# Supplementary Material for Recurrent Multi-frame Single Shot Detector for Video Object Detection

Alexander Broad[1]
alex.broad@u.northwestern.edu
Michael Jones[2]
mjones@merl.com
Teng-Yok Lee[2]
tlee@merl.com

[1] Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL USA

[2] Mitsubishi Electric Research Labs
201 Broadway, 8th floor
Cambridge, MA USA

## 1 RMf-SSD Network Architecture (SqueezeNet+)

| Layer Name | Activation Dims. | Filter Size/Stride |
|---|---|---|
| Input | 1242x375x3 | |
| conv1 | 620x187x64 | 3x3/2 |
| maxpool1 | 309x93x64 | 3x3/2 |
| fire2 | 309x93x128 | |
| fire3 | 309x93x128 | |
| maxpool3 | 154x46x128 | 3x3/2 |
| fire4 | 154x46x256 | |
| fire5 | 154x46x256 | |
| maxpool5 | 76x22x256 | 3x3/2 |
| fire6 | 76x22x384 | |
| fire7 | 76x22x384 | |
| fire8 | 76x22x512 | |
| ConvGRU | 76x22x512 | 3x3/1 |
| fire9 | 76x22x768 | |
| fire10 | 76x22x768 | |
| ConvDet | 76x22x72 | 3x3/1 |

Table 1: Details of Recurrent Multi-frame Single Shot Detector Network Architecture.

A more detailed description of the Recurrent Mf-SSD network architecture can be found in Table 1. Our network architecture is based on SqueezeDet+ [14], which in turn, is based on the SqueezeNet+ [9] architecture. SqueezeNet(+) was designed specifically to reduce the number of parameters used to define a model in order to improve efficiency both during training and at test time. The key architectural design strategies described in the SqueezeNet paper were (1) replacing $3x3$ filters with $1x1$ filters, (2) decreasing the number of input

channels to 3$x$3 filters, and (3) downsampling late in the network to allow convolutional layers to have large activation maps. To achieve this goal, the authors describe a *fire module* which is used to help solve design strategies (1) and (2). Fire modules are comprised of a squeeze layer and an expand layer. The squeeze layer is a convolutional layer defined only by 1$x$1 filters, which is used to reduce the number of total parameters in the network. The expand layer is also a convolutional layer but is defined by a set of 1$x$1 filters and 3$x$3 filters. The total number of filters in the squeeze layer is set to be less than the number of filters in the expand layer to reduce the number of input channels to the larger feature maps. Finally, the main modification we make to the baseline network architecture is the addition of a convolutional GRU (ConvGRU) layer, which is used to learn spatio-temporal correlations between features extracted from sequential video frames.

# 2 RMf-SSD Training Details (SqueezeNet+)

To train the model, we initialize the base feature-extractor network with weights from a pre-trained single-frame SqueezeDet+ network. All other layers are initialized using the Xavier initialization [6]. We use the same training and validation splits to train the base SqueezeDet+ model and the Recurrent Mf-SSD model to ensure no data leakage. When training the base SqueezeDet+ we initialize the model with ImageNet weights and freeze the first convolutional layer while all other layers remain trainable. When training the Recurrent Mf-SSD model the first convolutional layer and the first three fire modules are frozen, while the remaining five fire modules in the feature extractor are left trainable. The convolutional GRU employs a $3 \times 3$ filter and outputs the same number of channels as the prior feature map (512). It is also important to note that during training, sequences of $N$ frames are fed into the network at the same time, where $N-1$ is the number of prior frames used in the Mf-SSD. However, during test time, the images can be fed in one at a time in sequence. The weights of the feature extractor that operates on each frame are shared which ensures that the same features are extracted from an image regardless of its position within the input sequence. Finally, to reduce the effect of network and training hyperparameters we use the same exact parameters chosen by Wu et al. [14]. When training the RMf-SSD we reduce the batch size from 20 to 10 to ensure the models fit in memory. Each model is trained for 90,000 iterations using a momentum optimizer (momentum = 0.9) and an initial learning rate of $10^{-2}$ which exponentially decays after every 10,000 training iterations.

## 2.1 RMf-SSD Training (Alternative Feature Extractors)

When experimenting with alternative feature extractors (SqueezeNet, VGG16, and ResNet-50), we used nearly the same training procedure as with SqueezeNet+. This includes using a batch size of 20 for single-frame models and 10 for multi-frame models. The recurrent convolutional layer is inserted into the same location in each model (directly after the feature extractor) and uses the same definition (number of layers and channels). For these models, we followed the fine-tuning strategy for squeezeNet+: We trained the single-frame versions by re-training layers after the first max-pooling operator, while the multi-frame models were fine-tuned by re-training the layers after the second max-pooling operator. All models were trained on a 12GB TitanX GPU.

One issue with training SSDs based on architectures with more parameters (like VGG-16 and ResNet-50) is the memory requirement. In particular, without making any modifications,

we found it impossible to train SSDs based on these architectures with batch sizes $\geq 10$ on a TitanX. To reduce memory constraints, we down-sampled both dimensions of the input image by two. This explains why the models based on VGG-16 and ResNet-50 do not perform as well as the model based on SqueezeNet+ in the main paper. Another way to reduce the memory consumption is to reduce the batch size, however, we found that reducing the batch size too much can have a very large impact on the final mAP of the models. The batch numbers in our experiments were empirically selected to trade-off between accuracies and memory consumption.

Of note, even with down-sized images, we still failed to train the multi-frame model with ResNet-50. One reason is that the last convolutional layer has a large number of channels (2048), which can leads to very large RNN layers. Our workaround was to skip the layers with 2048 channels of ResNet-50, which are all layers with the prefix conv5. This choice also likely explains why VGG-16 outperformed ResNet-50 in our experiments.

# 3 Alternative Multi-Modal Fusion Techniques

We also explore alternative methods for directly incorporating hand-computed features, such as optical flow and depth, into the video object detection process. The multi-frame fusion methods described in the main paper all require the object detection framework to learn video features (such as motion and temporal correspondence) from unlabeled training data. However, we can also consider models that directly incorporate well known video features, such as optical flow and depth. By integrating optical flow as an input to the network one can hope to stimulate motion sensitive cues and other temporal features. By integrating depth as an input one can hope to incorporate three dimensional spatial features. Here we evaluate the effect of multi-modal fusion techniques using a multi-stream approach.

| | Car | | | Pedestrian | | | Cyclist | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | E | M | H | E | M | H | E | M | H | mAP |
| SqueezeDet | 0.933 | 0.885 | 0.798 | 0.858 | 0.775 | 0.741 | 0.872 | 0.845 | 0.789 | 0.833 |
| FW Add | 0.924 | 0.882 | 0.796 | 0.859 | 0.778 | 0.750 | 0.884 | 0.858 | 0.793 | 0.836 |
| FW Max | 0.934 | 0.885 | 0.799 | 0.855 | 0.776 | 0.744 | 0.873 | 0.854 | 0.820 | 0.838 |
| FW Concat | 0.929 | 0.883 | 0.797 | 0.866 | 0.782 | 0.754 | 0.888 | 0.878 | 0.838 | 0.846 |
| 2t RGB + Flow | 0.929 | 0.890 | 0.803 | 0.875 | 0.789 | 0.749 | 0.894 | 0.848 | 0.805 | 0.842 |
| " " + Depth | 0.952 | 0.894 | 0.873 | 0.873 | 0.787 | 0.762 | 0.892 | 0.869 | 0.804 | 0.856 |

Table 2: KITTI Detection results using optical flow and depth as inputs.

To incorporate optical flow into our video object detection framework, we test two different techniques. First we feed flow-warped sequential images directly into a multi-stream network. Each of the previous three frames is warped so that it is in correspondence with the current frame using pre-computed flow fields. This simple idea does not require altering the previously described training process. The results of these experiments can be found in Table 2 (Lines 2-4). Similar to the results of the experiments on the unaltered input images, we find that simply concatenating the extracted feature maps provides the largest improvement. However, it is of note that the element-wise operations no longer have a deleterious effect. This is evidence that the degradation in accuracy that we observed in the prior experiments was due to a lack of correspondence between frames. By solving this issue, we

observe small improvements in the total accuracy of each model that uses element wise operations to merge the information. We do observe that the concatenation approach performs slightly worse than in the original unwarped images and we attribute this small difference to the effects of the warping process which can often produce unnatural features in the images warped over the longest time horizon.

The second way in which we incorporate optical flow into the network is by developing a separate parallel stream which learns features directly on a three color channel representation of a computed optical flow image [1]. We first separately train baseline SqueezeDet+ models on both RGB and pre-computed optical flow data (using the algorithm described in [2]). We then create a three-stream Mf-SSD with two standard RGB streams (which take the current and prior time-step images as input), and one optical flow stream (which takes the pre-computed optical flow image between the two RGB input images). The results from this approach can be found in Table 2 (Line 5). Here we again see a modest increase in overall mAP in comparison to the baseline single-frame approach.

A third comparison we perform is incorporating both optical flow and estimated depth as additional input modalities. Motion cues are not the only additional information available in video data compared to static images. Depth is a modality that has previously been explored in the context of object detection [4, 8]. Here, we again train an RGB stream, an optical flow stream and we add a pre-trained depth stream. This network is trained on pre-computed estimated depth information using the single-frame approach described by Godard et al. [7]. The results of the combined RGB, optical flow and depth network can also be found in Table 2 (Line 6). This *kitchen sink* approach does well, improving upon the baseline model by 2.3%. However, we note that it still does not do quite as well as the best performing Mf-SSD and is significantly more complicated in terms of network design and computational requirements. For example, this approach requires a multi-stage training process and pre-computed optical flow and depth images, whereas the described Mf-SSDs can be trained end-to-end and only require the original RGB images.

# 4 Additional Analyses of the KITTI Detection Results

Here we further analyze the improvement demonstrated by the RMf-SSD models over the single-frame baselines using a variety of feature extractors. As mentioned in the paper, each multi-frame models improve upon its respective single-frame baseline along all breakdowns.

| | SqueezeNet | SqueezeNet+ | VGG-16 | ResNet-50 | Avg. |
|---|---|---|---|---|---|
| Easy | 4.1 | 1.3 | 2.1 | 1.0 | 2.13 |
| Moderate | 4.3 | 1.6 | 1.1 | 1.4 | 2.10 |
| Hard | 4.7 | 5.1 | 1.9 | 1.9 | 3.40 |
| Avg. | 4.37 | 2.67 | 1.87 | 1.33 | |

Table 3: Raw % improvements in mAP demonstrated by multi-frame SSD over single-frame SSD baseline. The results are broken down by baseline architecture and detection difficulty (as defined in the KITTI dataset).

Table 3 demonstrates a consistent improvement in each level of detection difficulty (as defined in the KITTI dataset). These results hold for four different feature extractors.

### 4.1 Breakdown by Class Category

| | SqueezeDet+ | RMf-SSD | Raw % Impr. |
|---|---|---|---|
| Car | 0.872 | 0.902 | 3.0 |
| Pedestrian | 0.791 | 0.802 | 1.1 |
| Cyclist | 0.835 | 0.874 | 3.9 |

Table 4: KITTI Detection results broken down by object category. Figures presented are mean average precision (mAP).

Through further analysis of our results on the KITTI dataset, we also find a distinction in the amount of improvement we see between the different classes. In particular, we see the greatest improvement in the *Cyclist* class and the smallest improvement in the *Pedestrian* class. One possible reason we see this difference is that the *Pedestrian* class may simply be smaller in scale on average then the other classes, and smaller objects are a known challenge in CNN based object detectors [11]. If this is the underlying reason it is possible to improve our results by incorporating deconvolutions [6, 12] or dilated convolutions [15] to increase feature map resolution. However, it is also important to note that, if we look at the raw mAP values, both the baseline method and the RMf-SSD are simply better able to detect objects of the Car and Cyclist classes than the Pedestrian class. Therefore, if the challenge is particularly related to the *Pedestrian* class, there are numerous complementary methods we can incorporate into our framework (e.g. improving hard example mining [13] or updating the loss function to incorporate more negative information [10]). This breakdown is the motivating reason for evaluating our method on the Caltech Pedestrians dataset in the paper.

## 5 Training RMf-SSD for Caltech Pedestrians Dataset

The Caltech Pedestrian dataset consists of continuous videos and are broken down into pre-defined training (video sets 00 - 05) and testing sets (06 - 10). To work with this dataset, we make small modifications to our baseline Recurrent Mf-SSD implementation. Specifically, we change the input size of the network to retain the original resolution of the Caltech dataset (480x640). We also replace the default anchor boxes defined for KITTI with nine new anchor boxes in which the heights and aspect ratios are defined by running k-means on the labeled data. We train on the 'Person' class in the Caltech dataset and do not make use of other complementary pedestrian datasets (e.g. [4]). All other hyper-parameters are left unchanged.

## References

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, Mar 2011.

[2] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10, May 2001.

[3] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[4] Xianzhi Du, Mostafa El-Khamy, Jungwon Lee, and Larry Davis. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 953–961. IEEE, 2017.

[5] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. DSSD : Deconvolutional Single Shot Detector. *arXiv:1701.06659*, 2017.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[7] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[8] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[9] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv:1602.07360*, 2016.

[10] Davis E King. Max-margin object detection. *arXiv:1502.00046*, 2015.

[11] Jianan Li, Xiaodan Liang, ShengMei Shen, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Scale-aware fast r-cnn for pedestrian detection. *arXiv:1510.08160*, 2015.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[13] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[14] Bichen Wu, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. In *CVPR Workshops*, 2017.

[15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv:1511.07122*, 2015.