Deep Learning Optimization
- Domain Adaptation
& Online Training

March 27, 2023

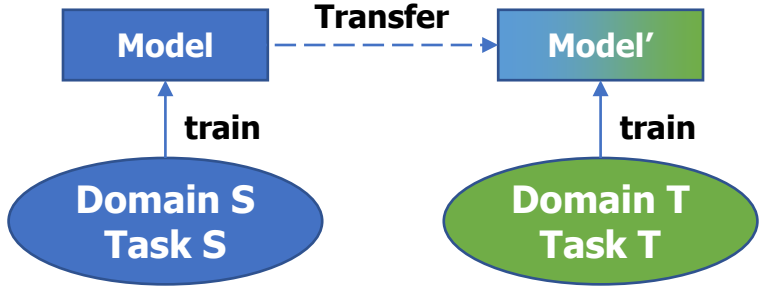
Eunhyeok Park

Agenda

- Transfer learning and domain adaptation
  - Network training using synthetic data
    - CNN for optical flow
      - Side story: evolution of optical flow network
      - Side story: application of flow network
    - Training object detector
  - Domain adaptation for low-cost video streaming

- Online training
  - Object tracking examples
  - Online network adaptation for low-video segmentation

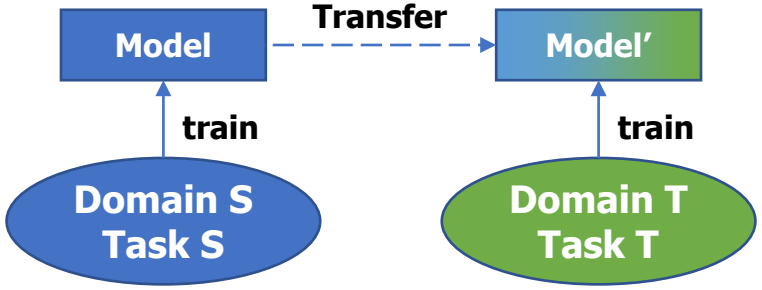
Transfer Learning



- Notation and definition
  - **Domain** $\mathbf{D} = \{\boldsymbol{\mathcal{X}}, \mathbf{P}(\boldsymbol{X})\}$,
    $where\ \mathcal{X}$ is a feature space and a marginal probability distribution $P(X)$ for $X \in \mathcal{X}$
    - Image, spectrogram, item history…
  - **Task** $\mathbf{T} = \{\boldsymbol{\mathcal{Y}}, \mathbf{P}(\boldsymbol{y}|\boldsymbol{x})\}$,
    $where\ \mathcal{Y}$ is a label space and a conditional probability distribution $P(Y|X)$ for $Y \in \mathcal{Y}$
    - Annotated label, i.e. class label, bounding box position…
- Transfer learning?:
  - Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, **transfer learning aims to help improve the learning of the target predictive function** $\boldsymbol{f_T}(\cdot)$ **in** $\boldsymbol{D_T}$ using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$

[A Survey on Transfer Learning, Pan]

Transfer Learning - 2



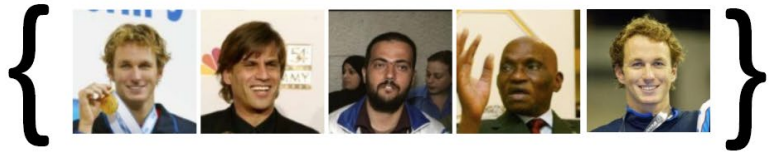
- $D_S \neq D_T, T_S = T_T$: **Domain adaptation**

0-9 digits classification
(e.g., 6, 2, 3, 8, 1, 0)

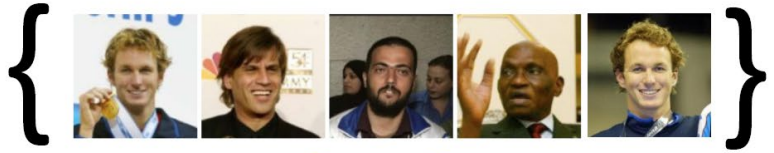
0-9 digits classification
(e.g., 0, 4, 8, 5, 3, 9)

$\mathcal{D}_S \neq \mathcal{D}_T$ $\mathcal{T}_S = \mathcal{T}_T$

- $D_S = D_T, T_S \neq T_t$: **Inductive transfer learning**



Age (e.g., 31, 49, 34, 50, 31)

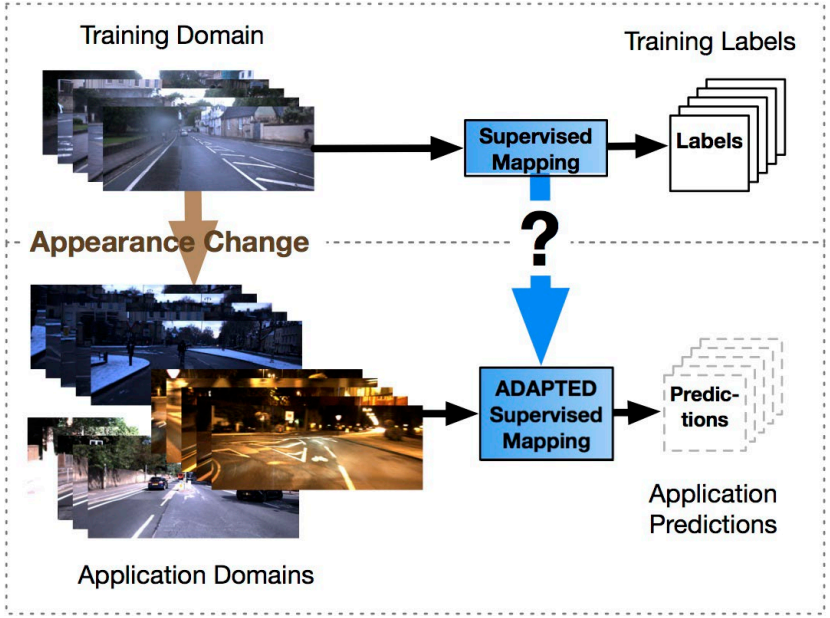


Person recognition
(e.g., John, Aaron, Adam, Will, John)

$\mathcal{D}_S = \mathcal{D}_T$ $\mathcal{T}_S \neq \mathcal{T}_T$

Domain Adaptation

- Domain adaptation can be seen as a subcategory of transfer learning. However, it is important as a standalone topic because of its wide and practical usage
  - The source and target domains all have the same feature space, but their distribution could be different
  - Domain shift, or distribution shift is a change of data distribution between source and target dataset
    - Sensor parameter difference
      - Lends distortion
      - Mic frequency response
    - Environment difference
      - Weather
      - External noise
    - Real vs synthetic data

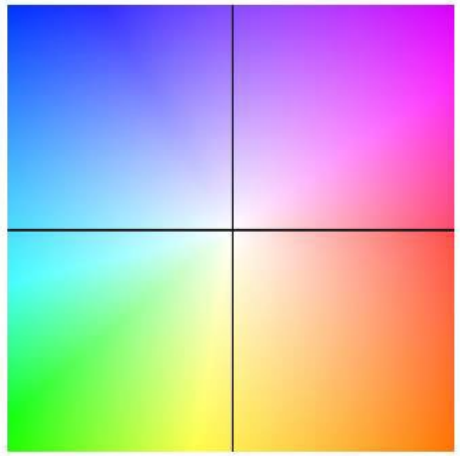


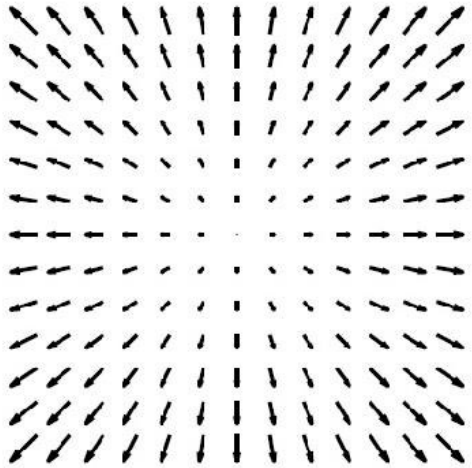
Training with Synthetic Data

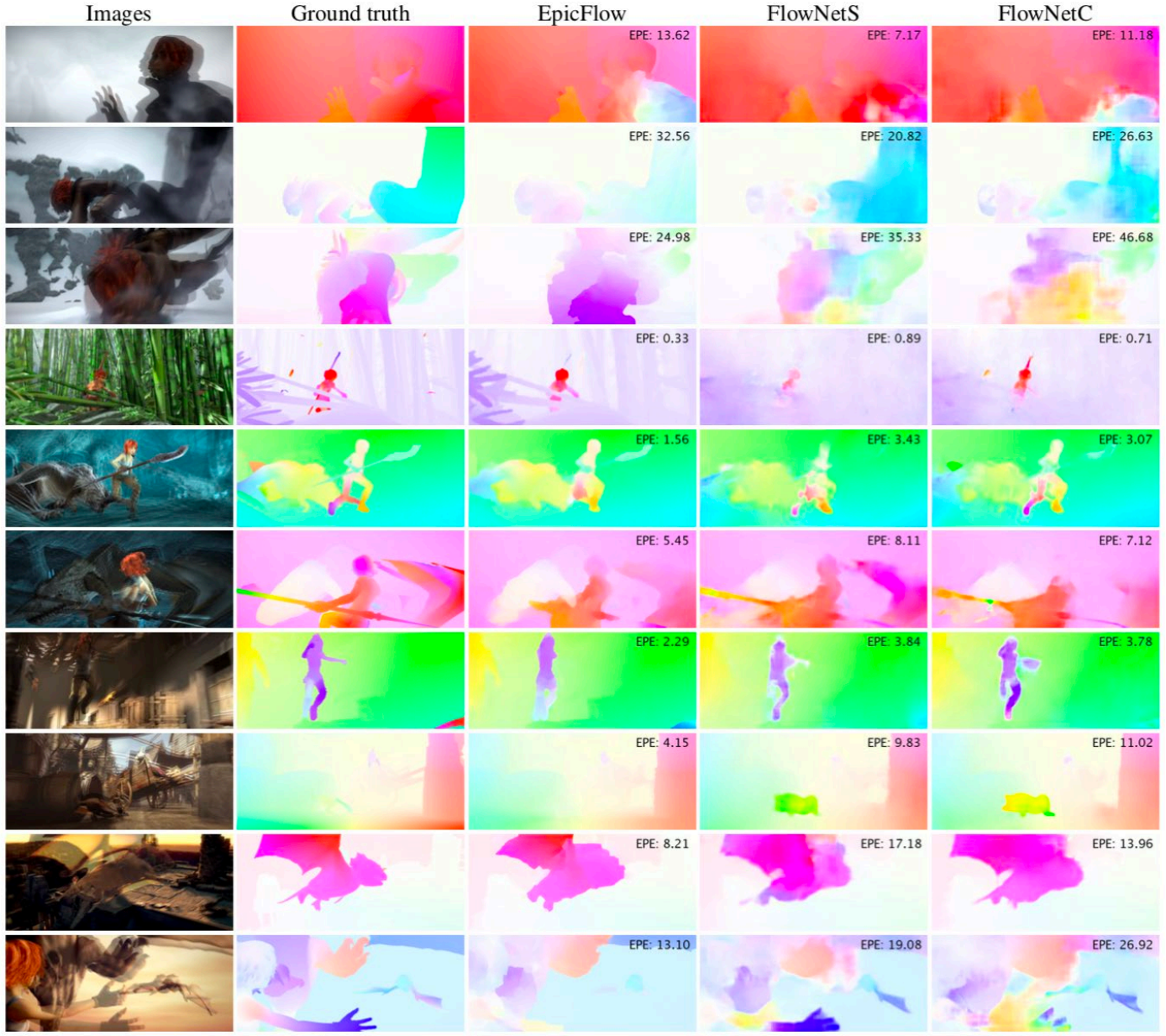
- Training network with real data and the human-annotated label is expensive
- As an alternative, synthetic data-based training has been proposed
  - Pros
    - Relatively easy to increase training dataset size
    - Fast & accurate annotation based on generation information
      - Segmentation, bounding box, depth, instance id, …
  - Cons
    - Effort to create a synthesize environment
    - Distribution difference between real and synthetic data
      - Network may learn the unexpected feature in the synthetic data

Optical Flow

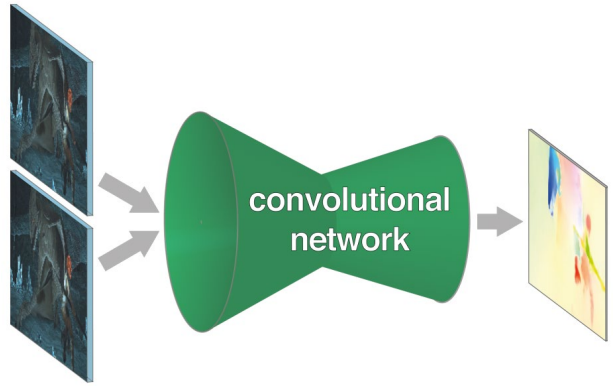
- Optical flow is a numerical estimation, expressed as vector map, of an object movement in a visual scene between the adjacent frames
  - Color: direction
  - Saturation: magnitude







FlowNet



- Conventional optical flow estimation is very expensive
  - What if we solve this problem using convolutional neural network?
    - We can exploit massive parallelism based on GPU/NPU

- Remained problem?
  - We need large amount of data to train the network
    - We can create the ground truth from realistic video can be possible but…
      - We need to spend large amount of computation time
      - Precise labeling is known to be extremely difficult
    - Use synthetic data!

	Frame pairs	Frames with ground truth	Ground truth density per frame
Middlebury	72	8	100%
KITTI	194	194	∽50%
Sintel	1,041	1,041	100%
Flying Chairs	22,872	22,872	100%

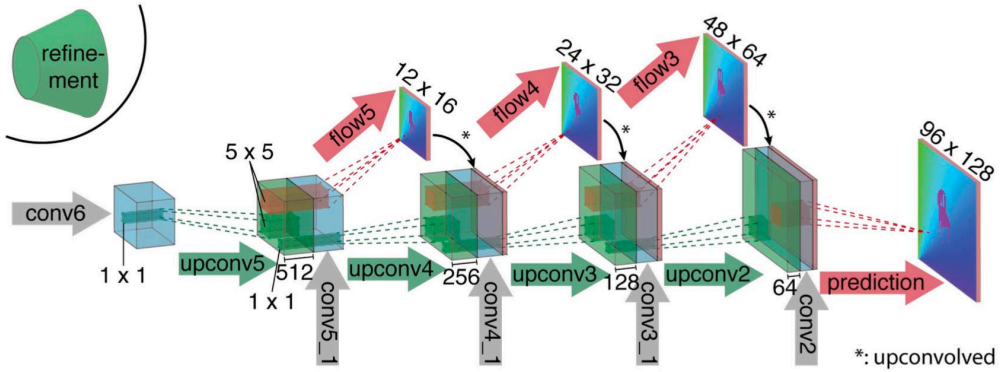
Table 1. Size of already available datasets and the proposed Flying Chairs dataset.

Dataset Generation

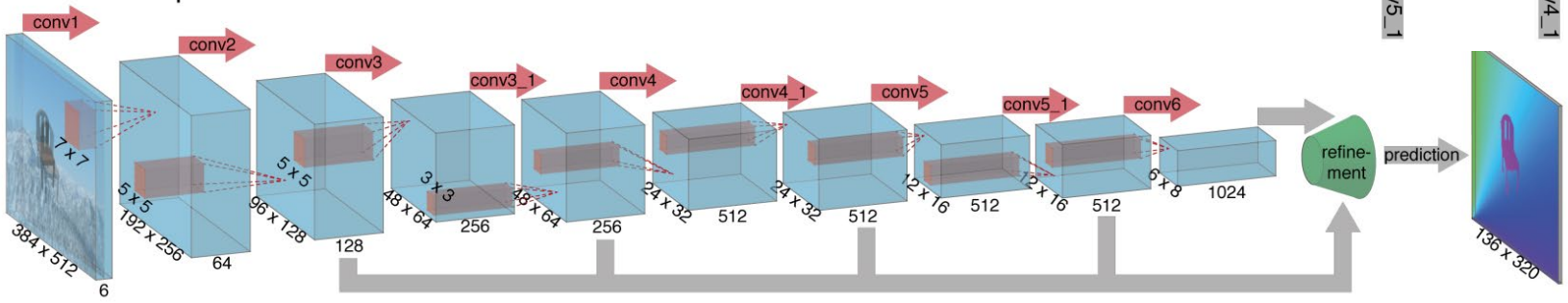
- Flying chair dataset
  - Use online images from Flickr as background (‘city’, ‘landscape’, ‘mountain’)
  - Randomly sample affine transformation parameters for the background and the chairs
    - Camera motion+ object moving
    - Mimic the realistic motion distribution of real dataset
  - Apply heavy regularization
    - e.g. translation, rotation, scaling, gaussian noise, color distortion

Figure 5. Two examples from the Flying Chairs dataset. Generated image pair and color coded flow field (first three columns), augmented image pair and corresponding color coded flow field respectively (last three columns).

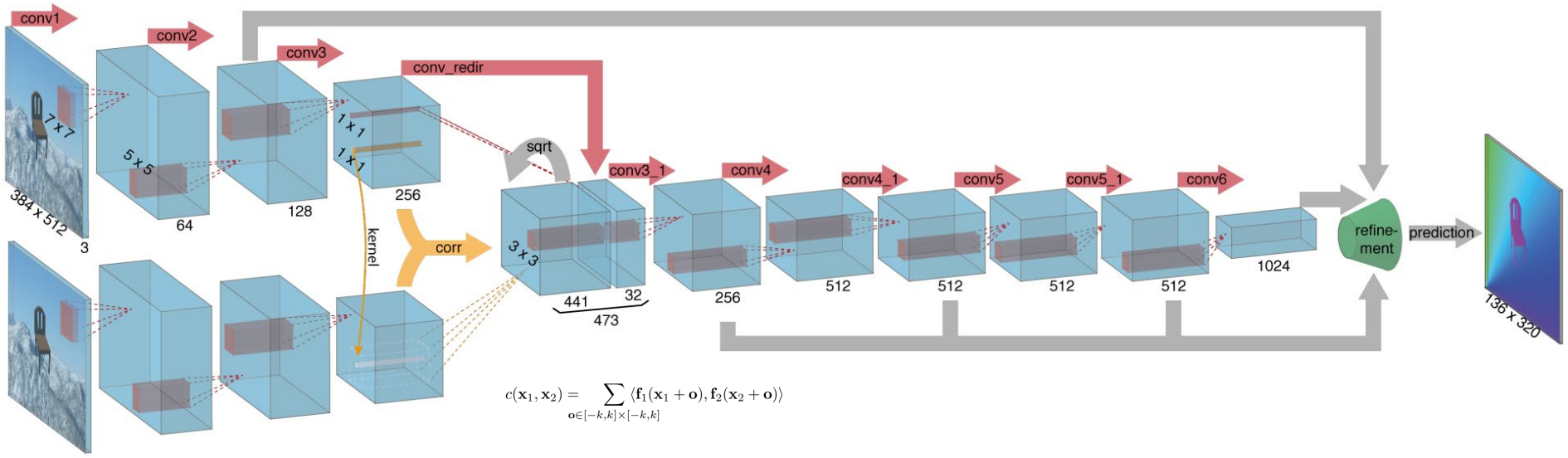
FlowNet Design



FlowNetSimple



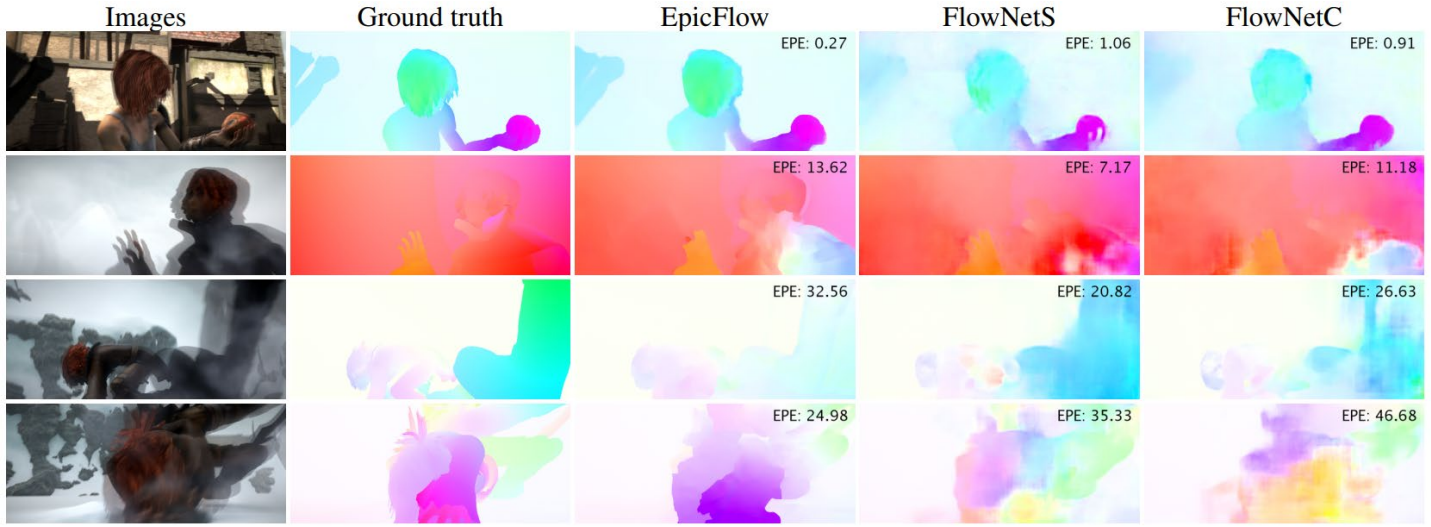
FlowNetCorr



$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

[Fischer, et al.]

Results



Method	Sintel Clean		Sintel Final		KITTI		Middlebury train		Middlebury test		Chairs	Time (sec)	
	train	test	train	test	train	test	AEE	AAE	AEE	AAE	test	CPU	GPU
EpicFlow [30]	2.40	4.12	3.70	6.29	3.47	3.8	0.31	3.24	0.39	3.55	2.94	16	-
DeepFlow [35]	3.31	5.38	4.56	7.21	4.58	5.8	0.21	3.04	0.42	4.22	3.53	17	-
EPPM [3]	-	6.49	-	8.38	-	9.2	-	-	0.33	3.36	-	-	0.2
LDOF [6]	4.29	7.56	6.42	9.12	13.73	12.4	0.45	4.97	0.56	4.55	3.47	65	2.5
FlowNetS	4.50	7.42	5.45	8.43	8.26	-	1.09	13.28	-	-	2.71	-	0.08
FlowNetS+v	3.66	6.45	4.76	7.67	6.50	-	0.33	3.87	-	-	2.86	-	1.05
FlowNetS+ft	(3.66)	6.96	(4.44)	7.76	7.52	9.1	0.98	15.20	-	-	3.04	-	0.08
FlowNetS+ft+v	(2.97)	6.16	(4.07)	7.22	6.07	7.6	0.32	3.84	0.47	4.58	3.03	-	1.05
FlowNetC	4.31	7.28	5.87	8.81	9.35	-	1.15	15.64	-	-	2.19	-	0.15
FlowNetC+v	3.57	6.27	5.25	8.01	7.45	-	0.34	3.92	-	-	2.61	-	1.12
FlowNetC+ft	(3.78)	6.85	(5.28)	8.51	8.79	-	0.93	12.33	-	-	2.27	-	0.15
FlowNetC+ft+v	(3.20)	6.08	(4.83)	7.88	7.31	-	0.33	3.81	0.50	4.52	2.67	-	1.12

Table 2. Average endpoint errors (in pixels) of our networks compared to several well-performing methods on different datasets. The numbers in parentheses are the results of the networks on data they were trained on, and hence are not directly comparable to other results.

[Fischer, et al.]

Side Story – Evolution of FlowNet

- Problem of FlowNet
  - Poor output quality
    - Noisy & blurred result
    - Miss small displacement

- FlowNet 2.0?
  - Stacking multiple FlowNet for high-quality output

FlowNet-2.0

- Key ideas - 1
  - Change training schedule with multiple dataset
    - Conclusion?
      - Easy to hard progressive change
      - Use easy training dataset at the early stage to learn the general concept of color matching without developing possibly confusing priors for 3D motion and realistic lighting too early

Architecture	Datasets	S_{short}	S_{long}	S_{fine}
FlowNetS	Chairs	4.45	-	-
	Chairs	-	4.24	4.21
	Things3D	-	5.07	4.50
	mixed	-	4.52	4.10
	Chairs→Things3D	-	4.24	**3.79**
FlowNetC	Chairs	3.77	-	-
	Chairs→Things3D	-	3.58	**3.04**

[Ilg et al.]

FlowNet-2.0

- Key ideas - 2
  - Stacking multiple layers for refinement
    - Most of the previous methods adopt iterative refinement – mimic it on neural network
    - Prepare specialized network for small displacement detection

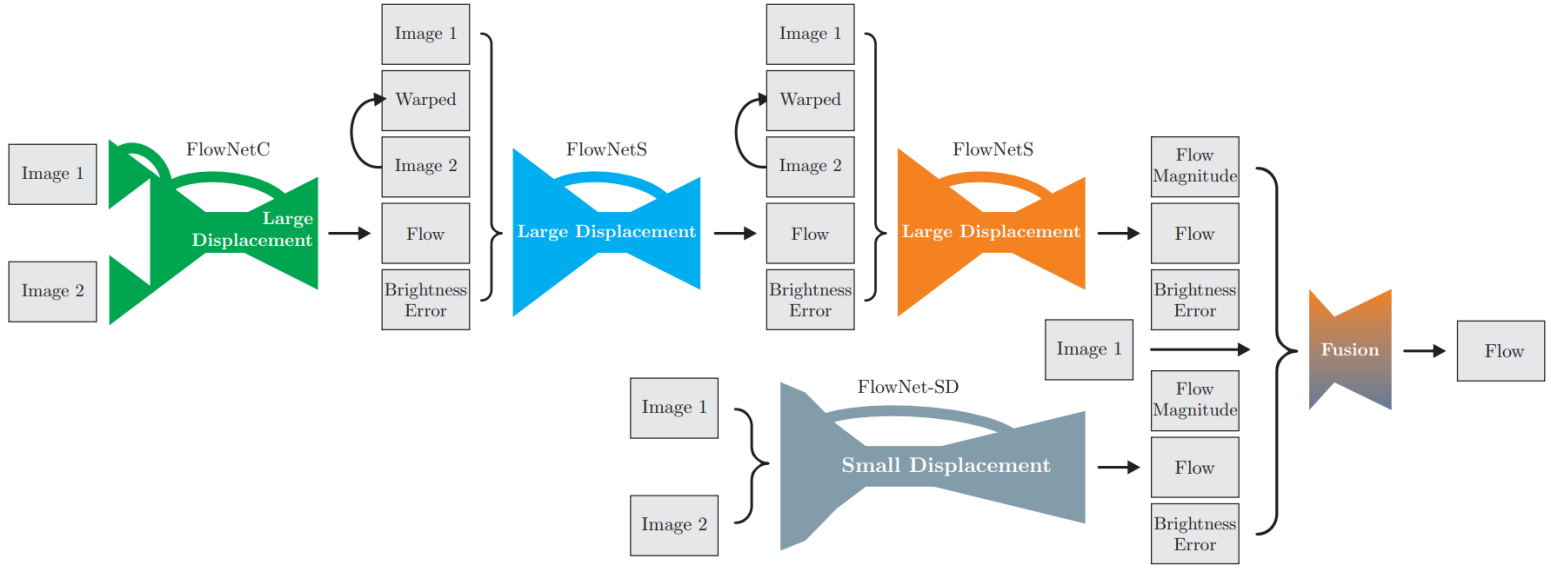
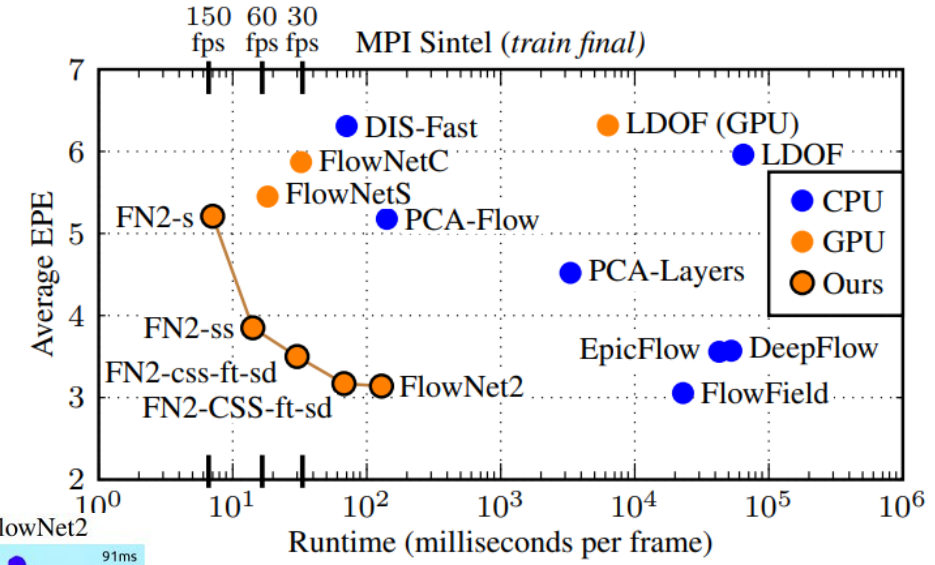
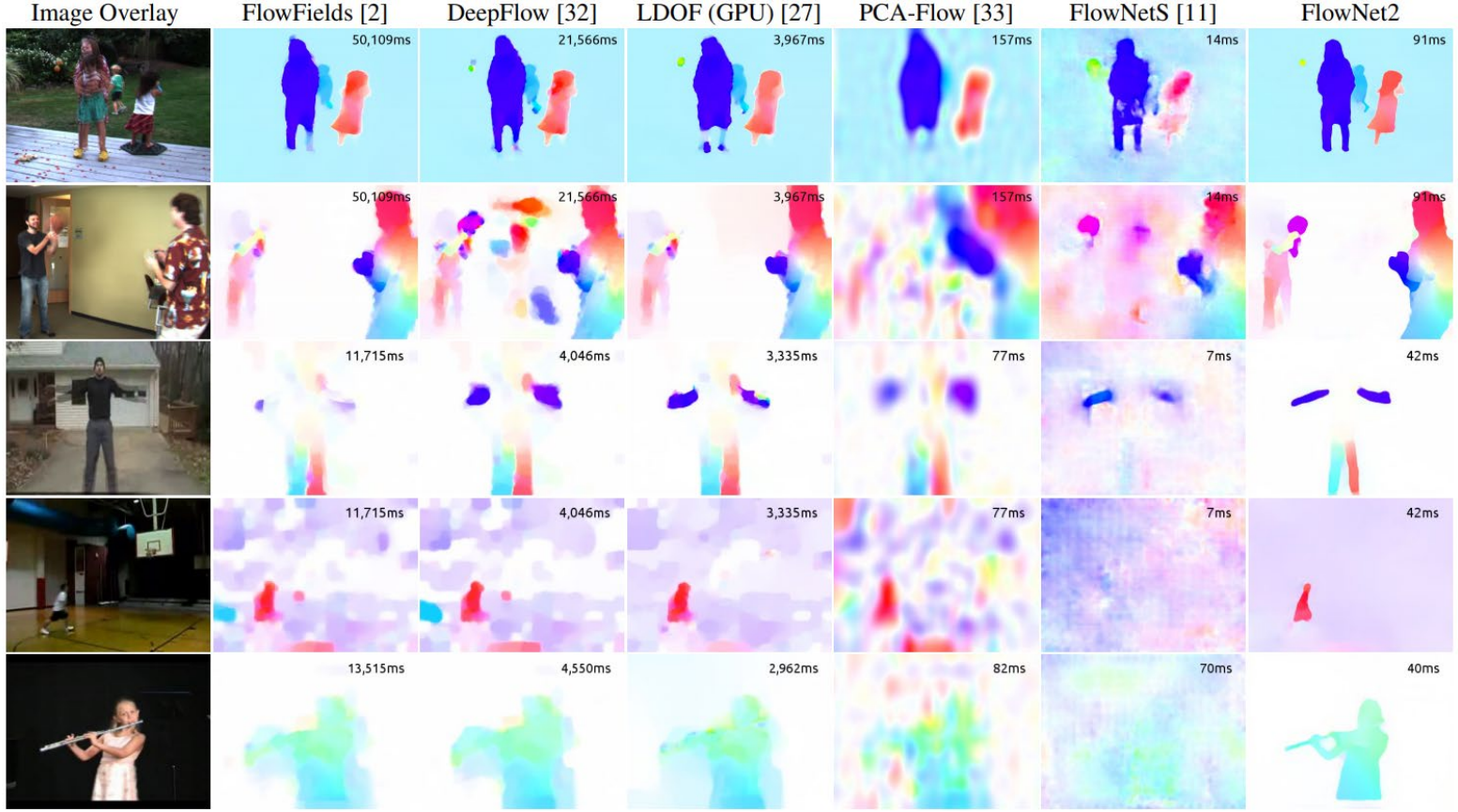


Figure 2. Schematic view of complete architecture: To compute large displacement optical flow we combine multiple FlowNets. Braces indicate concatenation of inputs. *Brightness Error* is the difference between the first image and the second image warped with the previously estimated flow. To optimally deal with small displacements, we introduce smaller strides in the beginning and convolutions between upconvolutions into the FlowNetS architecture. Finally we apply a small fusion network to provide the final estimate.

[Ilg et al.]

Results

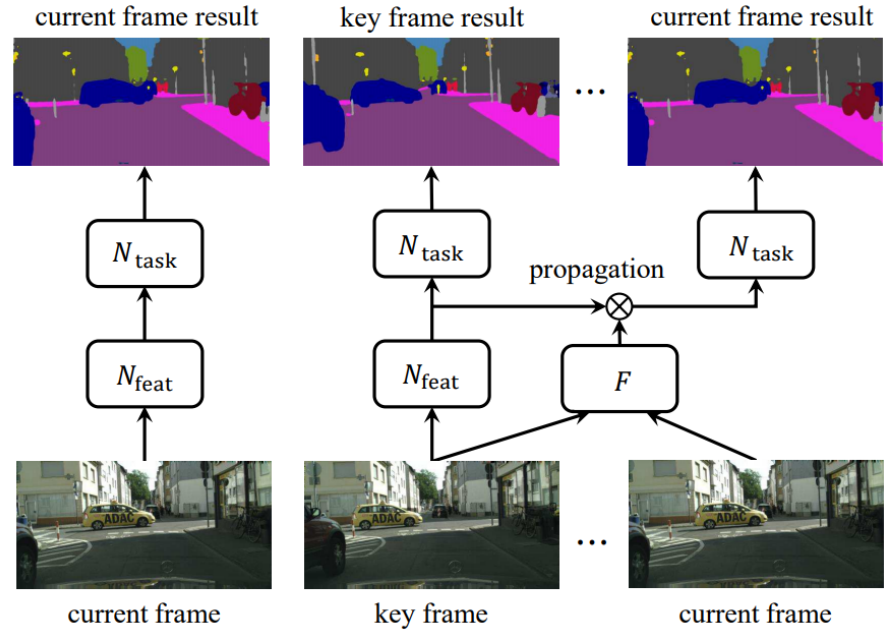




[Ilg et al.]

Application of Optical Flow for Optimization

- Video data has temporal correlation between the adjacent frames
  - We may not need to run our networks at every frames
  - We could estimate the feature of the current frame by transforming the feature of the key frame using optical flow



(a) per-frame network (b) deep feature flow (DFF) network

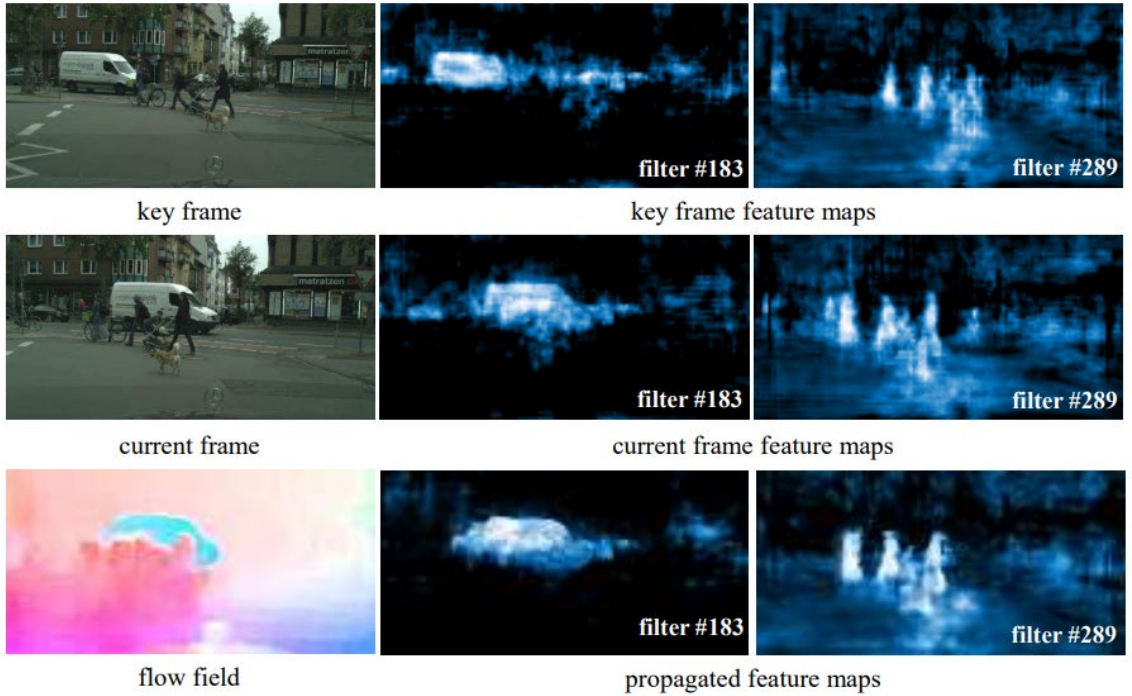


Figure 1. Motivation of proposed *deep feature flow* approach. Here we visualize the two filters' feature maps on the last convolutional layer of our ResNet-101 model (see Sec. 4 for details). The convolutional feature maps are similar on two nearby frames. They can be cheaply propagated from the key frame to current frame via a flow field.

[Zhou et al.]

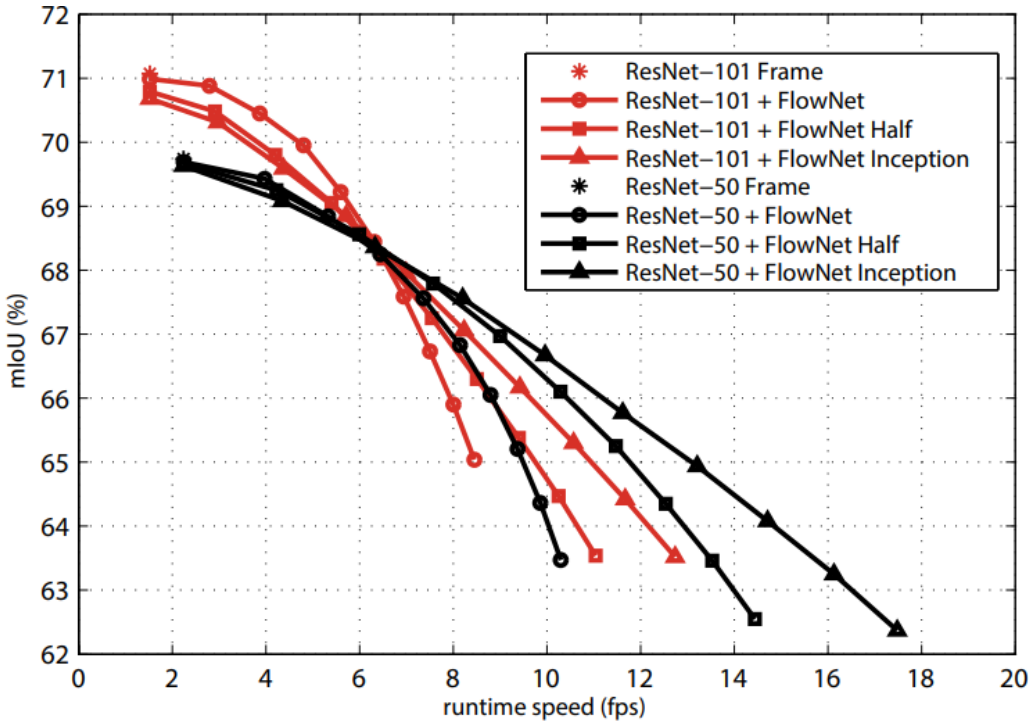
Deep Feature Flow

- Optical flow is obtained by FlowNet
- Bilinear interpolation used for feature propagation
  - Optical flow is scaled to the spatial dimension of feature map (of the last CONV layer)

$$\mathbf{f}_i^c(\mathbf{p}) = \sum G(\mathbf{q}, \mathbf{p} + \delta\mathbf{p})\mathbf{f}_k^c(\mathbf{q})$$

$$G(\mathbf{q}, \mathbf{p} + \delta\mathbf{p}) = g(q_x, p_x + \delta p_x) \cdot g(q_y, p_y + \delta p_y)$$

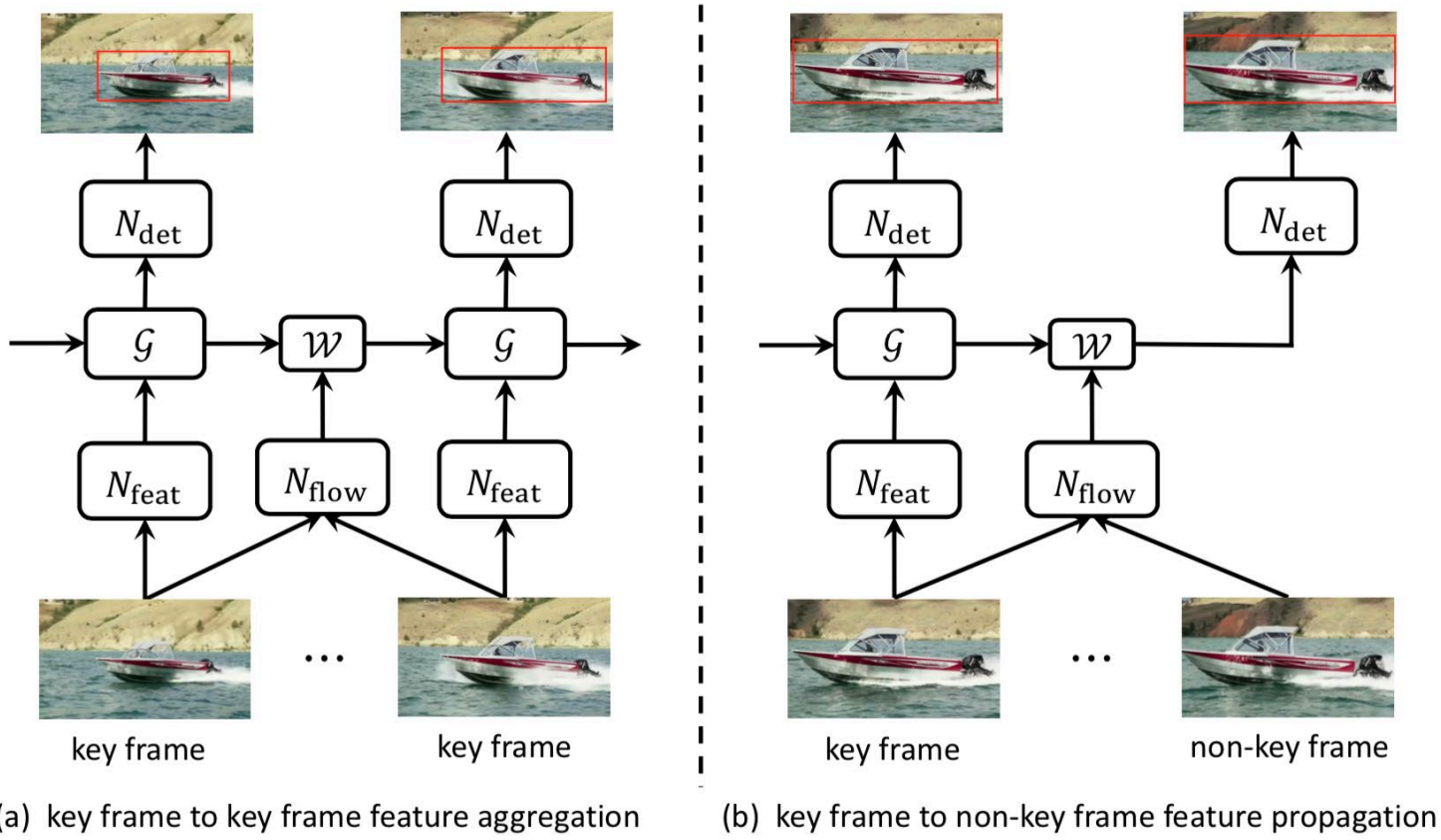
$$g(a, b) = max(0, 1 - |a - b|)$$



[Zhou et al.]

Towards High Performance Video Object Detection for Mobiles

- Basically, feature propagation for non-key frames
- Light optical flow
  - Smaller network for optical flow computation
- GRU-based feature aggregation
  - Feature aggregation over only key frames
  - More robust to deblur, …

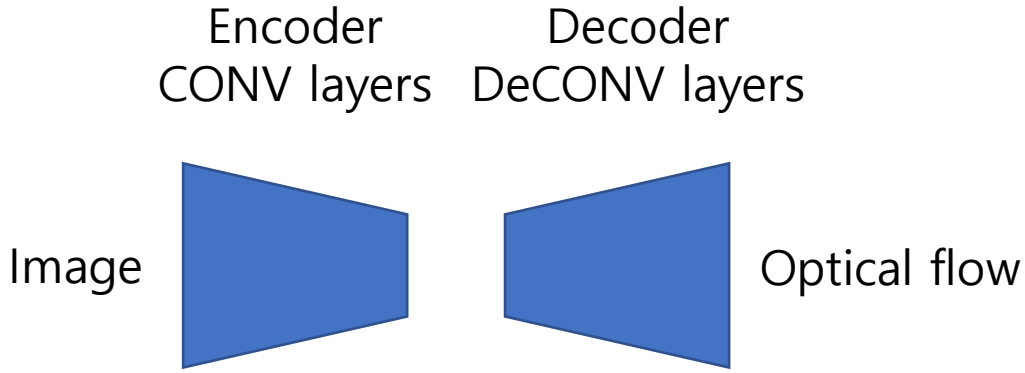


(a) key frame to key frame feature aggregation

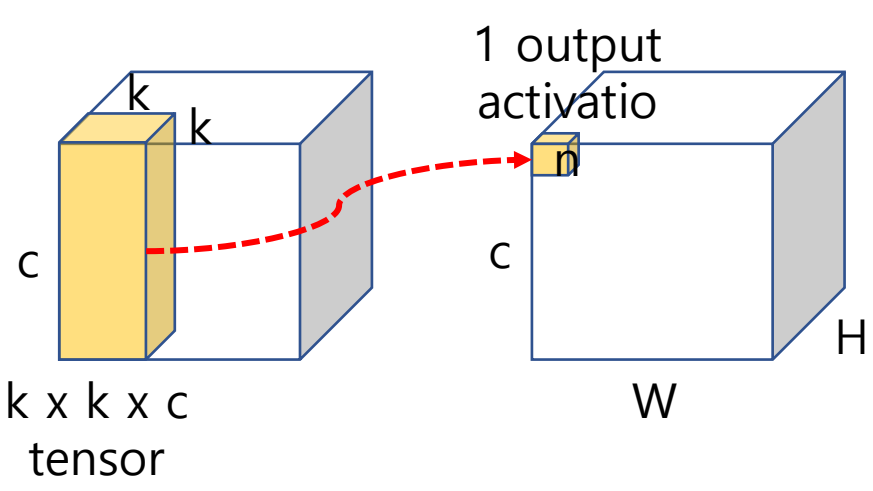
(b) key frame to non-key frame feature propagation

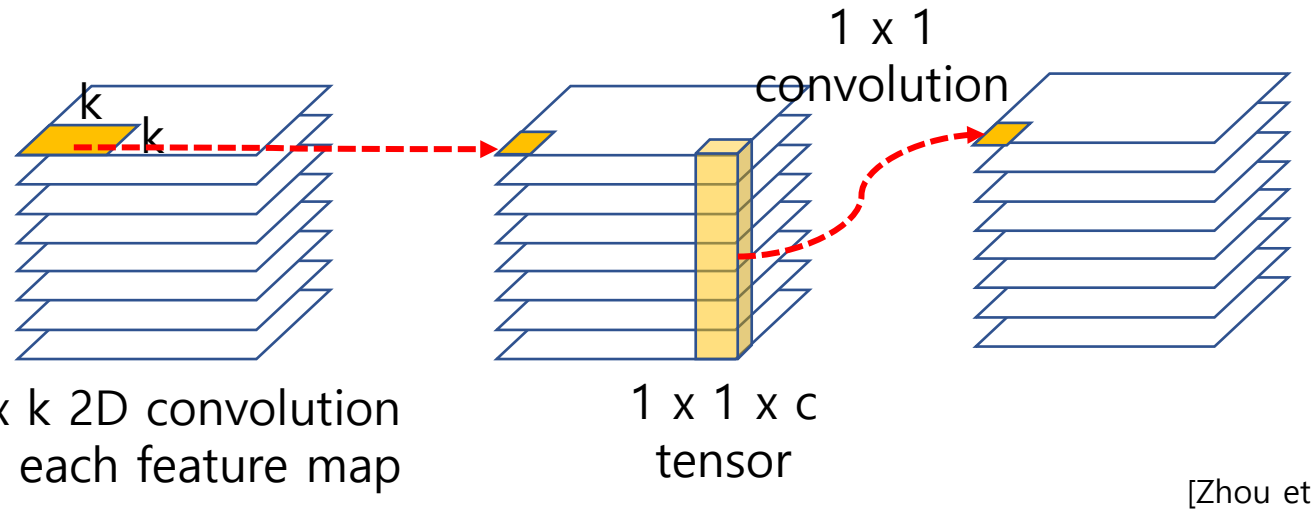
[Zhou et al.]

Light Flow



- Original FlowNet
  - Encoder + decoder
- Light Flow
  - 3D convolution → depthwise (DW) separable convolution in both encoder/decoder
  - Less multiplications: kxkxc x HxWxc vs. kxkxHxWxc + HxWxcxc, i.e., k^2c vs k^2+c
    - 3x3 conv on 100 channels: 900 vs. 109 multiplications → 9X reduction in computation

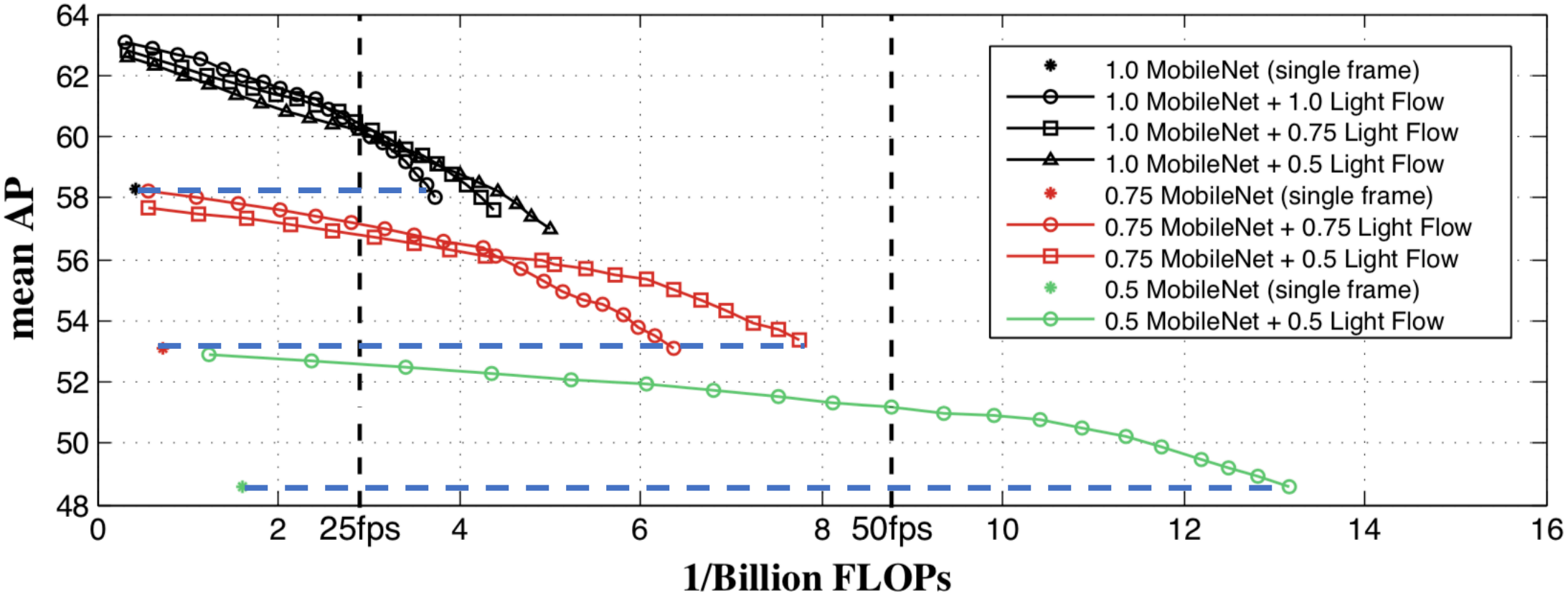




[Zhou et al.]

Key Results: >10X Speedup & Same mAP

- Key frame duration from 1 to 20



[Zhou et al.]

Training with Synthetic Data – Revisit

- Training network with real data and the human-annotated label is expensive
- As an alternative, synthetic data-based training has been proposed
  - Pros
    - Relatively easy to increase training dataset size
    - Fast & accurate annotation based on generation information
      - Segmentation, bounding box, depth, instance id, …
  - Cons
    - Effort to create a synthesize environment
    - Distribution difference between real and synthetic data
      - Network may learn the unexpected feature in the synthetic data

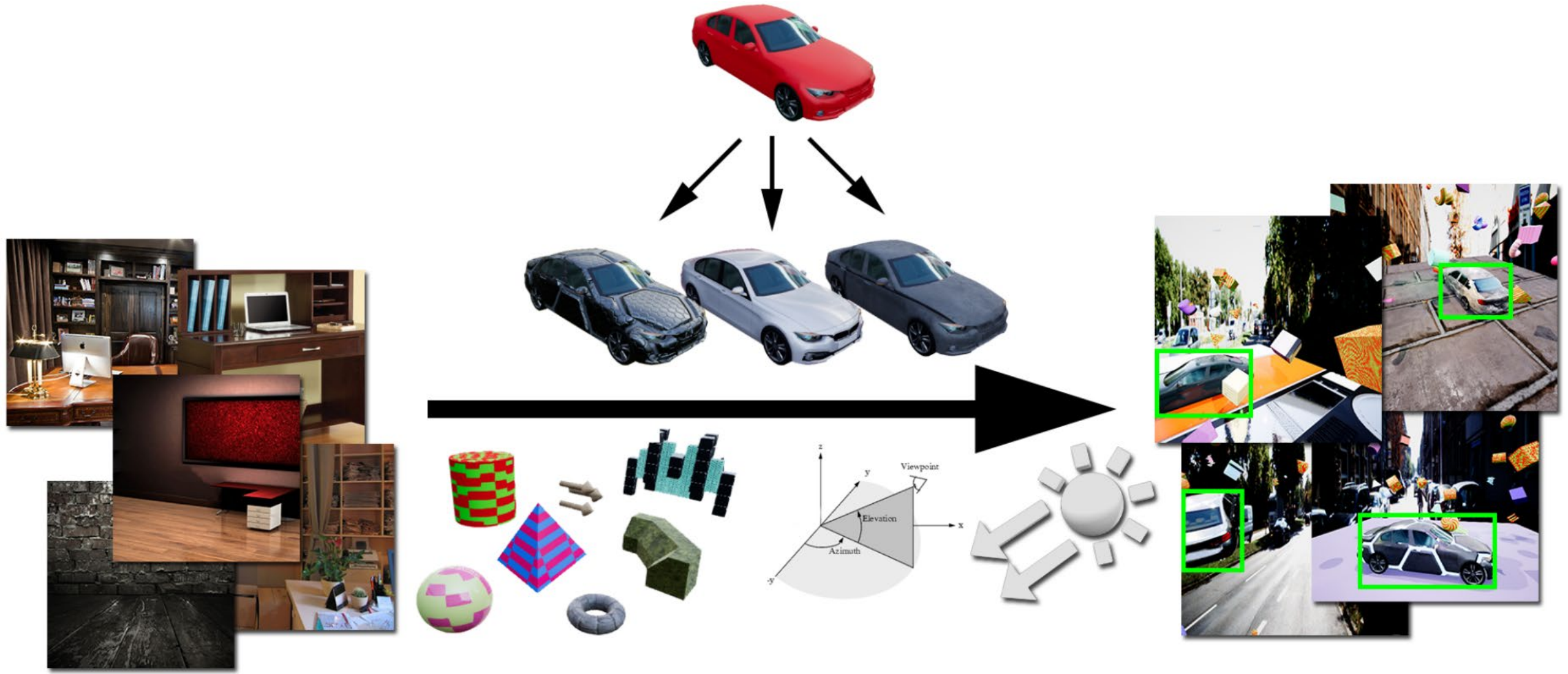
Domain Randomization

- Domain randomization is proposed to close the reality gap
  - Generating synthetic data with sufficient variation
  - The network views real-world data as just another variation

  - Random number of geometric shapes are added
    - Flying distractors
  - Random textures are applied to both the objects and flying distractors
  - Random number of lights of different types are inserted at random locations
  - The scene is rendered from a random camera viewpoint
  - The result is composed over a random background image

  - Generation pipeline uses Unreal Engine (UE4)

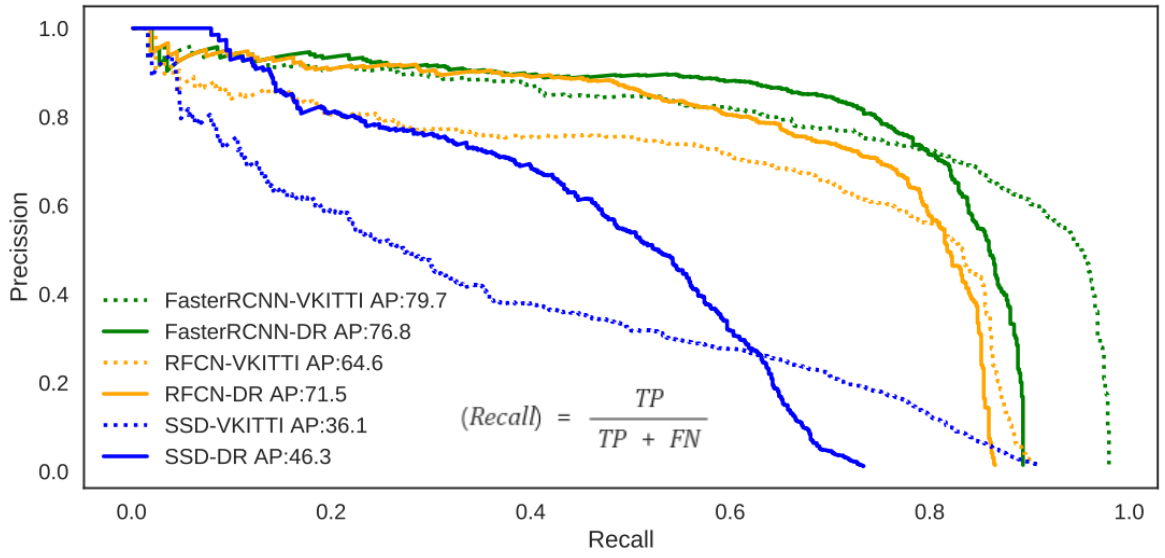
Generated Image Examples

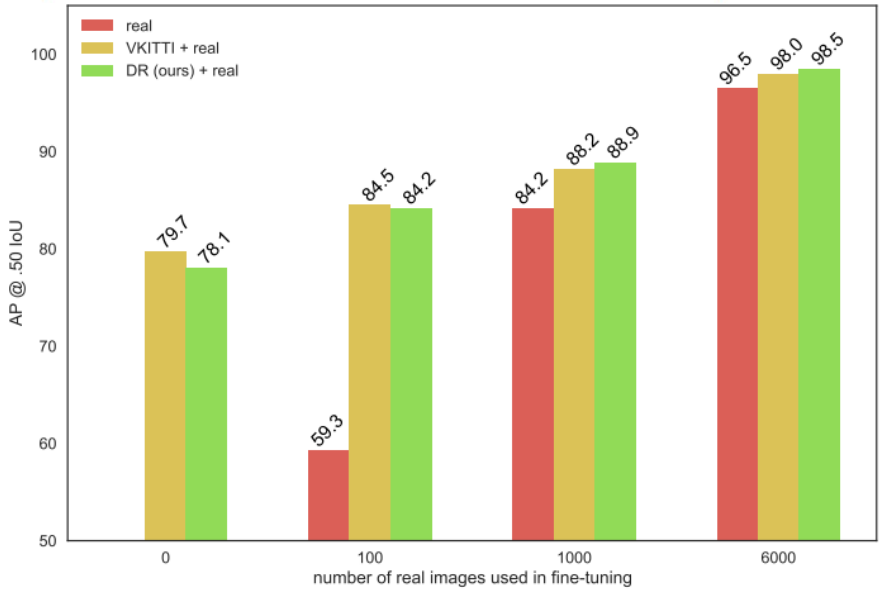


Results

$$(Precision) = \frac{TP}{TP + FP}$$

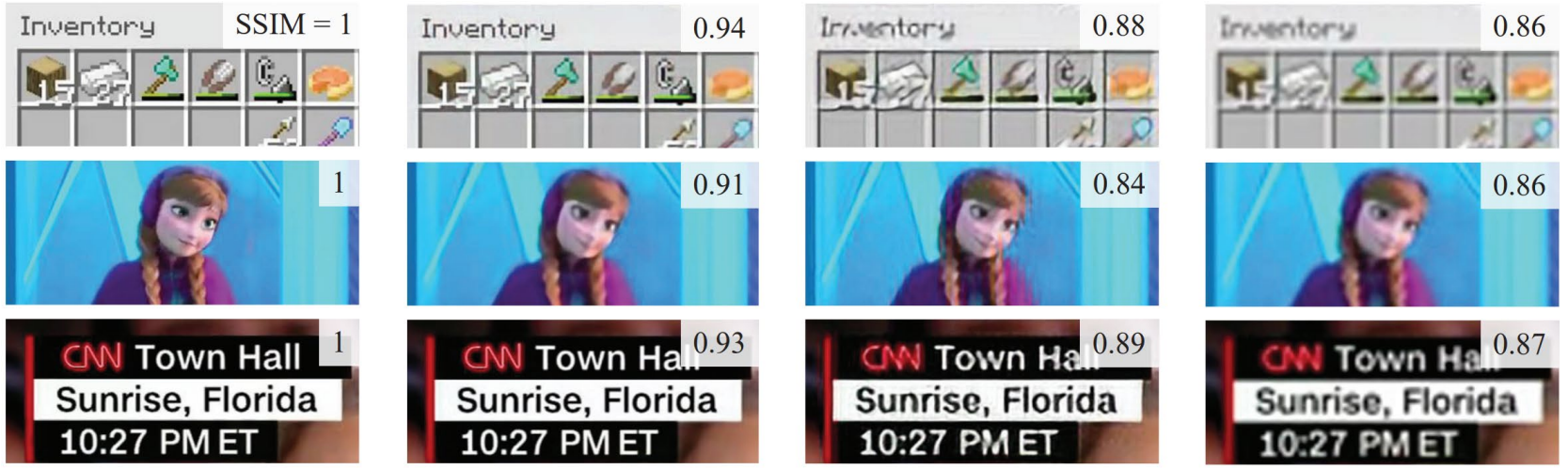
Figure 2. Sample images from Virtual KITTI (first row), and our DR approach (second row). Note that our images are easier to create (because of their lack of fidelity) and yet contain more variety to force the deep neural network to focus on the structure of the objects of





Neural Adaptive Content-aware Internet Video Delivery

- Video delivery requires large amount of bandwidth
  - When the bandwidth resource becomes scarce, user quality of experience suffers
  - DNN-based quality enhancement in limited bandwidth
    - learns a mapping from a low-quality video to a high-quality version, e.g., super resolution.
    - Content-aware DNN – use domain-specific network for each video



(a) Original (1080p) (b) Content-aware DNN (c) Content-agnostic DNN (d) 240p

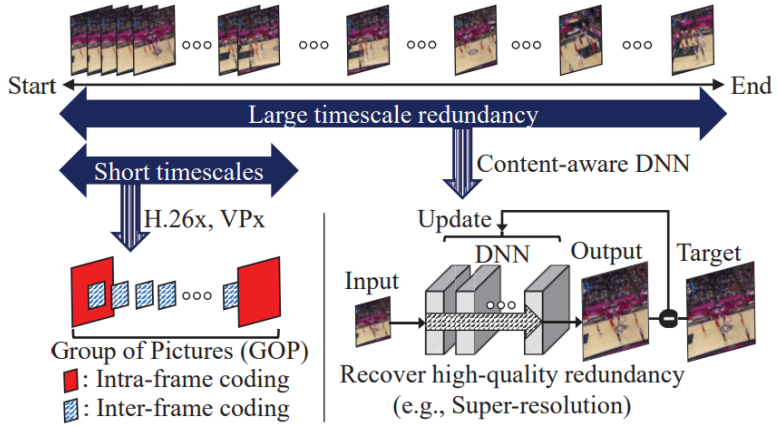


Figure 3: Content-aware DNN based video encoding

Results

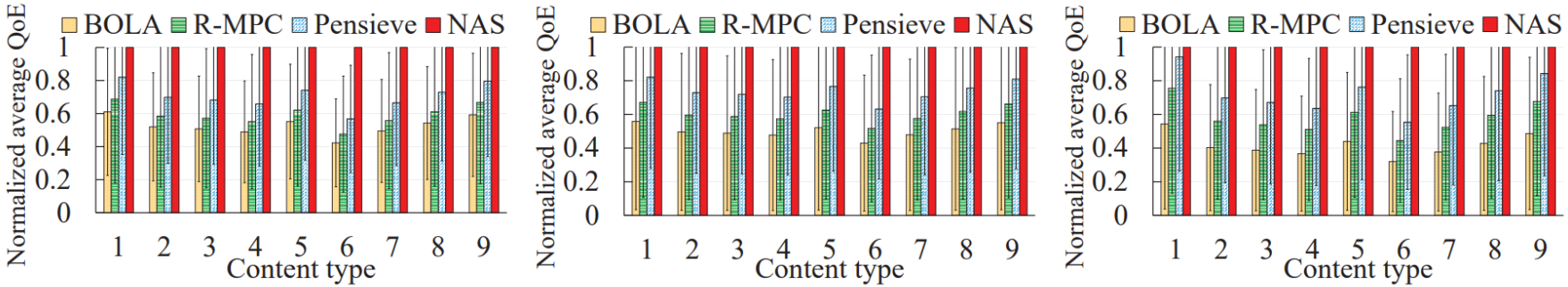


Figure 6: Normalized QoE comparison of video clips from the nine content categories of YouTube. (1: Beauty, 2: Comedy, 3: Cook, 4: Entertainment, 5: Game, 6: Music, 7: News, 8: Sports, 9: Technology)

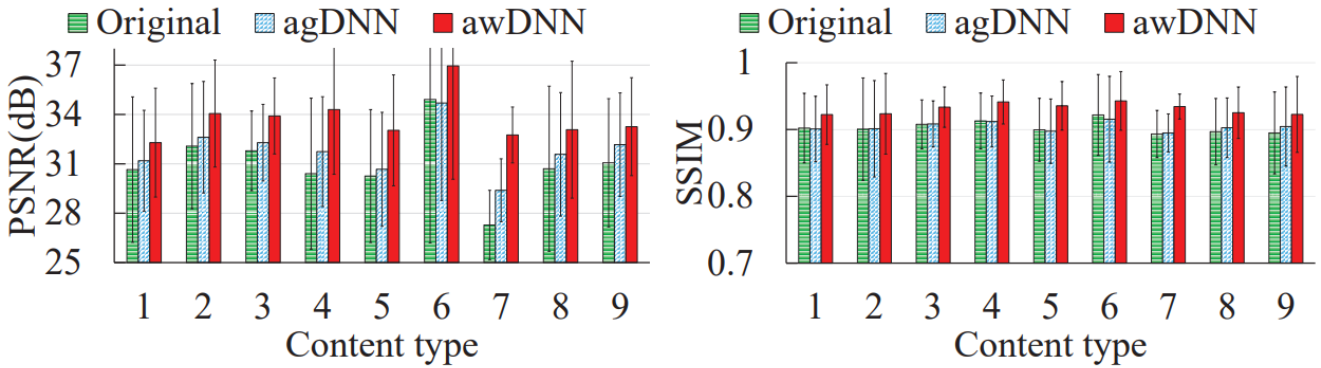


Figure 11: Video quality in PSNR and SSIM (240p input)

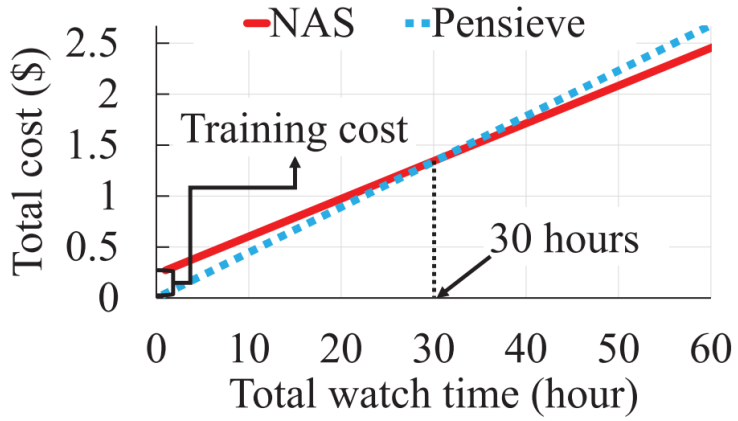
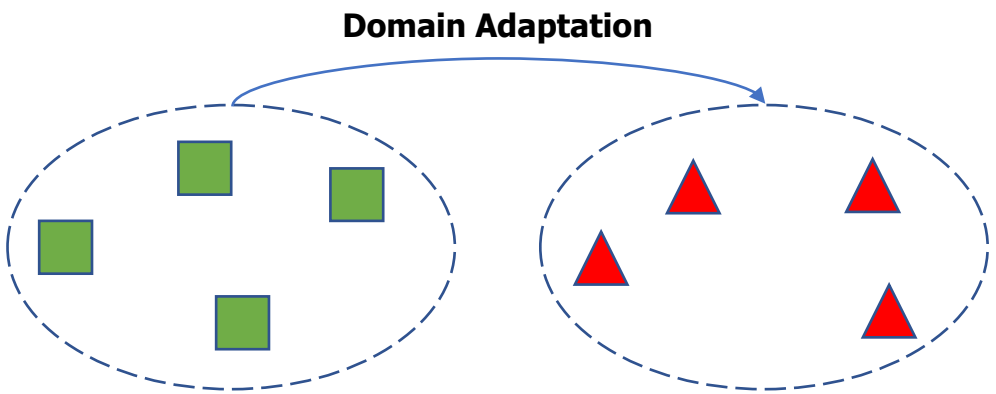


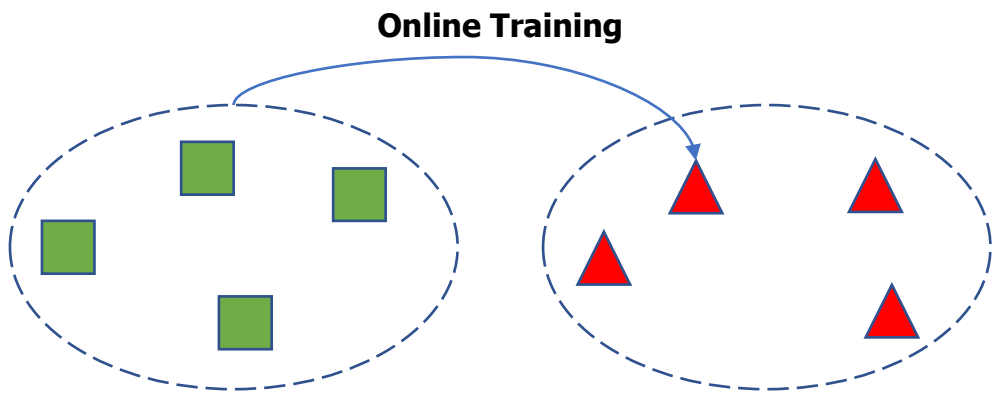
Figure 10: Cumulative server cost

Online Training

Domain Adaptation vs Online Training

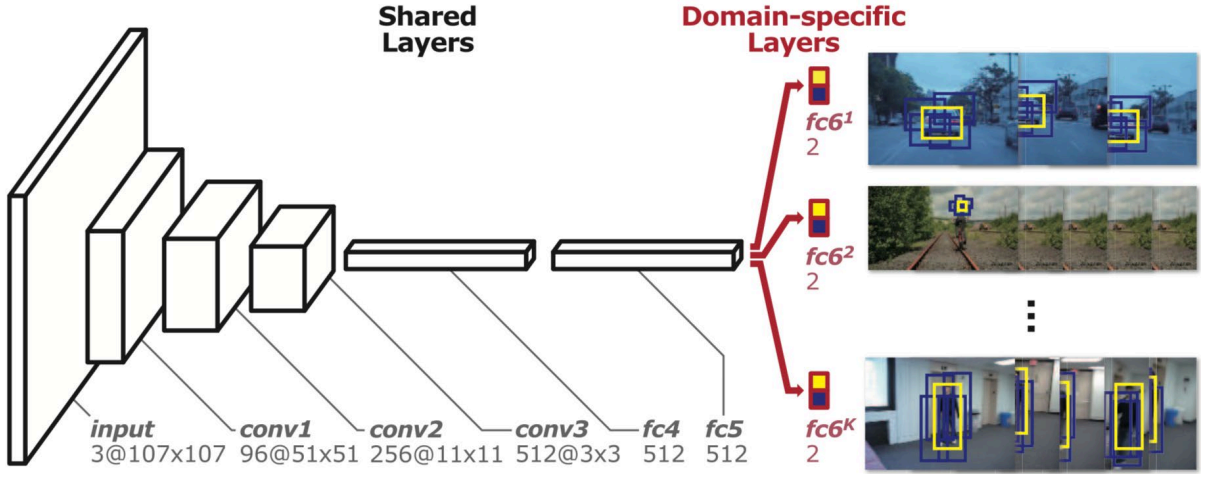
- Online training is a subcategory of domain adaptation
  - Focus on narrow distribution – single sequence of images
    - Tradeoff between accuracy and generalization
  - Based on on-device learning
    - A few iterations of SGD may be required





Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

- Object tracking task: keep tracking the object selected at the first frame
- MDNet (Multi-domain network)
  - Domain = video clip
  - Pre-training
    - K training video clips → K FC6 layers
      - For a k-th training video clip, only FC6k is trained while the other FC6 branches are not trained
    - CONV layers
      - Learn common representation
    - FC layers
      - Learn video-specific representation



[Hyeonseob Nam, Bohyung Han]

Online Training

- Train only FC layers
  - Randomly initialize FC6
  - Train with ground truth on 1st frame
- Periodically or if score < 0.5, train with positive/negative samples collected by then
- Generate candidates near true box based on Gaussian distribution
- Positive/negative samples
  - IoU > 0.7 & IoU < 0.3 among candidates

Algorithm 1 Online tracking algorithm

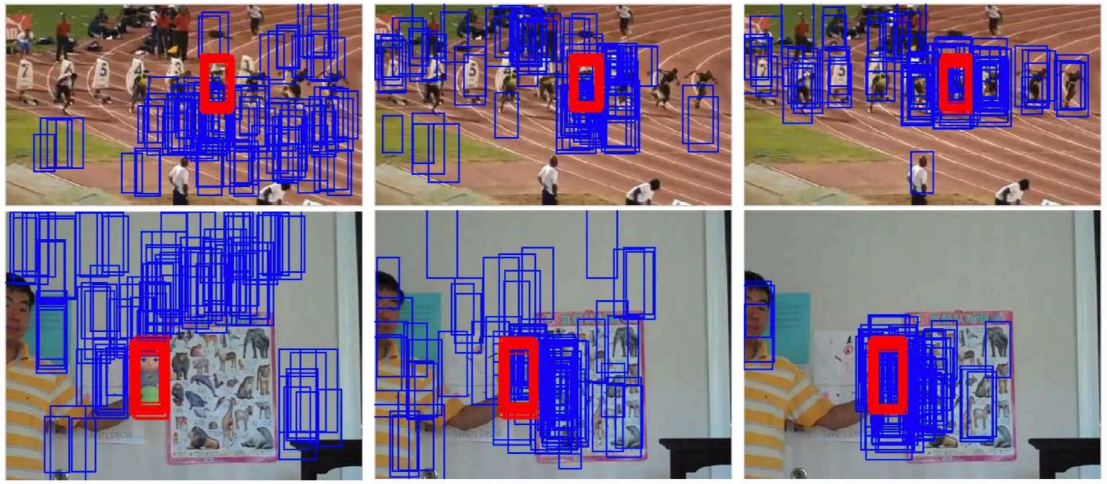
Input : Pretrained CNN filters $\{\mathbf{w}_1, \ldots, \mathbf{w}_5\}$
Initial target state $\mathbf{x}_1$
Output: Estimated target states $\mathbf{x}_t^*$

Randomly initialize the last layer $\mathbf{w}_6$.
Train a bounding box regression model.
Draw positive samples S_1^+ and negative samples S_1^-.
Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using S_1^+ and S_1^-;
$\mathcal{T}_s \leftarrow \{1\}$ and $\mathcal{T}_l \leftarrow \{1\}$.
repeat
  Draw target candidate samples $\mathbf{x}_t^i$.
  Find the optimal target state $\mathbf{x}_t^*$ by Eq. (1).
  if $f^+(\mathbf{x}_t^*) > 0.5$ **then**
    Draw training samples S_t^+ and S_t^-.
    $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{t\}$, $\mathcal{T}_l \leftarrow \mathcal{T}_l \cup \{t\}$.
    if $|\mathcal{T}_s| > \tau_s$ **then** $\mathcal{T}_s \leftarrow \mathcal{T}_s \setminus \{\min_{v \in \mathcal{T}_s} v\}$.
    if $|\mathcal{T}_l| > \tau_l$ **then** $\mathcal{T}_l \leftarrow \mathcal{T}_l \setminus \{\min_{v \in \mathcal{T}_l} v\}$.
    Adjust $\mathbf{x}_t^*$ using bounding box regression.
  if $f^+(\mathbf{x}_t^*) < 0.5$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S^+_{v \in \mathcal{T}_s}$ and $S^-_{v \in \mathcal{T}_s}$.
  else if $t \bmod 10 = 0$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S^+_{v \in \mathcal{T}_l}$ and $S^-_{v \in \mathcal{T}_s}$.
until end of sequence

[Hyeonseob Nam, Bohyung Han]

Online Training

- 250 positive/negative samples/training frame
- First frame: 30 iterations
- Other frame: 10 iterations with 3X larger learning rate



(a) 1^{st} minibatch (b) 5^{th} minibatch (c) 30^{th} minibatch

Algorithm 1 Online tracking algorithm

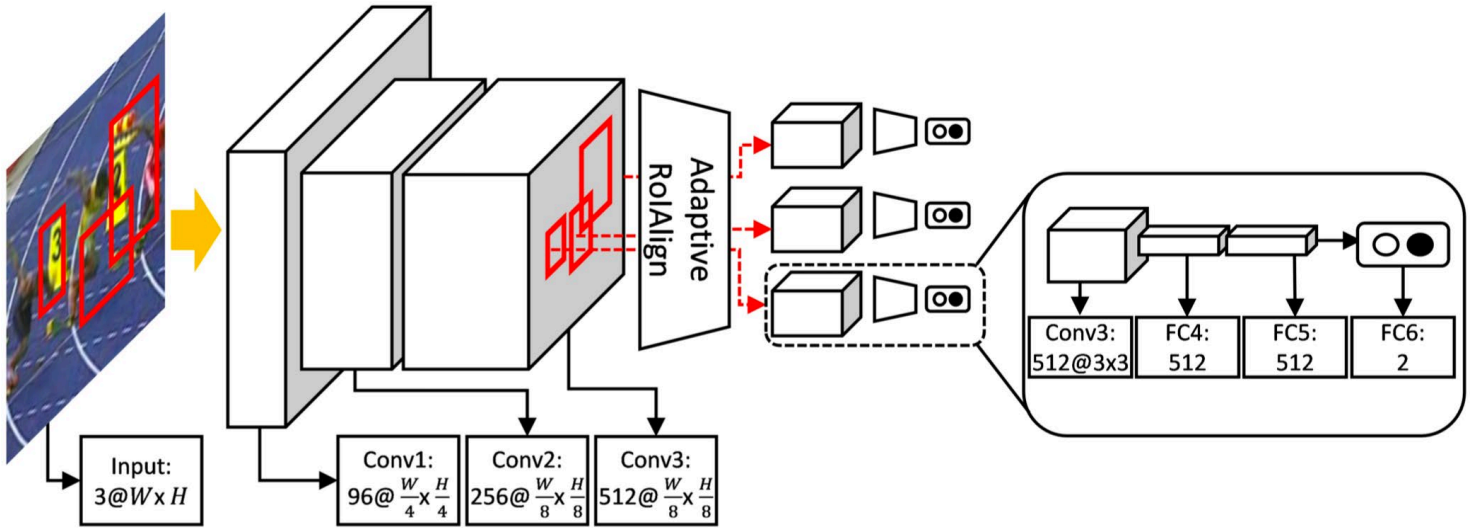
Input : Pretrained CNN filters $\{\mathbf{w}_1, \ldots, \mathbf{w}_5\}$
Initial target state $\mathbf{x}_1$
Output: Estimated target states $\mathbf{x}_t^*$

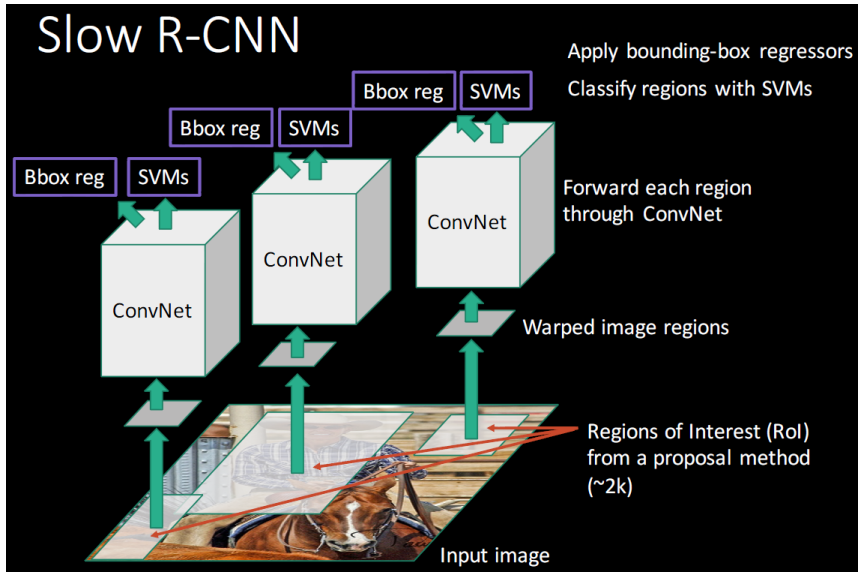
Randomly initialize the last layer $\mathbf{w}_6$.
Train a bounding box regression model.
Draw positive samples S_1^+ and negative samples S_1^-.
Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using S_1^+ and S_1^-;
$\mathcal{T}_s \leftarrow \{1\}$ and $\mathcal{T}_l \leftarrow \{1\}$.
repeat
  Draw target candidate samples $\mathbf{x}_t^i$.
  Find the optimal target state $\mathbf{x}_t^*$ by Eq. (1).
  if $f^+(\mathbf{x}_t^*) > 0.5$ **then**
    Draw training samples S_t^+ and S_t^-.
    $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{t\}$, $\mathcal{T}_l \leftarrow \mathcal{T}_l \cup \{t\}$.
    if $|\mathcal{T}_s| > \tau_s$ **then** $\mathcal{T}_s \leftarrow \mathcal{T}_s \setminus \{\min_{v \in \mathcal{T}_s} v\}$.
    if $|\mathcal{T}_l| > \tau_l$ **then** $\mathcal{T}_l \leftarrow \mathcal{T}_l \setminus \{\min_{v \in \mathcal{T}_l} v\}$.
    Adjust $\mathbf{x}_t^*$ using bounding box regression.
  if $f^+(\mathbf{x}_t^*) < 0.5$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S_{v \in \mathcal{T}_s}^+$ and $S_{v \in \mathcal{T}_s}^-$.
  else if $t \bmod 10 = 0$ **then**
    Update $\{\mathbf{w}_4, \mathbf{w}_5, \mathbf{w}_6\}$ using $S_{v \in \mathcal{T}_l}^+$ and $S_{v \in \mathcal{T}_s}^-$.
until end of sequence

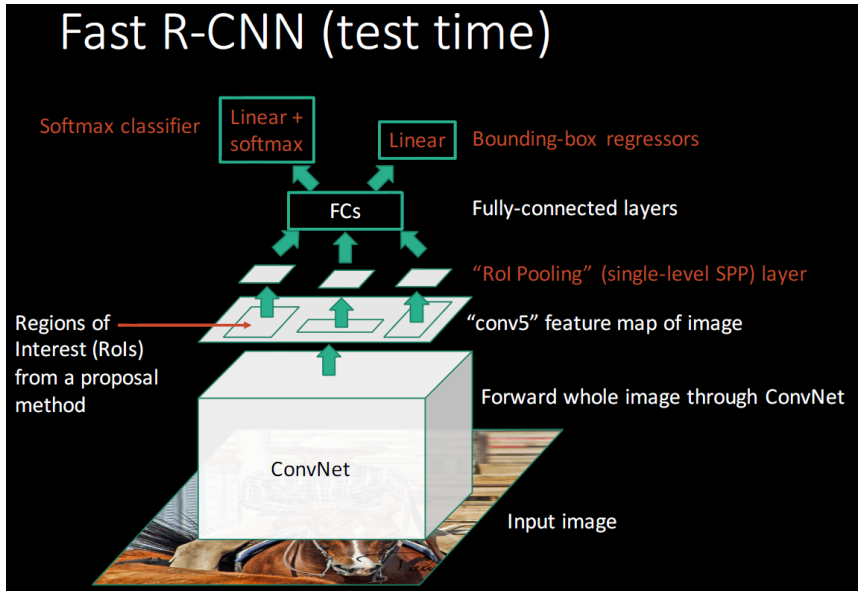
[Hyeonseob Nam, Bohyung Han]

Real-Time MDNet

- 25x Speedup over MDNet
- Draw candidate boxes at the feature map of CONV3 layer
  - Like Fast R-CNN w.r.t. R-CNN



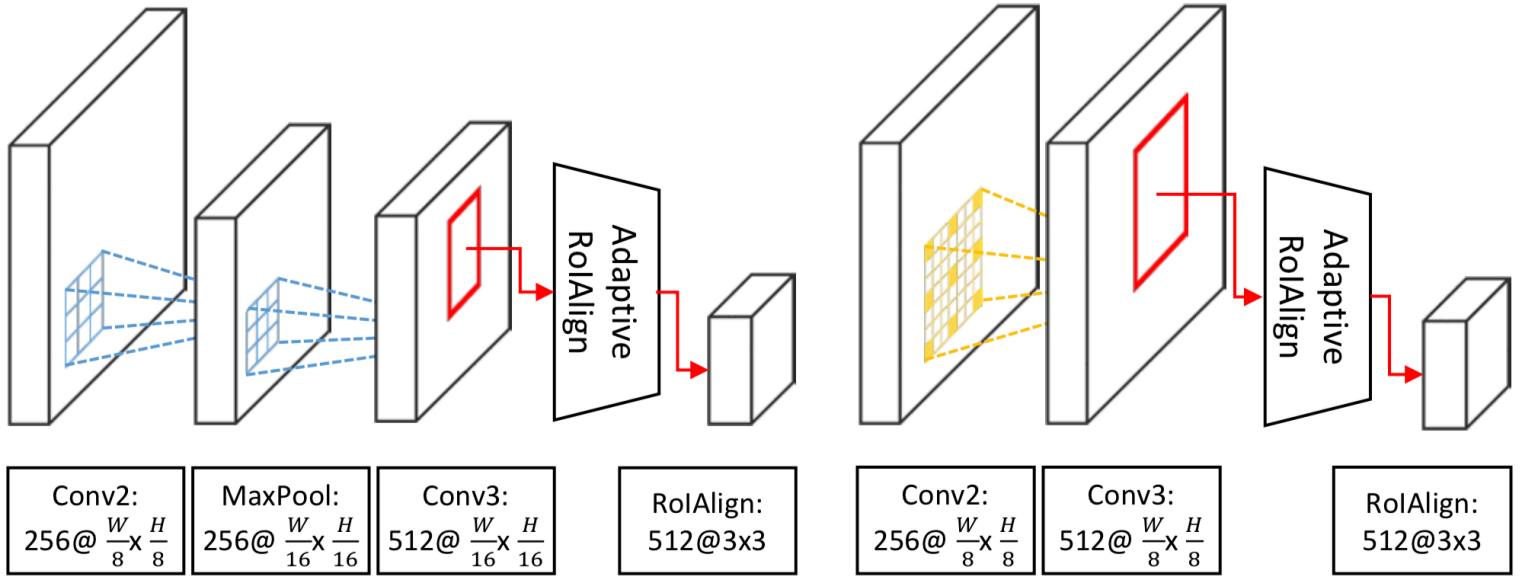




[Jung et al.]

Denser Feature Map Obtained with Dilated Convolution

- Higher spatial dimension is beneficial, but expensive in conventional CONV
- Adopt dilated convolution to increase receptive field without increasing computation cost

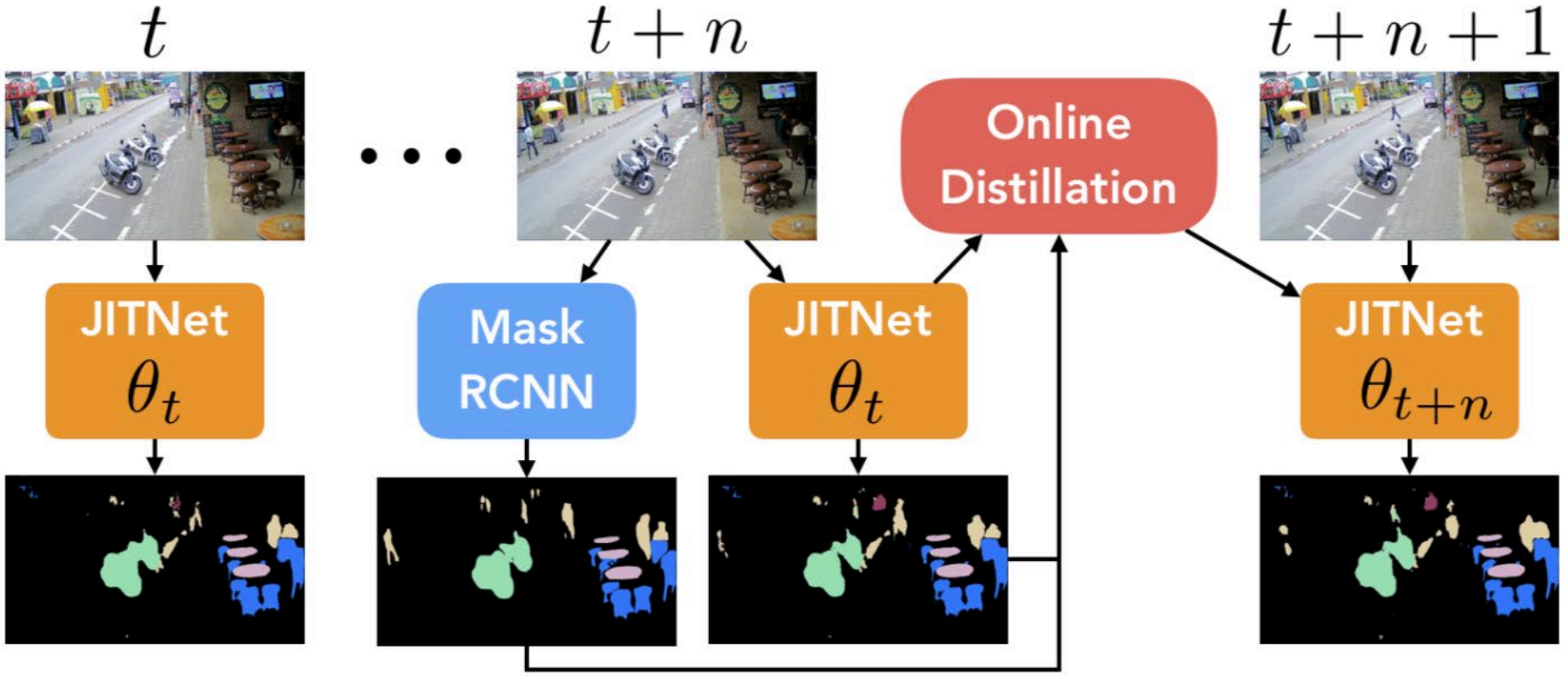


(a) Original VGG-M network (b) Network for dense feature map

[Jung et al.]

Online Model Distillation

- Online learning + knowledge distillation
  - Run a large model on a key frame and obtain high-quality results
  - Train a small model on them and run it on the following frames



[Mullapudi, 2018]

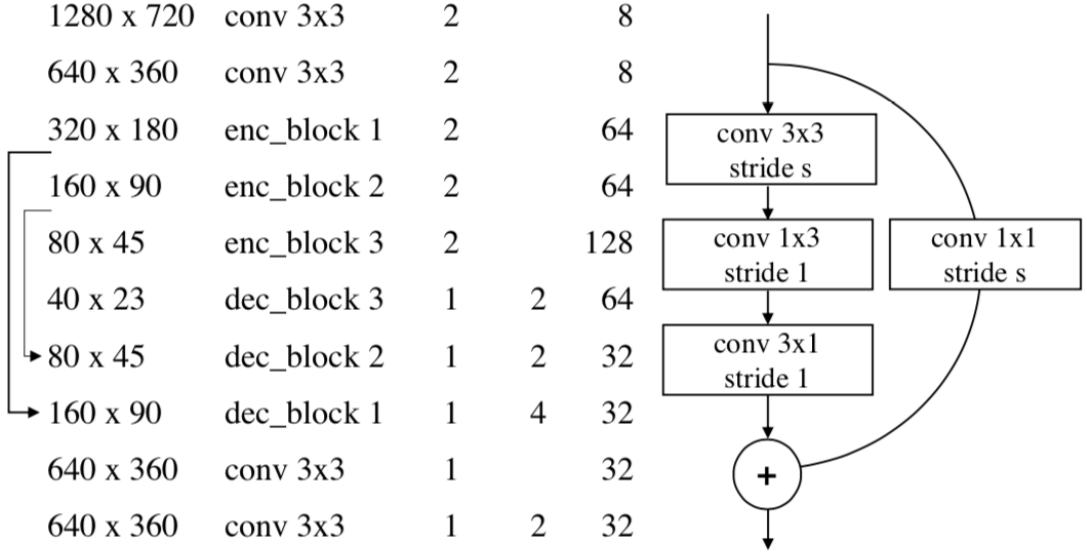
Small JITNet, Runtime and Key Frame Adjustment

- If JITNet accuracy > a threshold, double the key frame period
- Else, half the key frame period
- Key frame period = 8~64 frames

Model	FLOPS (B)		Params (M)	Time (ms)	
	Infer	Train		Infer	Train
JITNet	15.2	32.2	3	7	30
MRCNN	1390.0	-	141	300	-

Table 1: FLOPS (inference, training), parameter count, and runtime for both JITNet and MRCNN. JITNet has 47× fewer parameters and requires 91× (inference) and 40× (training) fewer FLOPS than MRCNN inference.

Input Size	Operation	s	r	c
1280 x 720	conv 3x3	2		8
640 x 360	conv 3x3	2		8
320 x 180	enc_block 1	2		64
160 x 90	enc_block 2	2		64
80 x 45	enc_block 3	2		128
40 x 23	dec_block 3	1	2	64
80 x 45	dec_block 2	1	2	32
160 x 90	dec_block 1	1	4	32
640 x 360	conv 3x3	1		32
640 x 360	conv 3x3	1	2	32
1280 x 720	conv 1x1	1		32



[Mullapudi, 2018]

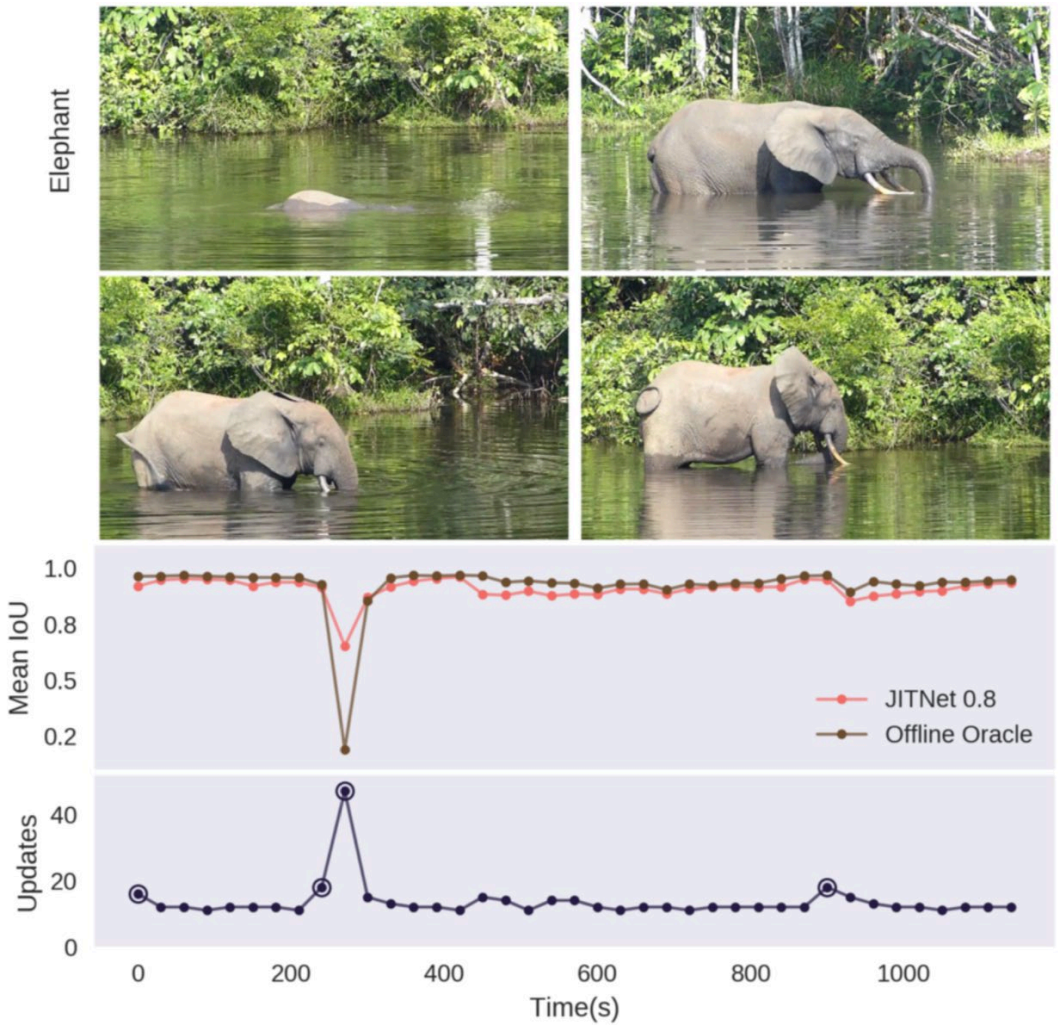
Significant Runtime Speedup with Better Accuracy

Video	Offline	Flow		Online Distillation		
	Oracle (20%)	Slow (2.2×) (12.5%)	Fast (3.2×) (6.2%)	JITNet 0.7	JITNet 0.8	JITNet 0.9
Overall	80.3	76.6	65.2	75.5 (17.4×, 3.2%)	78.6 (13.5×, 4.7%)	82.5 (×7.5, 8.4%)
Category Averages						
Sports (Fixed)	87.5	81.2	71.0	80.8 (24.4×, 1.6%)	82.8 (21.8×, 1.8%)	87.6 (10.4×, 5.1%)
Sports (Moving)	82.2	72.6	59.8	76.0 (20.6×, 2.1%)	79.3 (14.5×, 3.6%)	84.1 (6.0×, 9.1%)
Sports (Ego)	72.3	69.4	55.1	65.0 (13.6×, 3.7%)	70.2 (9.1×, 6.0%)	75.0 (4.9×, 10.4%)
Animals	89.0	83.2	73.4	82.9 (21.7×, 1.9%)	84.3 (19.6×, 2.2%)	87.6 (14.3×, 4.4%)
Traffic	82.3	82.6	74.0	79.1 (11.8×, 4.6%)	82.1 (8.5×, 7.1%)	84.3 (5.4×, 10.1%)
Driving/Walking	50.6	69.3	55.9	59.6 (5.8×, 8.6%)	63.9 (4.9×, 10.5%)	66.6 (4.3×, 11.9%)
Subset of Individual Video Streams						
Table Tennis (P)	89.4	84.8	75.4	81.5 (24.7×, 1.6%)	83.5 (24.1×, 1.6%)	88.3 (12.9×, 3.4%)
Kabaddi (P)	88.2	78.9	66.7	83.8 (24.8×, 1.6%)	84.5 (23.5×, 1.7%)	87.9 (7.8×, 6.3%)
Figure Skating (P)	84.3	54.8	37.9	72.3 (15.9×, 2.8%)	76.0 (11.4×, 4.1%)	83.5 (5.4×, 9.4%)
Drone (P)	74.5	70.5	58.5	70.8 (15.4×, 2.8%)	76.6 (6.9×, 7.2%)	79.9 (4.1×, 12.5%)
Birds (Bi)	92.0	80.0	68.0	85.3 (24.5×, 1.6%)	85.7 (24.2×, 1.6%)	87.9 (21.7×, 1.8%)
Dog (P,D,A)	86.1	80.4	71.1	78.4 (19.0×, 2.2%)	81.2 (13.8×, 3.2%)	86.5 (6.0×, 8.4%)
Ego Dodgeball (P)	82.1	75.5	60.4	74.3 (17.4×, 2.5%)	79.5 (13.2×, 3.4%)	84.2 (6.1×, 8.2%)
Biking (P,Bk)	70.7	71.6	61.3	68.2 (12.7×, 3.5%)	72.3 (6.7×, 7.3%)	75.3 (4.1×, 12.4%)
Samui Street (P,A,Bk)	80.6	83.8	76.5	78.8 (8.8×, 5.5%)	82.6 (5.3×, 9.5%)	83.7 (4.2×, 12.2%)
Driving (P,A,Bk)	51.1	72.2	59.7	63.8 (5.7×, 8.8%)	68.2 (4.5×, 11.5%)	66.7 (4.1×, 12.4%)

Table 2: Comparison of accuracy (mean IoU over all the classes excluding background), runtime speedup relative to MRCNN (where applicable), and the fraction of frames where MRCNN is executed. Classes present in each video are denoted by letters (A - Auto, Bi - Bird, Bk - Bike, D - Dog, E - Elephant, G - Giraffe, H - Horse, P - Person). Overall, online distillation using JITNet provides a better accuracy/efficiency tradeoff than baseline methods.

[Mullapudi, 2018]

Training Workload is Adaptive to Input Video



[Mullapudi, 2018]

Online Training: Concluding Remarks

- Online training of ultra-light model is a promising approach to realize ultra-low power inference. Especially, the networks handing data with temporal locality, e.g., video, speech, etc., can benefit from online training since the ultra-light model can be tailored to the current input though sacrificing generalizability.
- Online training needs to run on low precision NPU on servers (e.g., for recommendations) and mobile devices (e.g., for real-time tasks), which requires low precision training.
- Low-cost online training may enable online neural architecture search (NAS), which can tailor basic building blocks, e.g., NAS cells to the current input thereby further improving inference efficiency.

[S.Yoo]