# Efficient Multi-Tenant Inference on Video using Microclassifiers

Giulio Zhou, Thomas Kim, Christopher Canel, Conglong Li, Hyeontaek Lim,
David G. Andersen, Michael Kaminsky†, Subramanya R. Dulloor†
*Carnegie Mellon University;* †*Intel Labs*

## 1 INTRODUCTION

This paper addresses a growing challenge in processing video: The scaling challenge presented by the combination of an increasing number of video sources (cameras) and an increasing number of heavy-weight DNN-based applications (which we term "queries") to be run on each source. As a running example, we draw from an environmental and traffic monitoring deployment at CMU, one feed from which is depicted at right. This feed supports applications such as car and pedestrian counting, open parking spot detection, train detection (in support of an environmental monitoring research project attempting to determine locomotive emissions), and observing if building lights are left on. These cameras are deployed using a mix of the high-speed campus network, and a lower-speed/higher-cost cable modem deployment on power poles in the area.

Cost constraints motivate us to be parsimonious with bandwidth on the wide-area deployment (and planned future wirelessly-connected nodes), and, in general, of our CPU/GPU processing budgets. The high cost of truck rolls to install and upgrade nodes motivates us to put multiple state-of-the-art 4K vision cameras on the nodes to flexibly support future applications, but we lack the bandwidth to backhaul the full feeds. At the same time, both current and future applications may wish to run one or more state-of-the-art DNNs to perform image classification [6, 17], object detection [7, 13, 14], and video understanding [3, 19].

In this paper, we assume that cameras are fixed (e.g., traffic monitoring). Most applications are interested in possibly-overlapping *subsequences* of frames (e.g., frames containing cars, or trains, or with people moving), and can express a notion of that importance using a *query*. Frames that match a query are sent back to the datacenter for further processing. Each application defines its own queries and submits them to the edge node. Our system is responsible for efficiently executing a multitude of queries at the edge node with low false positives (to avoid wasting resources) and low false negatives (to preserve application fidelity).

While an abstract query could be a black-box DNN that takes a frame as input and outputs a binary forward/discard decision, such an approach scales poorly as the number of queries executed on each stream grows. Instead, we develop an idea called *microclassifiers*, which are small classifiers taking as input a subset of the activations of a known, standard convolutional neural network (such as MobileNet [5]). Each query is represented using a unique microclassifier, which specifies both its own internal DNN structure, as well as identifying which (small) subset of the reference CNN activations it accepts as input. Microclassifiers enable an edge node to serve tens of queries or more with high accuracy by amortizing the cost of the CNN activations, adding only a small cost per query.

**Dataset.** We evaluate the microclassifier approach on a novel dataset containing 83.5 hours of footage from a camera overlooking train tracks, sampled at 1fps, for a total of 290,758 frames, as shown in Figure 1. We use the first 100,000 frames as a training set and the remainder as the test set. We created two sets of labeled data by annotating frames in which (a) a train appears in Region A (the *Train* dataset); and (b) a car appears in Region B (the *Car* dataset). There are 2,777 frames that contain images of 21 different trains, and 16,070 frames that contain images of 1296 different vehicles. Our goal is for this dataset to be a simple yet representative example of a typical traffic monitoring workload.

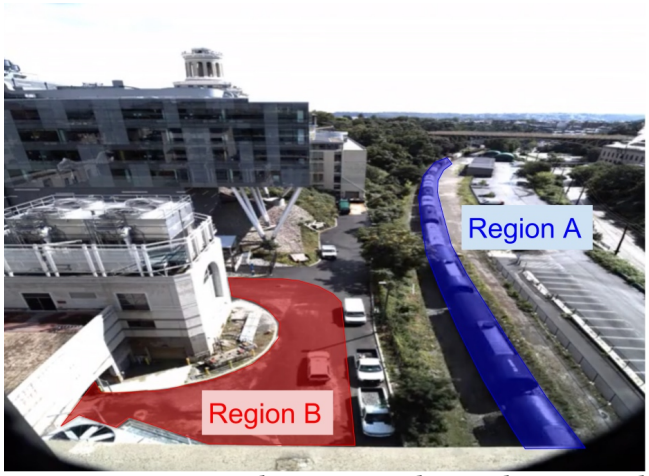


**Figure 1: Regions A and B, corresponding to the *Train* and *Car* datasets respectively.**

## 2 SCALABLE VIDEO QUERIES AT THE EDGE

**Transfer Learning Doesn't Work Well.** In our preliminary evaluations, traditional transfer learning—fine-tuning the last *n* layers of a CNN—yielded poor results [Table 2]. There are two contributing factors. First, the frames (and by extension, the features) from a fixed-view camera are generally quite similar, which makes it easy for a fine-tuned MobileNet to overfit (as evidenced by fast training convergence and poor test results). Second, because many events in this setting occur in a spatially constrained part of the frame, globally pooled features have weak discriminative ability. Cropping the image to cover only the spatially relevant portion is not scalable nor sufficiently general because cropping (a) cannot handle non-rectangular inputs, (b) distorts non-square inputs, and (c) requires the network to be run separately for each image crop.

**Use Shared CNN Feature Maps Instead.** CNNs trained on image classification tasks produce nonlinear hierarchical features that offer a trade-off between spatial localization and semantic information. Selective processing of these features has been used successfully in tasks such as object region proposals, segmentation, and tracking [2, 4, 10, 14], as well as video action classification [16]. Our approach uses a single pretrained CNN that runs on every frame in a multi-task fashion. We allow all of our query models

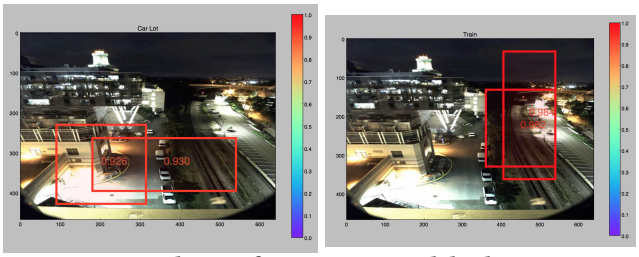


**Figure 2: *Car* and *Train* feature crops with highest ROC AUC scores overlaid on the original image.**

(described below) to share the entire CNN feature hierarchy. Doing so gives each model flexibility to choose its degree of spatial localization and enables a single resource-constrained edge node to run more application queries.

For our experiments, we use feature maps from MobileNet-224 [5] trained on ImageNet [15]. MobileNet layers are referred using the layer names of the Caffe port [11]. While MobileNet features were sufficient for the tasks that we investigated, we plan to consider training a CNN on multiple tasks to obtain feature maps that generalize better across many domains.

**Input Selection Beats Microclassifier Structure.** Our results show that the flexibility for a microclassifier to draw from arbitrary activations from the reference CNN is more important to their accuracy than the exact structure of the microclassifier. Table 3 shows that both SVMs, KNN, and simple multilayer perceptrons (MLP)s perform well when using a well-chosen feature map from one internal layer of MobileNets. Both the linear SVM and MLP are comparatively cheap to execute (about 10K multiply-adds dot product for the SVM, and the MLP requiring about 2M multiply/adds).

In contrast, the choice of reference network activations (feature sets) has a much larger effect. Table 1 shows the effect of using different layers and crops of layers from MobileNet as input to the microclassifier.

As shown above, basic transfer learning (which uses the last pooling layer as its feature set) performs particularly poorly. Choosing the best random crops of conv6/sep and conv5_6/sep (the last and second-to-last feature maps) yielded better results but was not significantly better than using a cropped image's pool6 features. We obtained much better FP and FN rates using the best crops of an earlier layer of the network (conv4_2/sep) [Figure 2].

As expected, the best performing crops came from regions that overlapped with the event regions. These results are somewhat intuitive for the spatially-segregated tasks we evaluated, but the order of magnitude of the improvement was large, and, we believe, particularly applicable for the monitoring applications we consider.

**Automated Feature Search.** Given that a key feature of microclassifiers is their flexibility to draw from internal activations in a reference CNN, how can we "train" a microclassifier such that it does so effectively?

As a proof-of-concept, we implemented a search procedure that randomly samples and evaluates crops of a given feature map as a first step towards a more general automated solution. (Random search is competitive baseline for hyperparameter optimization [1].) Our procedure first samples (in normalized coordinates between 0 and 1) the center of the crop box from a truncated Gaussian with mean 0.5, standard deviation 0.2, and range [0.1, 0.9]. The box dimensions are then (independently) sampled from a second truncated Gaussian with mean 0, standard deviation 0.2, and range [0.1, 0.5] (out of range box coordinates are clipped to the edges of the image). This results in crops of intermediate size that tend towards the center.

| Task | Features | Accuracy (%) | FP (%) | FN (%) |
|---|---|---|---|---|
| car | pool6 | 66.41 | 34.36 | 21.42 |
| car | pool6 (cropped image) | 74.42 | 26.64 | 8.72 |
| car | conv4_2/sep (2:6, 7:12) | 85.37 | 14.73 | 13.04 |
| train | pool6 | 98.04 | 1.26 | 65.27 |
| train | pool6 (cropped image) | 97.75 | 2.06 | 19.57 |
| train | conv4_2/sep (9:10, 4:12) | 98.84 | 1.07 | 9.47 |

**Table 1: A comparison of 2-layer MLP microclassifier accuracy across different feature sets. Feature map crop notation is given by (x_start:x_end, y_start:y_end), end coordinates are inclusive.**

| Setup | Iterations | Accuracy (%) | FP (%) | FN (%) |
|---|---|---|---|---|
| From scratch | 100 | 51.56 | 51.05 | 44.98 |
| Fine tune last 2 layers | 100 | 74.23 | 25.52 | 48.89 |
| Fine tune last layer | 100 | 84.37 | 14.53 | 57.25 |

**Table 2: Transfer learning results using MobileNet-224 on the *Train* dataset. On all three setups, the training loss had already converged after 100 training iterations.**

| Classifier | Accuracy (%) | FP (%) | FN (%) |
|---|---|---|---|
| Linear SVM (C=1e-3) | 99.76 | 0.009 | 21.49 |
| Linear SVM (C=1) | 97.47 | 2.448 | 10.00 |
| Linear SVM (C=1 cost class balance=0.1:1) | 98.20 | 1.715 | 9.47 |
| KNN (k=10, >= 2 positive examples) | 99.50 | 0.104 | 36.38 |
| 2-layer MLP (200 hidden units, ReLU activation) | 99.75 | 0.141 | 10.48 |

**Table 3: Microclassifier architecture accuracies on the *Train* dataset using the optimal feature crop as input.**

While this basic search shows that the general technique is feasible, we plan several improvements upon this basic procedure: First, rather than flattening the input features and thus discarding its spatial relationships, can we pass them directly into CNN microclassifiers with learned attention mechanisms [18] that can use those relationships? In addition, because the events we are interested in generally consist of a short, contiguous collection of frames, we may also consider creating microclassifiers that operate across multiple frames, and capture interesting video segments rather than just individual frames.

Second, we plan to broaden the scope of the feature search to cover non-contiguous parts of the feature map and to pull from multiple layers simultaneously. We would also like to explore better feature selection strategies. Some mechanisms for accomplishing the former include introducing regularization that encourages block-sparsity [8, 12] or devising an evolutionary procedure that effectively searches the feature space. Another way of exploring the feature space could involve asking the user to specify the regions of interest in the image and then using the network's receptive field [9] to create a sampling distribution.

# REFERENCES


[1] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. 2016. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*. Springer, 850–865.

[3] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*.

[4] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 447–456.

[5] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017). http://arxiv.org/abs/1704.04861

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.

[8] Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning Sparse Neural Networks through $L\_0$ Regularization. *arXiv preprint arXiv:1712.01312* (2017).

[9] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. 2016. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 4898–4906.

[10] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. 2015. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*. 3074–3082.

[11] mobilenetcaffe 2017. MobileNet-Caffe. http://github.com/shicai/MobileNet-Caffe. (2017).

[12] Sharan Narang, Eric Undersander, and Gregory Diamos. 2017. Block-Sparse Recurrent Neural Networks. *arXiv preprint arXiv:1711.02782* (2017).

[13] Joseph Redmon and Ali Farhadi. 2016. YOLO9000: Better, Faster, Stronger. *CoRR* abs/1612.08242 (2016). http://arxiv.org/abs/1612.08242

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.

[16] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. 2015. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119* (2015).

[17] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[18] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. 2048–2057.

[19] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. [n. d.]. End-to-End Learning of Action Detection from Frame Glimpses in Videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016 (CVPR 2016)*.