RN-VID: A Feature Fusion Architecture for Video Object Detection

Hughes Perreault[1], Maguelonne Heritier[2], Pierre Gravel[2], Guillaume-Alexandre Bilodeau[1] and Nicolas Saunier[1]

Polytechnique Montreal[1], Genetec[2]
{hughes.perreault, gabilodeau, nicolas.saunier}@polymtl.ca,
{mheritier, pgravel}@genetec.ca

Abstract. Consecutive frames in a video are highly redundant. Therefore, to perform the task of video object detection, executing single frame detectors on every frame without reusing any information is quite wasteful. It is with this idea in mind that we propose RN-VID, a novel approach to video object detection. Our contributions are twofold. First, we propose a new architecture that allows the usage of information from nearby frames to enhance feature maps. Second, we propose a novel module to merge feature maps of same dimensions using re-ordering of channels and 1×1 convolutions. We then demonstrate that RN-VID achieves better mAP than corresponding single frame detectors with little additional cost during inference.

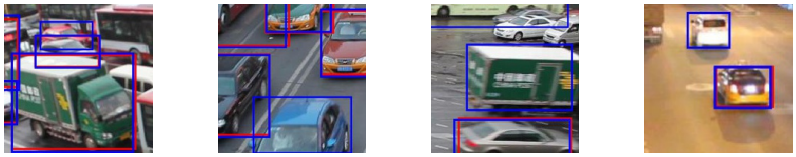
Keywords: Video object detection · Feature fusion · Road users · Traffic scenes

1 Introduction

Convolutional neural network (CNN) approaches have been dominant in the last few years for solving the task of object detection, and there has been plenty of research in that field. On the other hand, research on video object detection has received a lot less attention. To detect objects in videos, some approaches try to speed up inference by interpolating feature maps [16], while others try to combine feature maps using optical flow warping [30]. In this work, we present an end-to-end architecture that learns to combine consecutive frames without prior knowledge of motion or temporal relations.

Even though research on video object detection has been less popular than its single frame counterpart, the applications are not lacking. To name a few: autonomous driving, intelligent traffic systems (ITS), video surveillance, robotics, aeronautics, etc. In today's world, there is a pressing need to build reliable and fast video object detection systems. The number of possible applications will only grow over time.

Using multiple frames to detect the objects on a frame presents clear advantages, if used correctly. It can help solve problems like occlusion, motion blur, compression artifacts and small objects (see in figure 1). When occluded, an object might be difficult or nearly impossible to detect and classify. When moving, or when the camera is moving, motion blur can occur in the image making it more challenging to locate and recognize objects because it changes their appearance. In digital videos, compression artifacts can alter the image quality and make some parts of the frame more difficult



(a) (b) (c) (d)

Fig. 1: Qualitative examples where our model (blue) performs better than the RetinaNet baseline (red). (a) the two cars in the back are heavily occluded by the green truck, (b) the car in the bottom center is being occluded by the frame boundary, (c) the green truck is blurry due to motion blur, (d) as cars become smaller, they become harder to detect, like the white one at the top.

to analyze. Small objects can be difficult to locate and recognize, and having multiple frames allows us to use motion information (implicitly or explicitly) as a way to help us find them. Implicitly by letting the network learn how to do it, explicitly by feeding the network optical flow or frame differences.

Our model relies on the assumption that a neural network can learn to make use of the information in successive frames to address these challenges, and this paper demonstrates the advantages of such a model. Frame after frame, the object instances are repeated several times under slightly different angles, occlusion levels and illuminations, in a way that could be thought as similar to data augmentation techniques. We seek to make the network learn what is the best fusion operation for each feature map channel originating from several frames. Our proposed method contains two main contributions: an object detection architecture based on RetinaNet [15] that merges feature maps of consecutive frames, and a fusion module that merges feature maps without any prior knowledge or handcrafted features. Combined together, these two contributions form an end-to-end trainable framework for video object detection and classification.

Since this domain contains a lot of interesting challenges and applications, our evaluation is concentrated on traffic surveillance scenes. The effectiveness of our method is evaluated on two popular object detection datasets composed of video sequences, namely UA-DETRAC [28] and UAVDT [7]. We compare both with the RetinaNet baseline from which we build upon, and state-of-the-art methods from the public benchmarks of those datasets. Results show that our method outperforms both the baseline and those state-of-the art methods.

2 Related Work

2.1 Object Detection

Over the last few years, the research focus for object detection has been on single frame detectors. Deep learning based methods have been dominant on all benchmarks. The two main categories are two-stage detectors, which use a region proposal network, and single-stage detectors, which do not. R-CNN [9], a two-stage detector, was the first dominant object detector to use a CNN. It used an external handcrafted object proposal

method called selective search [25] to produce bounding boxes. It would then extract features for each bounding box using a CNN, and would classify those features using SVM. Fast R-CNN [8] builds upon this idea by addressing the bottleneck (passing each bounding box in a CNN). The way it solves this problem is by computing deep features for the whole image only once, and cropping these corresponding features for each bounding box proposals. Faster R-CNN [23] improves further more by making the architecture completely trainable end-to-end by using a CNN to produce bounding box proposals, and by performing a classification and regression to refine the proposals. R-FCN [4] improves Faster R-CNN by introducing position sensitivity of objects, and by doing so can localize them more precisely. It divides each proposal into a regular grid, and classifies each cell separately. In Evolving Boxes [26], the authors build an architecture specialized for fast vehicle detection that is composed of a proposal and an early discard sub-network to generate candidates under different feature representation, as well as a fine-tuning sub-network to refine those boxes.

Single-stage object detectors aim to speed up the inference by removing the object proposal phase. That makes them particularly well suited for real-time applications. The first notable single-stage network to appear was YOLO [20], which divides the image into a regular grid and makes each grid cell predict two bounding boxes. The main weakness of YOLO is thus large numbers of small objects, due to the fact that each grid cell can only predict two objects. A high density of small objects is often found in the traffic surveillance context. Two improved versions of YOLO later came out, YOLOv2 [20] and YOLOv3 [22]. SSD [18] tackles the problem of multi-scale detection by combining feature maps at multiple levels, and applying a sliding window with anchor boxes at multiple aspect ratio and scale. RetinaNet [15] works similarly to SSD, and introduces a new loss function, called focal loss that addresses the imbalance between foreground and background examples during training. RetinaNet also uses the state-of-the-art way of tackling multi-scale detection, Feature Pyramid Network (FPN) [14]. FPN builds a feature pyramid at multiple levels with the help of lateral and top-down connections, and performs classification and regression on each of these levels.

2.2 Video Object Detection

Here we present an overview of some most notable work on video object detection. In Flow Guided Feature Aggregation (FGFA) [30], the authors use optical flow warping in order to integrate feature maps from temporally close frames, which allows them to increase detection accuracy. In MANet [27], the authors use a flow estimation and train two networks to perform pixel-level and instance-level calibration. Some works incorporate the temporal aspect explicitly, for example, STMM [29] uses a recurrent neural network to model the motion and the appearance change of an object of interest over time. Other works focus on increasing processing speed by interpolating feature maps of intermediate frames, for instance in [16] where convolutional Long Short-Term Memories (LSTMs) are used. These previous works use some kind of handcrafted features (temporal or motion), while our work aims to train a fusion module completely end-to-end. Kim *et al.* [2] trained a model by using deformable convolutions that could compute an offset between frames. Doing so allowed them to sample features from

close frames to better detect objects in a current frame. This helps them in cases of occlusion or blurriness. In 3D-DETNet [13], to combine several frames, the authors focus on using 3D convolutions on concatenated features maps, generated from consecutive frames, to improve them. Perreault *et al.* [19] trained a network on concatenated image pairs for object detection, but could not benefit from pre-trained weights and therefore had to train the network from scratch to outperform the detection on a single frame.

2.3 Optical flow by CNNs

Works on optical flow by CNNs showed that we can train a network to learn motion from a pair of images. Therefore, similar to our goal, these works put together information from consecutive frames. FlowNet [6] is the most notorious work in this field, being the first to present an end-to-end trainable network for estimating optical flow. In the paper, two models are presented, FlowNetSimple and FlowNetCorr. Both models are trained on an artificial dataset of 3D models of chairs. FlowNetSimple consists of a network that takes as input a pair of concatenated images, while FlowNetCorr used a correlation map between higher level representation of each image of the pair. The authors later released an improved version named FlowNet 2.0 [10] that works by stacking several slightly different versions of FlowNet on top of each other to gradually refine the flow.

3 Proposed Method

Formally, the problem we want to solve is as follows: given a target image, a window of n preceding and n future frames and predetermined types of objects, place a bounding box around and classify every object of the predetermined types in the target image.

To address this problem, we propose two main contributions, a novel architecture for object detection and a fusion module to merge feature maps of the same dimensions. We crafted this architecture to allow the usage of pre-trained weights from ImageNet [5] in order to build over methods from the state-of-the-art.

3.1 Baseline: RetinaNet

We chose to use the RetinaNet [15] as a baseline upon which to build our model, due to its high speed and good performance. To perform detection at various scales, RetinaNet uses a FPN, which is a pyramid of feature maps at multiple scales (see figure 2). The pyramid is created with top-down and side connections from the deepest layers in the network, and going back towards the input, thus growing in spatial dimension. A sliding window with boxes created with multiple scales and aspect ratios is then applied at each pyramid level. Afterwards, every box is passed through a classification and a regression sub-network. Finally, non maximal suppression is performed to remove duplicates. The detections with the highest confidence scores are the ones we keep. As a backbone extractor, we used VGG-16 [17] for the good trade-off between speed and size that it offers. RetinaNet uses the focal loss for classification:

$$FL(p') = -\alpha_t(1-p')^{\gamma} log(p') \tag{1}$$

where γ is a factor that diminish the loss contributed by easy examples. α_t is the inverse class frequency, and its purpose is to give more representation to underrepresented classes during training. p' is the probability of the predicted label if it corresponds to the ground-truth label, and is $1-$the probability of the predicted label otherwise.

So if the network predicts with a high probability and is correct, or a low probability and is incorrect, the loss will be marginally affected due to those examples being easy. For the cases where the network is confident (high probability) and incorrect at the same time, the examples will be considered hard and the loss will be affected more.

3.2 Model Architecture

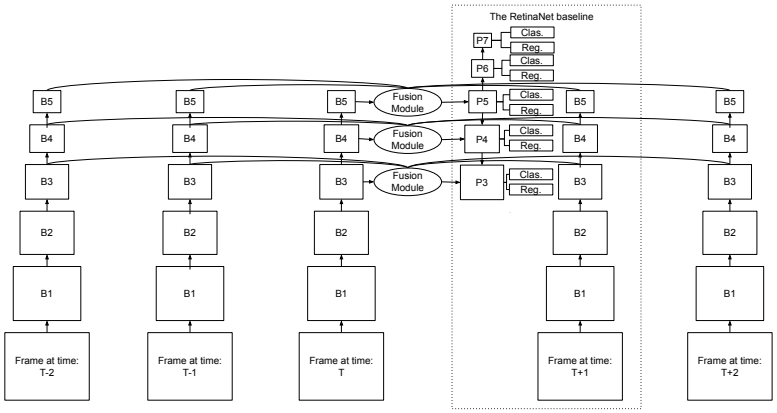


Fig. 2: A representation of our architecture with $n = 2$. Each frame is passed through a pre-trained VGG-16, and the outputs of block 3, block 4 and block 5 are collected for fusion. B1 to B5 are the standard VGG-16 [17] blocks, and P3 to P7 are the feature pyramid levels. In the dotted frame is an overview of our baseline, a RetinaNet [15] with VGG-16 as a backbone.

The main idea of the proposed architecture is to be able to compute features for every frame of a sequence only once, and to be able to use these pre-computed features to enhance the features for a target frame t. The fusion module thus comes somewhat late in the network.

Our network uses multiple input streams that eventually merge into a single output stream, as shown in figure 2. For computing the feature pyramid for a frame at time t, we will use n preceding frames and n future frames. All the $2n + 1$ frames are passed through the VGG-16 network, and we keep the outputs of blocks B3, B4 and B5 for each frame. In RetinaNet, these outputs are used to create the feature pyramid. We then use our fusion module to merge the corresponding feature maps of each frame (block B3 outputs together, block B4 outputs together, etc.) in order to enhance them, before

building the feature pyramid. This allows us to have higher quality features and better localize objects. We then use the enhanced maps as in the original RetinaNet to build the feature pyramid.

During the training process, we have to use multiple frames for one ground-truth example, thus slowing down the training process. However, for inference on video, the features computed for each frame are used multiple times making the processing time almost identical to the single frame baseline.

3.3 Fusion Module

In order to combine equivalent feature maps of consecutive frames, we designed a lightweight and trainable feature fusion module (see figure 3). The inspiration for this module is the various possible way a human would do the task. Let us say you wanted to combine feature map channels of multiple consecutive frames. Maybe you would look for the strongest responses and only keep those, making the merge operation an element-wise maximum. Maybe you would want to average the responses over all the frames. This 'merge operation' might not be the same for all channels. The idea is to have a network learn the best way to merge feature maps for each channel separately, with 1×1 convolutions over the channels.

In our fusion module, we use 1×1 convolutions in order to reduce the dimension of tensors. In the Inception module [24] of the GoogLeNet, the 1×1 convolution is actually used as a way to reduce the dimensions which inspired our work. The inception module allowed them to build a deeper and wider network while staying computationally efficient. In contrast, in our work, we use 1×1 convolutions for learning to merge feature maps.

The module takes as input $2n + 1$ feature maps of dimension $w * h * c$ (for width, height and channels respectively), and outputs a single enhanced feature maps of dimension $w * h * c$. The feature maps that we are combining come from corresponding pre-trained VGG-16 layers, so it is reasonable to think that corresponding channels are responses from corresponding 'filters'. The idea is to take all corresponding channels from the consecutive frames, and combine them to end up with only one channel, and thus re-build the wanted feature map, as shown in figure 3.

Formally, for $2n+1$ feature maps of dimension $w*h*c$, we extract each c channels one by one and concatenate them, ending up with c tensors of dimension $w*h*(2n+1)$. We then perform a 2D convolution with a 1×1 convolution kernel $(1*1*(2n+1))$ on the c tensors, getting c times $w*h*1$ as an output. The final step is to concatenate the tensors channel-wise to finally get the $w * h * c$ tensor that we need. The module is entirely learned, so we can interpret this module as the network learning the operation that best combines feature maps, for each channel specifically, without any prior knowledge or handcrafted features.

4 Experiments

4.1 Datasets

The training process of our method requires consecutive images from videos. We chose two datasets containing sequences of moving road users: UA-DETRAC [28] (fixed cam-

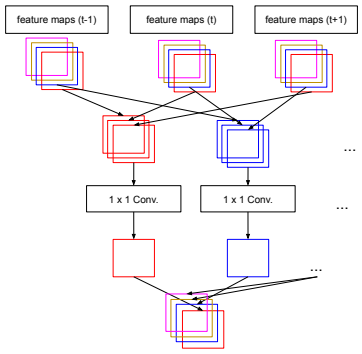
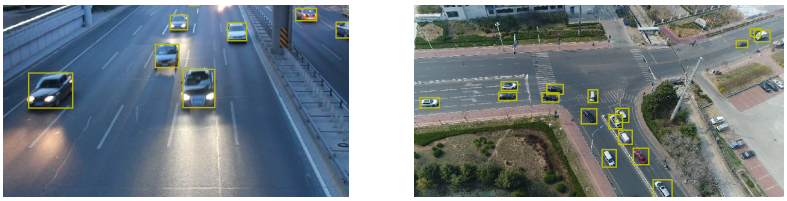


Fig. 3: Our fusion module consists of channel re-ordering, concatenation, 1×1 convolution, and a final concatenation (better seen in color).



(a) (b)

Fig. 4: (a) Example frame of UA-DETRAC and its ground-truth annotations. (b) Example frame of UAVDT and its ground-truth annotations.

era, 960x540, 70000 images, 4 possible labels, see figure 4a) and the Unmanned Aerial Vehicle Benchmark (UAVDT) [7] (mobile camera, 3 possible labels, 80000 images, high density of small objects, see figure 4b).

4.2 Implementations Details

We implemented the proposed model in Keras [3] using TensorFlow [1] as the backend. We used a standard RetinaNet as our baseline, without any bells and whistles or post-processing. We want to keep the models simple in order to properly show the contributions of our architecture and fusion module.

We built a feature pyramid with five different levels, called P3, P4, P5, P6, P7, with the outputs of block 3, 4, 5 of VGG-16. P3 to P5 are the pyramid levels corresponding to block 3 to 5. P6 and P7 are obtained via convolution and down-sampling of P5, and their size is reduced in half at each level: P6 is the half the size of P5, and P7 is the half the size of P6. This is standard for RetinaNet.

For UAVDT and UA-DETRAC, we adapted the scales used for the anchor boxes by reducing them, due to the high number of small objects in the tested datasets. Instead of using the classic $2^0, 2^{(1.0/3.0)}, 2^{(2.0/3.0)}$ scale ratios, we used $2^0, 1/(2^{(1.0/3.0)}), 1/(2^{(2.0/3.0)})$.

This modification did not affect the results on UA-DETRAC, but improved them on UAVDT, causing a bigger gap with the reported state-of-the-art results in the paper. Since we uses the same scales for our baseline, this has no effect on our conclusions. The focal loss parameter γ is 2 and we used an initial learning rate of 1e-5.

To train both the model and the baseline, we used the adam optimizer [11]. In order to fit the model into memory, we had to freeze the first four convolutional blocks of the VGG-16 model during training, and only retrained the other weights, with a batch size of one. For a fair comparison, we used the same training setting for our baseline. Despite this limitation, we still achieve state-of-the-art results when compared to single frame object detectors. Even though the first four convolutional blocks are frozen, they are still initialized with fine-tuned weights for each dataset. Note that the weights used to initialize the backbone are the same for the baseline and the model.

To select the hyperparameter n of our method (the number of frames used before and after), we used a validation set and tried a few values. $n = 2$ was the value that worked best for us, so that is the value we use for the final results. We show the results of different values of n in an ablation study in table 3.

4.3 Evaluation Metrics

For the two datasets, the test set is predetermined and cannot be used for training or to fix hyperparameters. We split the training data into training and validation by choosing a few whole sequences for validation, and the others for training. We did this to prevent overfitting on the validation data that would likely happen if we would split randomly between all frames. We trained the models until the validation loss started to increase, meaning the model was overfitting.

The metric used for evaluation is the mAP, meaning Mean Average Precision. The mAP is the mean AP for every class. The AP is the average precision considering the recall and precision curves, thus it is the area under the precision-recall curve. The minimum IOU between the ground-truth and the prediction bounding box, to consider a detection valid, is 0.7 for UA-DETRAC and UAVDT, as defined by the datasets protocols. The IOU, or the Jaccard index, is the intersection area between two rectangles divided by the union area between them.

4.4 Results

Results on UA-DETRAC Results on the UA-DETRAC dataset are reported in table 1. We drew the ROC curves for our model, the baseline and few other state-of-the-art models in figure 5a. Our detector outperforms all classic state-of-the-art models evaluated on UA-DETRAC as well as the baseline by a significant margin.

Something interesting to notice is that our model outperforms R-FCN for the categories labeled “hard” and “cloudy”, confirming our hypothesis that the features are indeed enhanced for hard cases like occlusion and blur (from motion or from clouds). As a result it raised the mAP for “overall” above R-FCN’s “overall”. We have to keep in mind that most VGG-16 layers are frozen during training, and that the final score would probably be much higher if this was not case. Nonetheless, our model convincingly surpasses the baseline in all categories, showing that features are enhanced not only for

Table 1: mAP reported on the UA-DETRAC test set compared to our baseline as well as classic state-of-the-art detectors. Results for "Ours" and "RN-VGG16" are generated using the evaluation server on the UA-DETRAC website, 3D-DETNet [13] is reported as in their paper, and others are as reported in the results section of the UA-DETRAC website. **Boldface**: best result, *Italic*: baseline.

Model	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
RN-VID (Ours)	**70.57**%	87.50%	75.53%	**58.04**%	**80.69**%	69.56%	56.15%	83.60%
R-FCN [4]	69.87%	**93.32**%	**75.67**%	54.31%	74.38%	**75.09**%	**56.21**%	**84.08**%
RN-VGG16	69.14%	86.82%	73.70%	56.74%	79.88%	66.57%	55.21%	82.09%
EB [26]	67.96%	89.65%	73.12%	53.64%	72.42%	73.93%	53.40%	83.73%
Faster R-CNN [23]	58.45%	82.75%	63.05%	44.25%	66.29%	69.85%	45.16%	62.34%
YOLOv2 [21]	57.72%	83.28%	62.25%	42.44%	57.97%	64.53%	47.84%	69.75%
RN-D [19]	54.69%	80.98%	59.13%	39.23%	59.88%	54.62%	41.11%	77.53%
3D-DETnet [13]	53.30%	66.66%	59.26%	43.22%	63.30%	52.90%	44.27%	71.26%

hard cases, but at all times. We outperform other video object detection for which we found results on UA-DETRAC, that is, 3D-DETNet [13] and RN-D-from-scratch [19]. The other video object detectors mentioned in the related works section did not produce results on this dataset.

Results on UAVDT Results on the UAVDT dataset are reported in table 2. We drew the ROC curves for our model, the baseline and few other state-of-the-art models in figure 5b. Our detector outperforms all classic state-of-the-art models evaluated on UAVDT as well as the baseline by a significant margin. The mAP scores on this dataset are quite low compared to UA-DETRAC due to its very challenging lighting conditions, weather conditions and smaller vehicles. We show that by adapting the scales used for the anchor boxes on each dataset, we can greatly improve results. Also, our model shows results on UAVDT that are consistent with UA-DETRAC's results, having an improvement of $\sim$1.2 mAP points against the $\sim$1.4 on UA-DETRAC.

5 Discussion

To explain the gains we get from our model, we now discuss a few reasons why aggregating features from adjacent frames is beneficial.

5.1 Analysis

Small objects: The smaller the object, the harder it will be to detect and classify, as a general rule. There is a large number of small objects in the evaluated datasets, as

Table 2: mAP reported on the UAVDT test set compared to our baseline as well as classic state-of-the-art detectors. Results for "Ours" and "RN-VGG16" are generated using the official Matlab toolbox provided by the authors, others are reported as in their paper. **Boldface**: best result, *Italic*: baseline.

Model	Overall
RN-VID (Ours)	**39.43**%
RN-VGG16	38.26%
R-FCN [4]	34.35%
SSD [18]	33.62%
Faster-RCNN [23]	22.32%
RON [12]	21.59%

there is in traffic surveillance scenes in general. Having multiple frames allows RN-VID to see the object from slightly different angles and lighting conditions, and a trained network can combine these frames to obtain richer features.

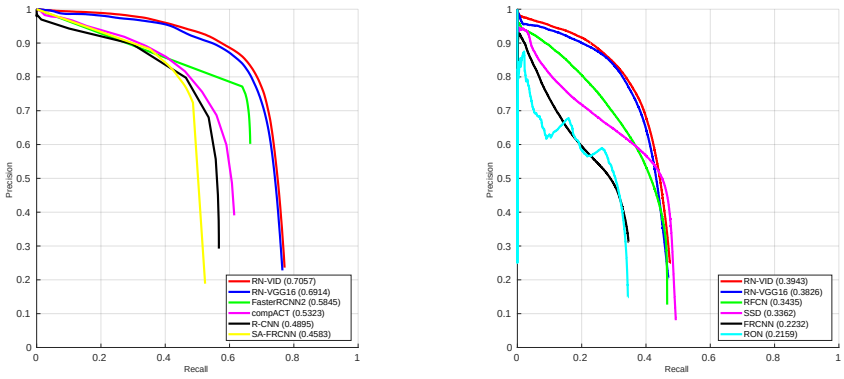
Blur: Blur is omnipresent in traffic surveillance datasets due to road users constant motion (motion blur), and weather/lighting conditions. A blurred object can be harder to classify and detect. Since its appearance is changed, the network could recognize it as none of the predetermined labels, and not considering it as a relevant object. Having multiple slightly different instances of theses objects allows the network to refine the features and output finer features to the classification sub-network. A convincing example of our model performing better in blurry conditions is the "Cloudy" category in which it got the best result.

Occlusion: Occlusion from other road users or from various road structures is very frequent in traffic surveillance scenes. Having access to adjacent frames gives our model a strong advantage against temporary occlusions, allowing it to select features from less occluded previous or future frames, making the detections more temporally stable. Figure 1 shows a qualitative example of our model performing better than the baseline in a case of occlusion.

5.2 Ablation Study

To properly assess the contribution of each part of our model, we performed an ablation study. We tried to isolate, as best as we could, our two contributions and looked at the impact of each of them. We justify the choice of using five consecutive frames with an experiment in which we varied this parameter on the UAVDT dataset. We tried several combinations and reported results in table 3. We can see than using five frames is better than using three, and that using three is better than using only one. We did not test with seven frames due to memory issues.

To remove the contribution of the fusion module, we trained a model where instead of merging the feature maps, we would simply concatenate them and continue to build



(a) (b)

Fig. 5: Precision-Recall curves on UA-DETRAC [28] (a) and UAVDT [7] (b) for RN-VID (Ours), RN-VGG16 (Baseline) and a few other state-of-the-art methods.

Table 3: mAP reported on the UAVDT test set of different variations of our model to conduct an ablation study. Results are generated using the official Matlab toolbox provided by the authors. Number of frames is the number of frames used for each detection.

Model	num. frames	Overall
RN-VID (Ours)	5	**39.43**%
RN-VID	3	39.05%
RN-VGG16 (Baseline)	1	38.26%
RN-VID-NO-FUSION	5	26.95%

the feature pyramid as usual, by adjusting the kernel size of the convolutions to adapt to the new input size. Doing this actually degrades the performance a lot as shown by the RN-VID-NO-FUSION model in table 3. This is easily understandable by the fact that combining feature maps like this is noisy, and we might need way more data and parameters in order to make this work. We can conclude from this that the fusion module is an essential part of our model.

5.3 Limitations of our Model

A limitation of our model is for border situations, the first and last frames of a sequence where we cannot use our architecture to its full potential. However, this is not a problem since we can do a padding by repeating the first and last frame the number of time needed to without a real loss of performance. Also, it takes more memory to train the

model then its single frame counterpart, due to the fact that we need multiple frames to train one single ground-truth example.

6 Conclusion

A novel approach for video object detection named RN-VID was introduced, composed of an object detection architecture and a fusion module for merging feature maps. This model was trained and evaluated on two different traffic surveillance dataset, and a general video object detection dataset, and compared with a baseline RetinaNet model and several classic state-of-the-art object detectors. We show that by using adjacent frames, we can increase mAP by a significant margin by addressing challenges in the traffic surveillance domain like occlusion, motion and general blur, small objects and difficult weather conditions.

Acknowledgments. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RDCPJ 508883 - 17], and the support of Genetec. The authors would like to thank Paule Brodeur for insightful discussions.

References

1. Abadi, M., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `http://tensorflow.org/`, software available from tensorflow.org
2. Bertasius, G., Torresani, L., Shi, J.: Object detection in video with spatiotemporal sampling networks. In: The European Conference on Computer Vision (ECCV) (2018)
3. Chollet, F., et al.: Keras. `https://keras.io` (2015)
4. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems 29, pp. 379–387. Curran Associates, Inc. (2016)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
6. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: The IEEE International Conference on Computer Vision (ICCV) (2015)
7. Du, D., Qi, Y., Yu, H., Yang, Y., Duan, K., Li, G., Zhang, W., Huang, Q., Tian, Q.: The unmanned aerial vehicle benchmark: Object detection and tracking. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 370–386 (2018)
8. Girshick, R.: Fast r-cnn. In: The IEEE International Conference on Computer Vision (ICCV) (2015)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
10. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

12. Kong, T., Sun, F., Yao, A., Liu, H., Lu, M., Chen, Y.: Ron: Reverse connection with objectness prior networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition. vol. 1, p. 2 (2017)
13. Li, S., Chen, F.: 3d-detnet: a single stage video-based vehicle detector. In: Third International Workshop on Pattern Recognition. vol. 10828, p. 108280A. International Society for Optics and Photonics (2018)
14. Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
15. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. IEEE transactions on pattern analysis and machine intelligence (2018)
16. Liu, M., Zhu, M.: Mobile video object detection with temporally-aware feature maps. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
17. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. In: 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). pp. 730–734 (2015). https://doi.org/10.1109/ACPR.2015.7486599
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
19. Perreault, H., Bilodeau, G.A., Saunier, N., Gravel, P.: Road user detection in videos. arXiv preprint arXiv:1903.12049 (2019)
20. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
21. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
22. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
23. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
24. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
25. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2), 154–171 (2013)
26. Wang, L., Lu, Y., Wang, H., Zheng, Y., Ye, H., Xue, X.: Evolving boxes for fast vehicle detection. In: 2017 IEEE International Conference on Multimedia and Expo (ICME). pp. 1135–1140. IEEE (2017)
27. Wang, S., Zhou, Y., Yan, J., Deng, Z.: Fully motion-aware network for video object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 542–557 (2018)
28. Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.C., Qi, H., Lim, J., Yang, M.H., Lyu, S.: UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking. arXiv CoRR **abs/1511.04136** (2015)
29. Xiao, F., Jae Lee, Y.: Video object detection with an aligned spatial-temporal memory. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 485–501 (2018)
30. Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 408–417 (2017)