BlockCopy: High-Resolution Video Processing with Block-Sparse Feature Propagation and Online Policies

Thomas Verelst   Tinne Tuytelaars

ESAT-PSI, KU Leuven

Leuven, Belgium

{thomas.verelst, tinne.tuytelaars}@esat.kuleuven.be

Abstract

In this paper we propose BlockCopy, a scheme that accelerates pretrained frame-based CNNs to process video more efficiently, compared to standard frame-by-frame processing. To this end, a lightweight policy network determines important regions in an image, and operations are applied on selected regions only, using custom block-sparse convolutions. Features of non-selected regions are simply copied from the preceding frame, reducing the number of computations and latency. The execution policy is trained using reinforcement learning in an online fashion without requiring ground truth annotations. Our universal framework is demonstrated on dense prediction tasks such as pedestrian detection, instance segmentation and semantic segmentation, using both state of the art (Center and Scale Predictor, MGAN, SwiftNet) and standard baseline networks (Mask-RCNN, DeepLabV3+). BlockCopy achieves significant FLOPS savings and inference speedup with minimal impact on accuracy.

1. Introduction

Most contemporary convolutional neural networks (CNN) are trained on images and process video frame-by-frame, for simplicity or due to the lack of large annotated video datasets. For instance, the popular COCO dataset [21] for large-scale object detection does not include video sequences. However, video typically contains a considerable amount of redundancy in the temporal domain, with some image regions being almost static. Image-based convolutional neural networks do not take advantage of temporal and spatial redundancies to improve efficiency: they apply the same operations on every pixel and every frame. Representation warping has been proposed to save computations [8, 51, 17], but optical flow is expensive and warping cannot cope with large changes such as newly appearing objects. Other video processing methods, *e.g.* using 3D con-
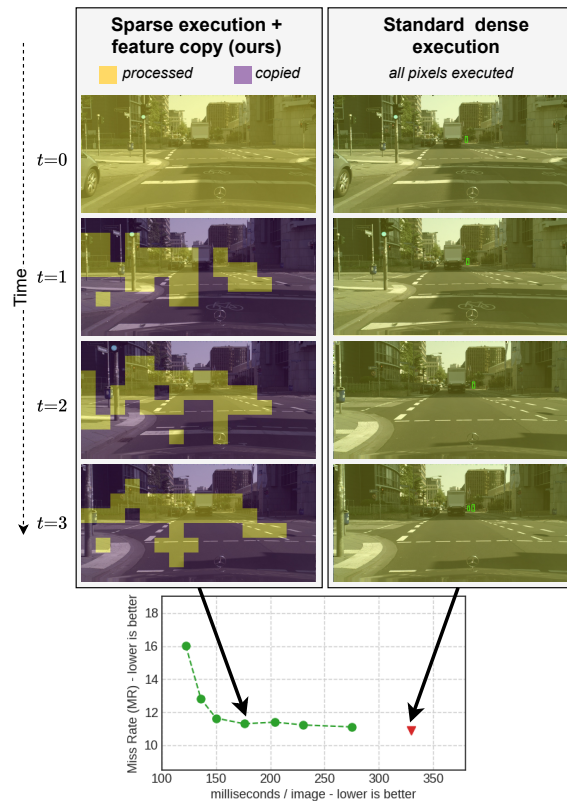

Figure 1: BlockCopy accelerates existing CNNs by sparsely executing convolutions, while copying features from previous executions in non-important regions. In this example on pedestrian detection, inference speed is more than doubled with negligible increase in detection miss rate.

volutions or recurrent neural networks [18, 28, 22], focus on improving accuracy by using temporal information, instead of reducing computations by exploiting redundancies.

In this work, we propose a method to improve the efficiency and inference speed of convolutional neural networks for dense prediction tasks, by combining temporal feature propagation with sparse convolutions as illus-

trated in Figure 1. A lightweight, trainable policy network selects important image regions, and the expensive task network sparsely executes convolutions on selected regions only. Features from non-important regions are simply copied from the previous execution, thereby saving computations.

The policy network is trained with reinforcement learning in an online fashion: the output of the large task network, for example Mask-RCNN [12], serves as supervisory signal to train the online policy. Using online reinforcement learning has several advantages. First, no labeled data is required and off-the-shelf networks can be optimized during deployment without designing a separate training pipeline. Second, online training allows the network to fine-tune the policy to the task and dataset at deployment time. Finally, models of different computational costs can be obtained by simply adjusting the policy's computational target parameter.

The main contributions of this work are as follows:
- We propose BlockCopy to adapt existing CNNs for more efficient video processing, using block-sparse convolutions and temporal feature propagation. Our framework is implemented in PyTorch, using custom CUDA operations.
- We utilize reinforcement learning to train a policy network in an online fashion without requiring ground-truth labels.
- We demonstrate our method on pedestrian detection, instance segmentation and semantic segmentation tasks and show that existing off-the-shelf CNNs can be significantly accelerated without major compromises in accuracy.
- We show that BlockCopy improves the accuracy-speed trade-off by comparison with existing methods, lower resolution and lower frame rate baselines.

The code is available online[1].

2. Related work

Well-known methods to reduce the computational cost of convolutional neural networks are pruning [19], quantization [15] or knowledge distillation [14]. Recently, dynamic methods [43, 46] gained interest, adapting the network's operations based on the image's difficulty. Video processing methods are complementary, as they avoid redundant computations using the temporal dimension.

2.1. Conditional execution and sparse processing

Dynamic neural networks, also known as conditional execution [3, 39], adapt the network complexity based on the image's difficulty. SkipNet [43] and ConvNet-AIG [39] skip residual blocks for easy images, reducing the

average amount of computations for image classification. The policy, determining which blocks to skip, is learned using reinforcement learning [43] or a reparametrization trick [39]. Recently, this has been extended to the spatial domain [7, 47, 40] by skipping individual pixels. However, pixel-wise sparse convolutions are not supported by default in deep learning frameworks such as PyTorch and TensorFlow, and are challenging to implement efficiently on GPU [9]. As a consequence, most work only demonstrates performance improvements on CPU [47], on specific architectures [40], or only consider theoretical advantages [7].

Block-based processing, where pixels are grouped in blocks, are more feasible and have been accelerated on GPU [35, 41]. Simply splitting images in blocks and then processing those individually is not sufficient, as features should propagate between blocks to achieve a large receptive field and avoid boundary artefacts. To this end, SBNet [35] proposes to use partly overlapping blocks and applies this for 3D object detection. SegBlocks [41] introduces a framework with BlockPadding modules to process images in blocks. In this work, we extend conditional execution to the video domain, using a policy trained with reinforcement learning in combination with block-based sparse processing and feature transfer.

2.2. Video processing

Most video processing methods focus on video classification and action recognition applications [6, 38], incorporating multi-frame motion information using methods such as multi-frame non-maximum suppression [10], feature fusion [26], 3D convolutions [18, 28], recurrent neural networks [22] or other custom operations [42].

Efficiency and speed can be improved by exploiting temporal redundancies, as changes between frames are often small. Clockwork Nets [36] proposed a method to adaptively execute network stages based on semantic stability. Deep Feature Flow [51], NetWarp [8], GRFP [30], Awan and Shin [2], and Paul *et al*. [34] warp representations between frames using optical flow, with Accel [17] introducing a lightweight second branch to fine tune representations. DVSNet [48] proposes an adaptive keyframe scheduler, selecting the key frames to execute while other frames use warped representations. Awan and Shin [1] use reinforcement learning to train the keyframe selection scheme. However, optical flow is an expensive calculation and therefore these methods mainly focus on large networks such as DeepLabV3+ [4], where the introduced overhead is modest compared to the segmentation network. Jain and Gonzalez [16] use block motion vectors, already present in compressed video. Low-Latency Video Semantic Segmentation (LLVSS) [20] does not use optical flow, but updates keyframe representations using a lightweight per-frame update branch. Mullapudi *et al*. [29] demonstrate online
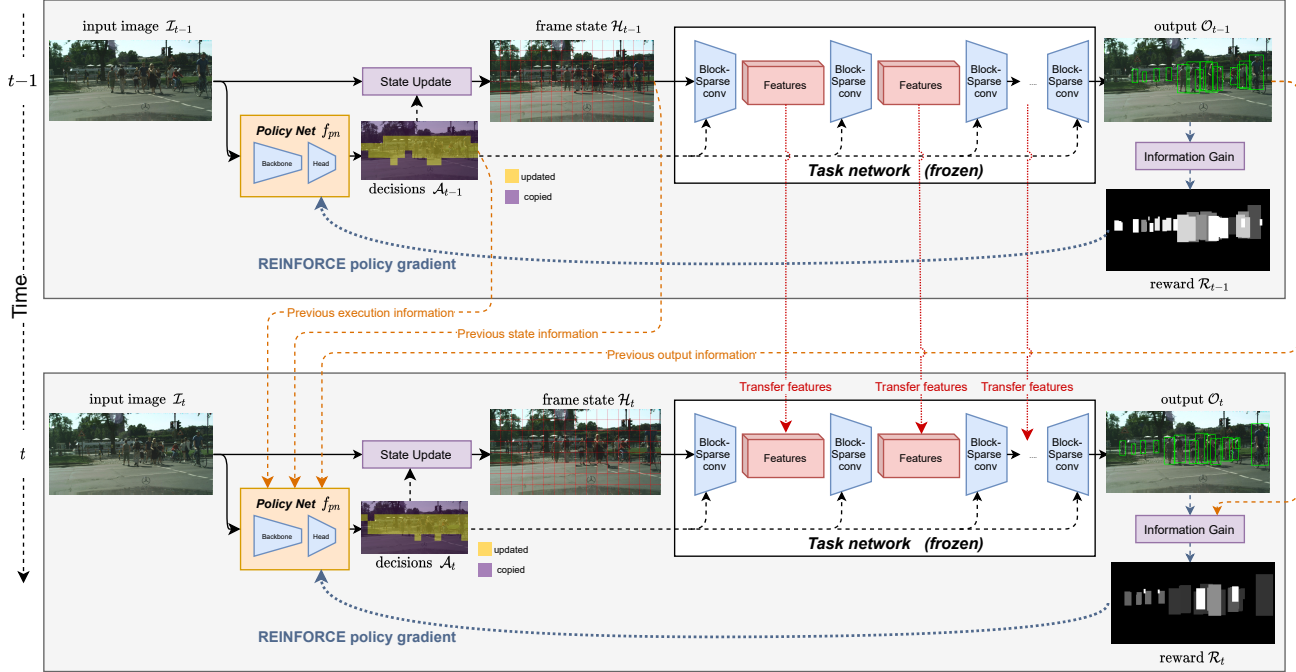

Figure 2: Overview of the BlockCopy pipeline, illustrated for two video frames. The policy network outputs execution decisions. The network is executed using block-sparse convolutions and features from the previous iteration are copied to non-executed regions. The importance of each block is measured using Information Gain, which serves as a reward to update the policy weights.

knowledge distillation for video object segmentation, where a lightweight student network is finetuned for a given situation in the video by learning from a large teacher network. Our method does not require keyframes, and only the first frame of each clip is executed completely. All other frames are executed sparsely, resulting in more consistent processing delays in comparison with keyframe-based methods.

3. BlockCopy method and policy network

BlockCopy optimizes a large task network for more efficient video processing, by combining block-sparse convolutions with feature transfers. Our method consists of two main components: a framework to efficiently execute a CNN architectures in a block-sparse fashion using temporal feature propagation, and a policy network determining whether blocks should be executed or transferred.

The policy network is a lightweight, trainable convolutional network, selecting the blocks to be executed. As the decision is binary for each region, *execute* or *transfer*, standard backpropagation cannot be used to train the policy network. Therefore, we use reinforcement learning, based on a reward per block, to train the policy network in an online self-distilled fashion based on the task network's output. The reward function is based on the *information gain*, representing the amount of task information gained by exe-

cuting the region instead of just transferring features. Note that the task network's weights are not updated, and only the policy network is trained. Figure 2 presents an overview of the components discussed in the next subsections.

3.1. Block-sparse processing with feature transfer

Standard libraries for deep learning such as PyTorch [33] do not support efficient sparse convolutions. We build on the framework introduced by SegBlocks [41] to process images in blocks, by first splitting images into blocks and applying their BlockPadding module avoiding discontinuities between blocks. At execution time, representations throughout the network are stored and copied with efficient and specialized CUDA modules.

3.2. Online policy with reinforcement learning

The policy network is trained to select important regions that have high impact on the output. Using ground-truth annotations of video sequences, one could extract the regions where the output changes. However, many computer vision datasets do not contain video sequences and annotating ground-truth is expensive. Instead of using ground-truth annotations, we opt for a more flexible approach with self-distillation and online reinforcement learning.

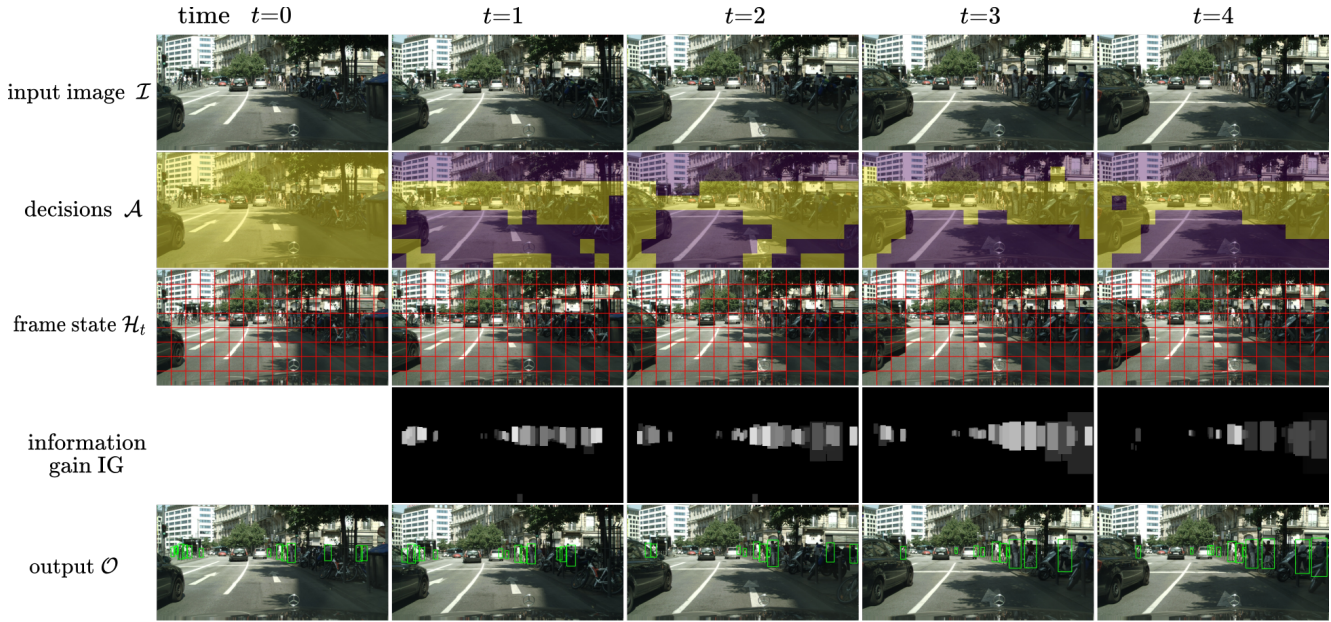When a block is executed, the importance of this execu-

Figure 3: Illustration of a video sequence, with execution grids, frame states and outputs. The frame state is only updated for selected regions (yellow), whereas features from other regions are re-used from the previous frame (purple). The output bounding boxes are visualized for detections with scores larger than 0.5, whereas the reward scales with the detection score. A video with visualizations can be found in supplemental material.
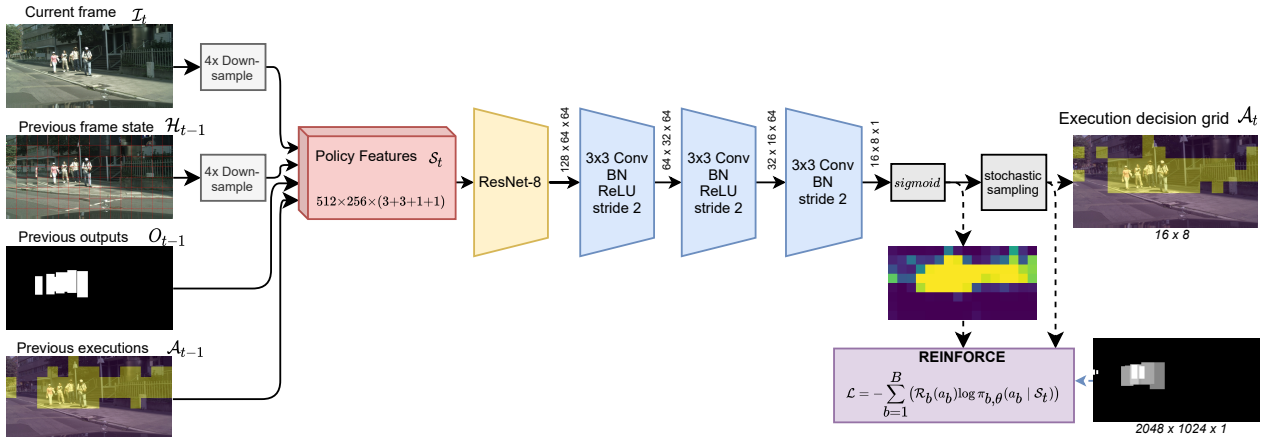

Figure 4: Policy network architecture. Dimensions are given as $W \times H \times C$, with input images of $2048 \times 1024 \times 3$ pixels.

tion is determined using the *information gain*. Blocks where large changes in the output occur have a large information gain. This way, the policy network can learn the relative importance of blocks at execution time, without requiring a separate training pipeline, expensive teacher network or annotated data.

3.3. Policy network architecture

The policy network uses a lightweight 8-layer ResNet backbone combined with a fully convolutional head, with the architecture depicted in Figure 4. The backbone operates on a feature representation $\mathcal{S}_t$ consisting of 4 inputs:

- Current frame I_t: the RGB frame at time t.
- Previous frame state $\mathcal{H}_{t-1}$: the previous frame state is an RGB frame, where each block has the image content of the last executed block for that position. By using the previous state, instead of simply the previous frame, we ensure that the network can detect small accumulating changes.
- Previous output $\mathcal{O}_{t-1}$: We represent detections or instances using a probability mask for each individual class. For segmentation, the output probabilities per pixel are used.
- Previous execution grid $\mathcal{A}_{t-1}$: The previous execution

grid is a binary mask indicating which blocks were executed for the previous frame. In combination with the previous output, this improves the exploration ability of the policy, as previously executed blocks with no information gain are less likely to contain new information.

3.4. Information Gain

To determine the importance of each region, we define the Information Gain (IG) for each output pixel, as a quantity representing the amount of additional information gained by executing the model for that pixel compared to using the previous output. The information gain IG_t at time t is a function of the output $\mathcal{O}_t$ and the previous output $\mathcal{O}_{t-1}$.

The formulation of information gain is task-dependent, exploiting a task's characteristics to minimize the number of required computations while maximizing the output accuracy. The information gain is determined per pixel p, and combined afterwards per block b using max-pooling:

$$IG_b = \max IG_p \quad \forall p \in b . \tag{1}$$

Object detection For object detection tasks, the information gain depends on the movement of objects and the score of the prediction. For every new frame, predicted bounding boxes are matched with the previous detections, by choosing the most overlapping detection using the intersection-over-union IoU_{bb} of the bounding boxes. High overlap means low information gain, with static objects having no information gain. If an object does not overlap with any object detected in the previous frame, the object is a new detection and has an information gain equal to the detection score. Objects in the previous frame not matched with any object in the current frame also have information gain equal to the score of the previous detection, as those detections should be removed.

Algorithm 1 is used to determine the information gain. Note that it is important to also assign information gain to pixels of the detections in the previous frame, in order to update or remove those detections when needed. Figure 3 contains visualizations for the information gain.

Instance segmentation The definition of information gain for instance segmentation is similar to the one of object detection, but the Intersection-over-Union is determined by the instance masks instead of the bounding boxes.

Semantic segmentation Semantic segmentation is a dense pixelwise classification task, where the network outputs a probability distribution per pixel. The information gain of each output pixel is determined by the pixelwise KL-Divergence between the output probability distributions.

Algorithm 1 Information Gain for Object Detection

---

require: outputs $\mathcal{O}_t$, previous outputs $\mathcal{O}_{t-1}$
\# Initialize information gain as zero-filled matrix of size $H \times W$
$IG \leftarrow 0^{H \times W}$
for all detections det **in** $\mathcal{O}_t$ **do**
  \# find most overlapping detection of previous output
  $IoU_{best} \leftarrow 0$
  $prevDet_{best} \leftarrow \text{NULL}$
  for all detections $prevDet$ **in** $\mathcal{O}_{t-1}$ **do**
    if $\text{IoU}(det, prevDet) > \text{IoU}_{best}$ **then**
      $IoU_{best} \leftarrow \text{IoU}(det, prevDet)$
      $prevDet_{best} \leftarrow prevDet$
  \# set IG for pixels in bounding box of detection
  for all pixels $p \in det$ **do**
    $IG_p \leftarrow \max(IG_p, (1 - IoU_{best}) \cdot det_{score})$
  \# set IG for pixels in bounding box of matched detection
  for all pixels $p \in prevDet_{best}$ **do**
    $IG_p \leftarrow \max(IG_p, (1 - IoU_{best}) \cdot prevDet_{score})$
\# process previous detections not overlapping with current ones
for all detections $prevDet$ **in** $\mathcal{O}_{t-1}$ **do**
  if $prevDet$ not processed **then**
    $IG_p \leftarrow prevDet_{score} \quad \forall$ pixels $p \in prevDet$
return IG

3.5. Reinforcement learning

The policy network f_{pn} with parameters θ outputs a probability p_b for each block b, indicating whether the features in that block should be calculated instead of just transferred. The network operates on the feature representation

$$\mathcal{S}_t = \{I_t, \mathcal{H}_{t-1}, \mathcal{A}_{t-1}, \mathcal{O}_{t-1}\} \tag{2}$$

and outputs execution probabilities for each block b:

$$\mathcal{P}_t = f_{pn}(\mathcal{S}_t; \theta) \tag{3}$$
$$\text{with } \mathcal{P}_t = [p_1, \ldots, p_b, \ldots, p_B] \in [0, 1]^B. \tag{4}$$

Probabilities $\mathcal{P}_t$ are sampled to execution decisions $\mathcal{A}_t = [a_1, \ldots, a_b, \ldots, a_B] \in \{0, 1\}^B$. The policy $\pi_{b,\theta}(a_b \mid \mathcal{S}_t)$ gives the probability of action a_b. Execution decision $a_b = 1$ results in execution of block b and $a_b = 0$ results in feature transfer from the previous execution.

Stochastic sampling according to the probabilities encourages search space exploration, in comparison to simple thresholding. As gradients cannot be backpropagated through the sampling operation, we adopt reinforcement learning in order to optimize the policy for the task at hand.

Actions should maximize the reward for each block, with the objective to be maximized at each time step given by

$$\max \mathcal{J}(\theta) = \max \sum_{b=1}^{B} \left(\mathbb{E}_{a_b \sim \pi_{b,\theta}} \left[\mathcal{R}_b(a_b) \right] \right) \tag{5}$$

where $\mathcal{R}_b$ is the reward based on the Information Gain IG as described later. The reward, loss, objective and

parameters are determined at every timestep t, which we omit for simplicity of notation. The policy network's parameters θ can then be updated using gradient ascent with learning rate α:

$$\theta \leftarrow \theta + \alpha \nabla_\theta [\mathcal{J}(\theta)] . \tag{6}$$

Based on REINFORCE policy gradients [44], we can derive the loss function as (see supplemental material)

$$\mathcal{L} = -\sum_{b=1}^{B} \big(\mathcal{R}_b(a_b) \log \pi_{b,\theta}(a_b \mid \mathcal{S}_t)\big) . \tag{7}$$

The reward $\mathcal{R}_b$ depends on the information gain of that block. A trivial state would be to execute all blocks. Therefore, we introduce a reward $\mathcal{R}_{cost}$ weighted by hyperparameter γ to balance the number of computations:

$$\mathcal{R}_b(a_b) = \mathcal{R}_{IG}(a_b) + \gamma \mathcal{R}_{cost}(a_b) . \tag{8}$$

Executed blocks have a positive reward for positive information gain. In contrast, non-executed blocks have a negative reward to increase the likelihood of being executed:

$$\mathcal{R}_{IG}(a_b) = \begin{cases} IG_b & \text{if } a_b = 1 , \\ -IG_b & \text{if } a_b = 0 . \end{cases} \tag{9}$$

with IG_b the information gain in a block.

The cost of a frame is the percentage of executed blocks:

$$\mathcal{C}_t = \frac{\sum_i^B a_i}{B} \in [0,1] . \tag{10}$$

As some frames might require more executed blocks than others, we define a moving average with momentum μ:

$$\mathcal{M}_t = (1 - \mu) \cdot \mathcal{C}_t + \mu \cdot \mathcal{C}_{t-1} . \tag{11}$$

Instead of simply minimizing the cost, we use a target parameter $\tau \in [0,1]$, which defines the desired average cost. This results in more stable training with less dependence on the exact value of γ. The cost reward is then given by

$$\mathcal{R}_{cost}(a_b) = \begin{cases} \tau - \mathcal{M}_t & \text{if } a_b = 1 , \\ -(\tau - \mathcal{M}_t) & \text{if } a_b = 0 . \end{cases} \tag{12}$$

Executed blocks, where $a_b{=}1$, have a positive reward when the number of computations is lower than target τ. The target could be adjusted at execution time, changing the model complexity on-the-fly.

4. Experiments

The acceleration achieved by BlockCopy is evaluated on pedestrian detection, instance segmentation and semantic segmentation tasks using datasets containing high-resolution video sequences. Pedestrian detection is a single-class object detection problem. Our method is particularly suited for this task, as small persons need to be detected in high-resolution images. For each task, we integrate BlockCopy in state of the art existing networks, using publicly available implementations. Implementation details are given in the respective subsections. In fully convolutional single-stage architectures, e.g. CSP [25] and Swift-Net [31], we integrate our block-sparse convolution with temporal feature propagation in all convolutional layers. For two-stage architectures, such as Mask-RCNN [12] and MGAN [32], only the backbone and region proposal network are optimized, with the network head applied as normal.

Datasets The Cityscapes [5] dataset is used to evaluate instance and semantic segmentation. The dataset consists of 2975, 500 and 1525 video sequences for training, validation and testing respectively. Each video sequence contains 20 frames of $2048{\times}1024$ pixels recorded at 17 Hz, with the last frame having detailed semantic and instance annotations. We use the standard 19 classes for semantic segmentation and 8 classes for instance segmentation. CityPersons [49] builds on Cityscapes by adding high-quality bounding box annotations for 35000 persons.

Note that we do not use ground-truth annotations to train our method, as the policy network is trained in a self-distilled fashion. Ground-truth labels are only used to evaluate accuracy after speeding up inference.

Evaluation Metrics The accuracy of pedestrian detection is evaluated using the log-average miss rate (MR) criterion, following the standard metrics (Reasonable, Bare, Partial, Heavy) of CityPersons [49]. Instance segmentation is evaluated using COCO-style mean average precision (AP) and AP50 for an overlap of 50%, while semantic segmentation uses the mean Intersection-over-Union (mIoU) metric.

Computational cost is measured by two metrics: the number of operations and the inference speed. The number of operations is reported as billions of multiply-accumulates (GMACS). Inference speed is measured as the average processing time per frame, including data loading and postprocessing, on an Nvidia GTX 1080 Ti 11 GB GPU with an Intel i7 CPU, PyTorch 1.7 and CUDA 11.

BlockCopy configuration For all tasks, we train the policy network in an online fashion using an RMS optimizer with learning rate $1e^{-4}$ and weight decay $1e^{-3}$. To mitigate the impact of the backward pass and weight updates, we only update the policy weights every 4 frames. Before running on the validation or test set, the policy is initialized on 400 training clips. The first frame of a video sequence is always executed completely, and the following 19 frames are processed sparsely using the BlockCopy framework and policy. We evaluate each network and task with varying cost targets $\tau \in [0,1]$ in order to obtain models with different computational complexities. Hyperparameter γ setting

Table 1: Results (log-average Miss Rate) on CityPersons val/test set. Values marked with † were determined using the Pedestron [11] implementation. Inference time per frame measured on a GTX 1080 Ti GPU.

Method	Backbone	Reasonable [test]	Reasonable	Bare	Partial	Heavy	GMACs	avg. sec/img
CSP + BlockCopy (τ=0.3)	ResNet-50	12.50	11.4	7.6	10.8	49.5	393	0.151 s
MGAN + BlockCopy (τ=0.3)	VGG-16	10.83	11.2	6.3	10.9	60.5	560	0.140 s
CSP (CVPR2019) [25]	ResNet-50	-	11.0	7.3	10.4	49.3	1128 †	0.330 s
MGAN (ICCV2019) [32]	VGG-16	-	11.0	-	-	50.3	1104 †	0.224 s †
MGAN scale ×1.3 [32]	VGG-16	9.29	9.9	-	-	45.4	1665 †	0.370 s †
ALFNet (ECCV2018) [24]	ResNet-50	-	12.0	8.4	11.4	51.9	-	0.270 s
AdaptiveNMS (CVPR2019) [23]	VGG-16	11.17	10.8	6.2	11.4	54.0	-	-
OR-CNN (ECCV2018) [50]	VGG-16	11.32	11.0	5.9	13.7	51.3	-	-
HBAN (Neurocomputing2020) [27]	VGG-16	11.26	10.7	-	-	46.9	-	0.760 s

the balance between the reward terms is fixed to 5, with the cost momentum μ set to 0.9.

Baselines Besides comparisons with other methods, we compare our inference speedup method with lower spatial resolution and lower frame rate baselines. Reducing the input resolution decreases the number of operations and increases inference speed, with worse predictions for small objects. Decreasing the frame rate by skipping frames decreases temporal resolution. Our experiments show that lowering the frame rate has a significant impact on accuracy, underlining the important of fast processing.

4.1. Pedestrian detection

We integrate our method in the Center and Scale Predictor (CSP) [25] and Mask-Guided Attention Network (MGAN) [32] architectures. CSP is a single-stage anchor-free detector, predicting the center and scale of each object with the aspect ratio fixed to 0.41. It builds on the ResNet-50 [13] backbone. MGAN is a dual-stage detector using the VGG [37] backbone. Our implementation is based on the Pedestron [11] framework. The standard evaluation setting uses 17 Hz video sequences at 2048×1024 resolution on the Citypersons dataset [49].

The detection results for CSP [25] with BlockCopy are shown in Figure 5a and Figure 5b, when comparing the number of operations (GMACS) and inference time respectively. BlockCopy models (with different target costs τ) achieve better detection results (lower miss rate) than lower resolution and lower frame rate baselines, demonstrating improved efficiency. With τ=0.3, the amount of operations and processing time is more than halved with only 0.4% increase in miss rate. Table 1 compares BlockCopy to other methods and demonstrates that our method is faster than existing methods while achieving competitive accuracy.

4.2. Instance segmentation

We integrate BlockCopy in the popular Mask-RCNN [12] architecture with ResNet-50 [13] backbone

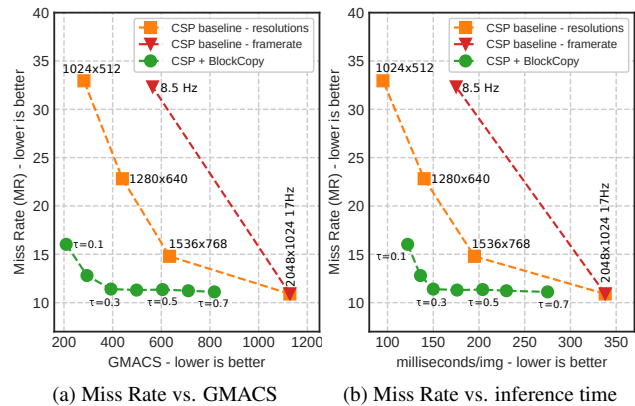(a) Miss Rate vs. GMACS   (b) Miss Rate vs. inference time

Figure 5: Results on CityPersons validation set on CSP [25] with BlockCopy. Miss Rate (MR) is reported for the Reasonable subset. Models with BlockCopy consistently outperform lower resolution baselines of similar complexity.

for instance segmentation, using the baseline provided by Detectron2 [45]. Figure 6a and 6b show that integrating BlockCopy with τ=0.3 halves the amount floating point operations with 0.9% accuracy decrease, while the frames processed per second increases from 6.7 to 11.0 FPS. The test set submission of our method (with τ=0.5) achieved 31.7 AP, compared to 32.0 AP of the Mask-RCNN baseline [12], with ×1.65 faster inference.

4.3. Semantic segmentation

We compare BlockCopy with other video processing using optical flow and adaptive keyframe scheduling on a semantic segmentation task. Our method can be seen as a sparse version of adaptive scheduling, where each block is scheduled individually.

BlockCopy is integrated in the popular DeepLabV3+ [4] network combined with a ResNet-101 [13] backbone, and the faster SwiftNet [31] model with ResNet-50.

Since other methods report inference time on various GPUs, we scaled those values to be equivalent to a Nvidia

Table 3: Overhead of BlockCopy components in the CSP network for pedestrian detection [25].

Method	Total	Task network			Policy			GMACS Task	GMACS Policy	Acc. (MR)
		Sparse Conv. Overhead	Feature Transfer	Ops	Policy Network	Inform. Gain	Backward pass + Weight update			
CSP baseline	330 ms	N.A.	N.A.	330	N.A.	N.A.	N.A.	1128	N.A.	11.0 %
CSP + BlockCopy ($\tau = 0.7$)	275 ms	30 ms	9 ms	215 ms	9 ms	3 ms	9 ms	812 (-28%)	6.5	11.1 %
CSP + BlockCopy ($\tau = 0.5$)	204 ms	25 ms	12 ms	146 ms	9 ms	4 ms	8 ms	599 (-47%)	6.5	11.3 %
CSP + BlockCopy ($\tau = 0.3$)	150 ms	21 ms	15 ms	92 ms	9 ms	3 ms	10 ms	388 (-65%)	6.5	11.4 %

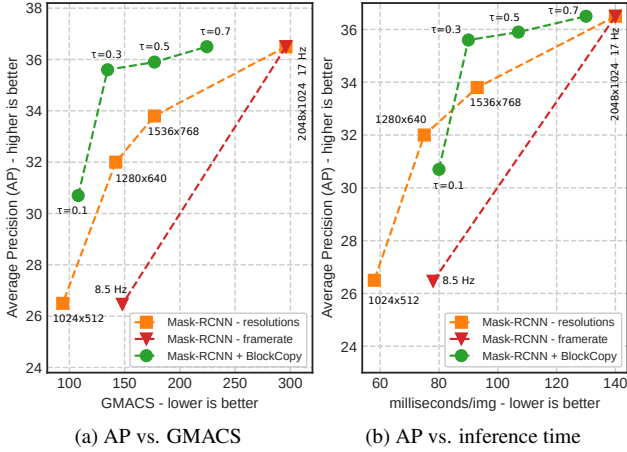(a) AP vs. GMACS (b) AP vs. inference time

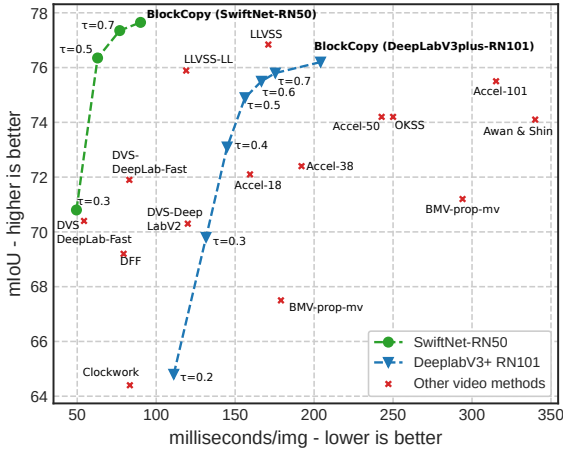Figure 6: Cityscapes instance segmentation *val* set.

Figure 7: Results on Cityscapes semantic segmentation validation set. Inference time of other methods is compensated for GPU performance to match GTX 1080 Ti (see supplemental). Other video methods are Accel [17], Awan and Shin [2], BMV-prop-mv [16], Clockwork [36], DFF [51], DVSNet [48], LLVSS [20] and OKSS [1].

GTX 1080 Ti, as proposed by other works for fair comparison [31]. All data, including non-compensated inference time and the GPU scaling factors, can be found in supplemental. Figure 7 shows that our method is competitive

Table 2: Ablation for policy network.

Backbone	Online	$\mathcal{I}_t$	$\mathcal{I}_{t-1}$	$\mathcal{F}_{t-1}$	$\mathcal{O}_{t-1}$	$\mathcal{A}_{t-1}$	GMACS	MR
ResNet-8	✓	✓					6.4	13.0 %
	✓	✓	✓				6.4	12.0 %
	✓	✓		✓			6.4	12.0 %
	✓	✓		✓	✓		6.5	11.7 %
	✓	✓	✓	✓		✓	6.5	11.5 %
	✓	✓		✓	✓	✓	6.5	11.4 %
		✓		✓	✓	✓	6.5	13.3 %
ResNet-20	✓	✓		✓	✓	✓	34.1	11.5 %

with methods designed specifically for semantic segmentation and achieves higher mIoU with lower inference time.

5. Ablation

Ablation for the policy network is given in Table 2 and shows that including more information about the previous frame is beneficial. Online learning slightly outperforms offline learning. The overhead introduced by BlockCopy is given in Table 3. The execution of the policy network and the updates of the weights based on information gain are relatively cheap compared to the task network.

6. Conclusion

We proposed the BlockCopy framework that can be applied to existing pre-trained convolutional neural networks, improving their efficiency for high-resolution video-based processing. We integrated the method in a variety of networks for different computer vision tasks, demonstrating strong inference speedup with only a small drop in accuracy. Tasks such as pedestrian detection and instance segmentation are particularly suited for this method, as only few image areas are important. By not requiring training labels, our method can be integrated in deployment pipelines starting from existing pre-trained models.

Acknowledgement

References

[1] Mehwish Awan and Jitae Shin. Online keyframe selection scheme for semantic video segmentation. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–5. IEEE, 2020. 2, 8

[2] Mehwish Awan and Jitae Shin. Semantic video segmentation with dense-and-dual warping spatial features. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 129–132. IEEE, 2020. 2, 8

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432 [cs]*, Aug. 2013. 2

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2, 7

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[6] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. Temporal 3d convnets: New architecture and transfer learning for video classification. *arXiv preprint arXiv:1711.08200*, 2017. 2

[7] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2017. 2

[8] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. Semantic video cnns through representation warping. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4453–4462, 2017. 1, 2

[9] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. *arXiv preprint arXiv:2006.10901*, 2020. 2

[10] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video pbject detection. *arXiv preprint arXiv:1602.08465*, 2016. 2

[11] Irtiza Hasan, Shengcai Liao, Jinpeng Li, Saad Ullah Akram, and Ling Shao. Generalizable pedestrian detection: The elephant in the room, 2020. 7

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 2, 6, 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015. 2

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017. 2

[16] Samvit Jain and Joseph E Gonzalez. Fast semantic segmentation on video using block motion-based feature interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 2, 8

[17] Samvit Jain, Xin Wang, and Joseph E Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8866–8875, 2019. 1, 2, 8

[18] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2012. 1, 2

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2

[20] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5997–6005, 2018. 2, 8

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[22] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko. Looking fast and slow: Memory-guided mobile video object detection. *arXiv preprint arXiv:1903.10172*, 2019. 1, 2

[23] Songtao Liu, Di Huang, and Yunhong Wang. Adaptive nms: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6459–6468, 2019. 7

[24] Wei Liu, Shengcai Liao, Weidong Hu, Xuezhi Liang, and Xiao Chen. Learning efficient single-stage pedestrian detectors by asymptotic localization fitting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 618–634, 2018. 7

[25] Wei Liu, Shengcai Liao, Weiqiang Ren, Weidong Hu, and Yinan Yu. High-level semantic feature detection: A new perspective for pedestrian detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5187–5196, 2019. 6, 7, 8

[26] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. Efficient semantic video segmentation with per-frame inference. In *European Conference on Computer Vision*, pages 352–368. Springer, 2020. 2

[27] Ruiqi Lu, Huimin Ma, and Yu Wang. Semantic head enhanced pedestrian detection in a crowd. *Neurocomputing*, 400:343–351, 2020. 7

[28] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 1, 2

[29] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3573–3582, 2019. 2

[30] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6819–6828, 2018. 2

[31] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In Defense of Pre-Trained ImageNet Architectures for Real-Time Semantic Segmentation of Road-Driving Images. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12599–12608, Long Beach, CA, USA, June 2019. IEEE. 6, 7, 8

[32] Yanwei Pang, Jin Xie, Muhammad Haris Khan, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Ling Shao. Mask-guided attention network for occluded pedestrian detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4967–4975, 2019. 6, 7

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 3

[34] Matthieu Paul, Christoph Mayer, Luc Van Gool, and Radu Timofte. Efficient video semantic segmentation with labels propagation and refinement. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2873–2882, 2020. 2

[35] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. SBNet: Sparse Blocks Network for Fast Inference. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8711–8720, Salt Lake City, UT, June 2018. IEEE. 2

[36] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016. 2, 8

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7

[38] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 2

[39] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018. 2

[40] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2320–2329, 2020. 2

[41] Thomas Verelst and Tinne Tuytelaars. Segblocks: Towards block-based adaptive resolution networks for fast segmentation. In *European Conference on Computer Vision*, pages 18–22. Springer, 2020. 2, 3

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 2

[43] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. SkipNet: Learning Dynamic Routing in Convolutional Networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Proc. European Conf. on Computer Vision (ECCV)*, volume 11217, pages 420–436. Springer Int. Publishing, Cham, 2018. 2

[44] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 6

[45] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 7

[46] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018. 2

[47] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *European Conference on Computer Vision*, pages 531–548. Springer, 2020. 2

[48] Yu-Syuan Xu, Tsu-Jui Fu, Hsuan-Kung Yang, and Chun-Yi Lee. Dynamic video segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6556–6565, 2018. 2, 8

[49] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *CVPR*, 2017. 6, 7

[50] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Occlusion-aware r-cnn: detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 637–653, 2018. 7

[51] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2349–2358, 2017. 1, 2, 8