[7] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[8] S. D. Jain, B. Xiong, and K. Grauman. Pixel objectness. *arXiv preprint arXiv:1701.05349*, 2017.

[9] O. Javed and M. Shah. Tracking and object classification for automated surveillance. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 343–357, London, UK, UK, 2002. Springer-Verlag.

[10] M. J. Jones and D. Snow. Pedestrian detection using boosted features over many frames. In *19th International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA*, pages 1–4, 2008.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[15] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang. Spatially supervised recurrent convolutional neural networks for visual object tracking. *arXiv preprint arXiv:1607.05781*, 2016.

[16] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 3153–3160. IEEE, 2011.

[17] D. Oneata, J. Revaud, J. Verbeek, and C. Schmid. Spatio-temporal object detection proposals. In *European conference on computer vision*, pages 737–752. Springer, 2014.

[18] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[19] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[21] G. Sharir and T. Tuytelaars. Video object proposals. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 9–14. IEEE, 2012.

[22] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[23] S. Tripathi, S. J. Belongie, Y. Hwang, and T. Q. Nguyen. Detecting temporally consistent objects in videos through object class label propagation. *CoRR*, abs/1601.05447, 2016.

[24] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision*, 63(2):153–161, July 2005.

[25] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv CoRR*, abs/1511.04136, 2015.

[26] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. *CoRR*, abs/1612.01051, 2016.

[27] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.