Patchwork: A Patch-wise Attention Network for Efficient Object Detection and Segmentation in Video Streams

Yuning Chai[*]
Google Inc.
chaiy@google.com

Abstract

Recent advances in single-frame object detection and segmentation techniques have motivated a wide range of works to extend these methods to process video streams. In this paper, we explore the idea of hard attention aimed for latency-sensitive applications. Instead of reasoning about every frame separately, our method selects and only processes a small sub-window of the frame. Our technique then makes predictions for the full frame based on the sub-windows from previous frames and the update from the current sub-window. The latency reduction by this hard attention mechanism comes at the cost of degraded accuracy. We made two contributions to address this. First, we propose a specialized memory cell that recovers lost context when processing sub-windows. Secondly, we adopt a Q-learning-based policy training strategy that enables our approach to intelligently select the sub-windows such that the staleness in the memory hurts the performance the least. Our experiments suggest that our approach reduces the latency by approximately four times without significantly sacrificing the accuracy on the ImageNet VID video object detection dataset and the DAVIS video object segmentation dataset. We further demonstrate that we can reinvest the saved computation into other parts of the network, and thus resulting in an accuracy increase at a comparable computational cost as the original system and beating other recently proposed state-of-the-art methods in the low latency range.

1. Introduction

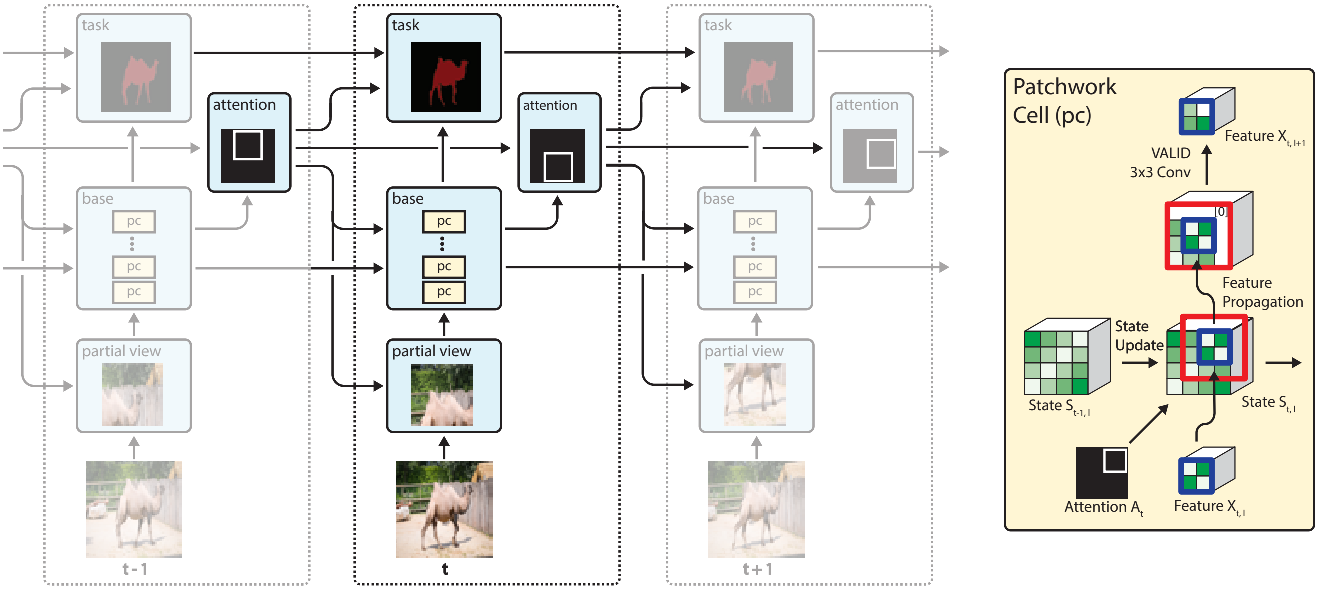
The human visual system is confronted with an overwhelming amount of information in a constant visual stream. Fortunately, our brain is capable enough to remember our environment from previous experiences and can instantly find the best sequence of visual attention decisions to achieve a seemingly effortless perception of our visual surroundings. As [4, 26] pointed out, we do not perceive our entire environment at once, and the "secret" to our success lies in a clever interplay between our ability to *remember* the bigger picture while *focusing* our attention to the objects of interests.

In recent years, there has been tremendous progress in object detection [9, 11, 25, 31] and segmentation [5, 6, 14] techniques thanks to the emergence of deep convolutional networks [15, 21, 35, 38]. It is natural to extend these powerful methods to applications that require continuous stream processing. However, our own human visual perception experience suggests that it is not efficient to naively apply single-frame detectors to every frame of a video at a fixed time interval without any temporal context.

Inspired by the human visual attention system [26], we introduce Patchwork, a model that explores the subtle interplay between memory and attention for efficient video stream processing. Fig. 1a illustrates an overview of Patchwork. At each time step, Patchwork crops a small window from the input frame and feeds it into the feature extractor network, which has been modified to contain a set of specialized memory cells scattered throughout the body of the network. The network eventually predicts task-specific outputs: a set of boxes for object detection or a mask for segmentation. Besides, the network predicts the attention window for the next frame that is most likely to contain useful information for the task.

The primary motivation for Patchwork is efficient stream processing. In other words, we aim to achieve the highest possible detection or segmentation quality while reducing latency and computational cost. For applications where there is no need to reduce the latency, we can re-invest the saved resources and boost quality. We demonstrate this latency reduction and quality improvement on two benchmark datasets: ImageNet VID [32] for video object detection, and DAVIS [29] for video object segmentation. This latency reduction is controlled a priori by a pair of hyperparameters. We explain a few hyperparameter choices in the experimental section. Some choices significantly reduce latency but take a hit in quality, and others save resources but achieve similar quality. There are also configurations where

[*] Now at Waymo LLC.



(a) Overview of the Patchwork architecture.

(b) The Patchwork Cell.

Figure 1: a) The proposed Patchwork architecture. At each time step in the video stream, our method only processes a small sub-window of the frame, but can still reasons about the full frame thanks to a series of stateful Patchwork Cells (pc). b) A zoomed-in view of the stateful Patchwork Cell, which modulates the input feature by encasing it with contextual features from previous states. t denotes the time step while l denotes a particular deep net layer. Please see Sec. 3.2 for details. These cells replace all traditional convolutional filters in a network, as shown in Supplementary Fig. 9.

a quality gain can be observed at a comparable amount of computation.

The contributions of this paper are three-fold: 1) We present Patchwork, a recurrent architecture inspired by the human visual perception system to perform efficient video stream processing. Our method makes use of 2) The Patchwork Cell that serves as the memory unit to carry environmental information across time and 3) An attention model that can predict the best location to attend to in the next frame. Our method is trained via Q-learning with novel object detection and segmentation reward functions.

2. Related work

Efficient stream processing. Driven by application needs, there is increasingly more interest in running deep learning models fast and in real-time. These advances fall into two buckets. The first bucket contains methods that change the network at its rudimentary level, e.g., quantization [17, 43], and layer decomposition [8, 22, 33]. The other bucket of methods, where this work also falls into, operates at a higher and algorithmic level. For image object detection, there are SSD [25] and YOLO [30], both of which are one-stage detectors aimed to achieve a better speed-accuracy trade-off over the two-stage approaches such as Faster-RCNN [31]. Specifically for video streams, under the assumption that the outputs for neighboring frames are often similar, [23, 34, 45] spend heavy computation only on dynamically select key frames, thus trading accuracy for computation. Our work also exploits the temporal consistency, but instead of only selecting key frames, we select spatial locations per frame in the hope of achieving a better latency-accuracy trade-off.

Recurrent attention models. Most attention models so far have been designed to repeatedly look at parts of the same image rather than across different frames in a video. One of the earliest works is the recurrent attention model (RAM) [27], which repeatedly places attention windows onto an MNIST image to classify digits. Several works extended the RAM framework to multi-label classification [1], image generation [12] and single-image object detection [3]. The object detector in [3] gradually moves the attention window around the image until an object is detected, which is vastly different than our approach. More importantly, its complexity grows linearly with the number of objects, where our method has a constant processing time per frame. In the video domain, [13, 19] applied the attention model to video streams, though they do not address the problem of lost context by the spatial attention, which is a key contribution of this work.

3. Patchwork

As shown in Fig. 1a, the Patchwork architecture is a recurrent system where the prediction for the current frame may depend on all previous frames. At each time step,

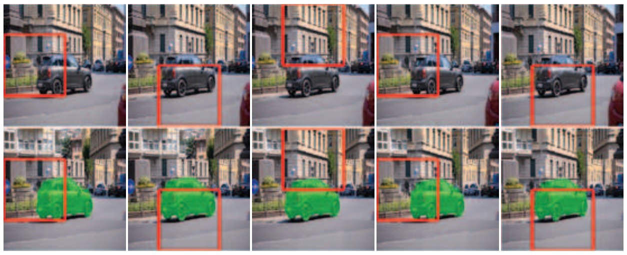


Figure 2: (Best viewed in color) The evolution of a Patchwork Cell memory over five time steps. **Top**: Input frames with attention windows in red rectangles. **Bottom**: The Patchwork Cell aggregates raw input patches cropped by the attention windows over time. Note that only features in the red rectangles are updated. Therefore, the feature map looks in disarray, forming a patchwork pattern (*https://en.wikipedia.org/wiki/Patchwork*).

the input frame undergoes four stages: cropping, feature extraction, task-specific prediction (detection or segmentation) and attention prediction. During cropping, a window of fixed size is cropped from the input frame, where the attention predictor from the previous frame dictates the location of the crop. Our choice to limit the window size to be constant is deliberate so that it allows us to control the computational cost ahead of time, as the cost is roughly proportional to the area of the window. For the feature extraction stage, we use a stateful network adapted from a standard backbone network, MobileNetV2 [33]. We replace **all** its convolutional layers which have kernel size larger than 1x1 with a custom stateful Patchwork Cell, which appears in details in Sec. 3.2. Finally, the attention and task-specific predictors build on an appropriate layer on top of the feature extraction. Please see Supplementary Sec. A for details.

We organize the rest of this section as follows: The two critical pieces of the model, the recurrent attention module and the Patchwork Cell, are described in Sec. 3.1 and Sec. 3.2. The last two sub-sections describe the model training: Sec. 3.3 describes the reward function for both the object detection and segmentation task, and the rest of the training details are concluded in Sec. 3.4.

Patchwork has two meanings. For one, it is a portmanteau for a PATCH-wise attention netWORK. However, Patchwork is also the English word for a form of needlework where multiple pieces of fabric are sewn together into a larger design—this resembles how the memory within each Patchwork cell appears during inference (see Fig. 2).

3.1. Recurrent attention

Fig. 1a shows an overview of the recurrent attention network. In prior literature, the attention window ("attention" in the figure) is parameterized using its center and size in a continuous space [1, 12, 27]. As for the training, [12] used a smooth attention window with gradients on the boundary

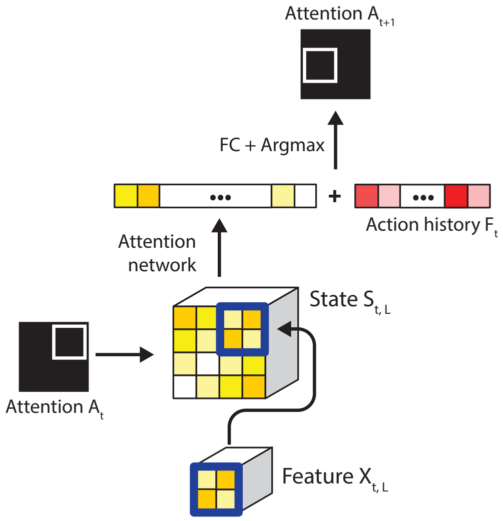


Figure 3: The attention mechanism. The attention network builds on top of the memory unit of a Patchwork Cell and an action history, and predicts the Q-values for the next time step. t denotes the time step and L is a fixed layer in the network. See Sec. 3.1 for details.

and therefore could train the mechanism end-to-end in a supervised way; [1, 27], on the other hand, made use of policy gradients, a flavor of reinforcement learning (RL). Note that experiments in these prior works are limited to datasets such as the MNIST and CIFAR, which explains why we had only limited success with either methodology on complex real-world object detection and segmentation tasks. Our best results were obtained from Q-learning on a discrete action space.

This discrete action space consists of all possible attention sub-windows, and is parameterized by two integers M and N. M denotes how many times a dimension is sliced, while N denotes how many adjacent slices form an attention window. Our experiments contain three such configurations. For $M = 2, N = 1$, there are 4 possible windows of relative size $[\frac{1}{2}, \frac{1}{2}]$. These windows have the top-left corners at $[\frac{i}{2}, \frac{j}{2}], \forall i \in 0, 1; j \in 0, 1$. $M = 4, N = 2$ has 9 possible windows of relative size $[\frac{1}{2}, \frac{1}{2}]$, with the top-left corners at $[\frac{i}{4}, \frac{j}{4}], \forall i \in 0, 1, 2; j \in 0, 1, 2$. $M = 4, N = 1$ has 16 possible windows of relative size $[\frac{1}{4}, \frac{1}{4}]$, with the top-left corners at $[\frac{i}{4}, \frac{j}{4}], \forall i \in 0, 1, 2, 3; j \in 0, 1, 2, 3$.

Note that M and N control the amount of computation a priori. For example, the $M = 2, N = 1$ configuration makes each attention window attending to 25% of the area of a frame, and hence the total computation is reduced to roughly 25%.

Next, we construct the attention network that takes a set of features from the network at time t and map them to Q-values $Q(S_t, A_t; \Theta)$ for deep Q-learning (DQN [28]). The attention network builds on top of the memory unit of a Patchwork Cell, which has a view of the full frame. Additionally, we would also like the attention network to know the action history, to encourage diversity among actions. To this end, we designed a simple exponentially-decaying action history F_t:

$$F_t = \min(\alpha \cdot F_{t-1} + onehot(A_t), 1.0) \qquad (1)$$

where A_t is the action (attention window) at the current time step, and α is a discount factor. As shown in Fig. 3, a Patchwork Cell memory is mapped to a one-dimensional feature vector and concatenated with the action history, where a final fully-connected layer is applied to compute the Q-values.

3.2. Patchwork Cell

Modern deep convolutional networks have receptive fields that are typically larger than the size of the input image. These large receptive fields are crucial in providing later logic (e.g., a detector) with contextual information. However, the hard attention in the recurrent attention module severs these receptive fields, as a cropped input loses any context beyond its own cropped view. Our experiments in Tab. 2 show that this loss of context would cause a significant drop in accuracy if not handled properly. To remedy this, we introduce a building block that is the core of this paper, the Patchwork Cell (see Fig. 1b).

The Patchwork Cell is a stateful replacement for the otherwise stateless 2D-convolution layer with the *SAME* padding that appears throughout modern deep neural networks. The 2D-convolution is defined as the mapping $SameConv2d(X, \Theta) \rightarrow Y$, where X is the input feature map, and Θ is the weight with an odd filter size of $(2k+1)^2$. Without loss of generality, we assume the stride to be 1. Hence, X_t and Y_t have the same spatial resolution $[H, W]$.

The Patchwork Cell adds two states in the form of a memory cell S_t and an action A_t into the mapping: $Patchwork(X_t, A_t, S_t, \Theta) \rightarrow Y_t$, where the subscript t describes the time step. The cell consists of three components: state update, feature propagation and 2D-convolution with *VALID* padding.

State update. The feature extracted from the sub-window X_t overwrites part of the memory S_{t-1} at locations specified by the initial crop window at the time step t, and yields the new state S_t:

$$S_t^{m,n} = \begin{cases} X_t^{m-\frac{a_t \cdot H}{h}, n-\frac{b_t \cdot W}{w}}, & \text{if } 0 \leqslant m - \frac{a_t \cdot H}{h} < H \\ & \text{and } 0 \leqslant n - \frac{b_t \cdot W}{w} < W \\ S_{t-1}^{m,n}, & \text{else} \end{cases}$$

$\forall m \in 0, 1, ..., {}^{H}/_{h}, n \in 0, 1, ..., {}^{W}/_{w}$. The superscripts describe coordinates. $[a_t, b_t]$ are the relative coordinates of the top-left corner of the attention window. $[h, w]$ are the relative height and width of the window, [0.5, 0.5] and [0.25, 0.25] in our experiments.

Feature propagation. We modulate the input feature X_t by encasing it with features from the past state S_{t-1}. This operation is equivalent to cropping a slightly larger feature $\hat{X}_t$ from the current state S_t:

The modulated feature $\hat{X}_t$

$$\hat{X}_t^{i,j} = \begin{cases} S_t^{i-k+\frac{a_t \cdot H}{h}, j-k+\frac{b_t \cdot W}{w}}, & \text{else if } 0 \leqslant i - k + \frac{a_t \cdot H}{h} < \frac{H}{h} \\ & \text{and } 0 \leqslant j - k + \frac{b_t \cdot W}{w} < \frac{W}{w} \\ 0, & \text{else} \end{cases}$$

$\forall i \in 0, 1, ..., H + 2k, j \in 0, 1, ..., W + 2k$.

Finally, Y_t is obtained by performing a 2D-convolution with *VALID* padding on the modulated feature $\hat{X}_t$: $Y_t = ValidConv2d(\hat{X}_t, \Theta)$.

The Patchwork Cell is placed numerous times throughout the network, namely, in front of every *SAME* padding convolution with a kernel size larger than 1, and before global operations. An exact comparison between the off-the-shelf detector and segmenter and their respected Patchwork-modified versions appears in Supplementary Sec. A.

Note that the context provided by the Patchwork Cells is not exact and is an approximation in two ways: 1) One type of approximation is the temporal staleness in the cell memory, as these stale features are extracted from past frames and used to augment features of the current frame. This approximation error becomes more severe the more drastically a scene changes. 2) The context gradually provided by the series of Patchwork Cells is not numerically identical to a natural receptive field, i.e., that of the original pre-cropped input frame. However, in a toy experiment, we found that this type of error is almost negligible, as shown in Supplementary Sec. B.

3.3. Q-Learning reward

Now that we have defined the state, the action, and the Q-function, we need to choose an appropriate reward to train the attention network. A good time-difference ($TD(0)$) reward [37] should be continuous, as a moderately good action shall receive a smaller reward than a vastly superior action. It should also have low variance, and the reward should associate more with the action than the overall difficulty of the scene. Hence, it is inappropriate to naively use the mean-average-precision (mAP) for object detection or the mean intersect-over-union (mIoU) for segmentation as the reward function, as they largely depend on the difficulty of the scene.

In the literature, subtracting a fixed baseline from the raw reward reduces the variance [27]. For our video case, we can do slightly better by using the prediction of the previous

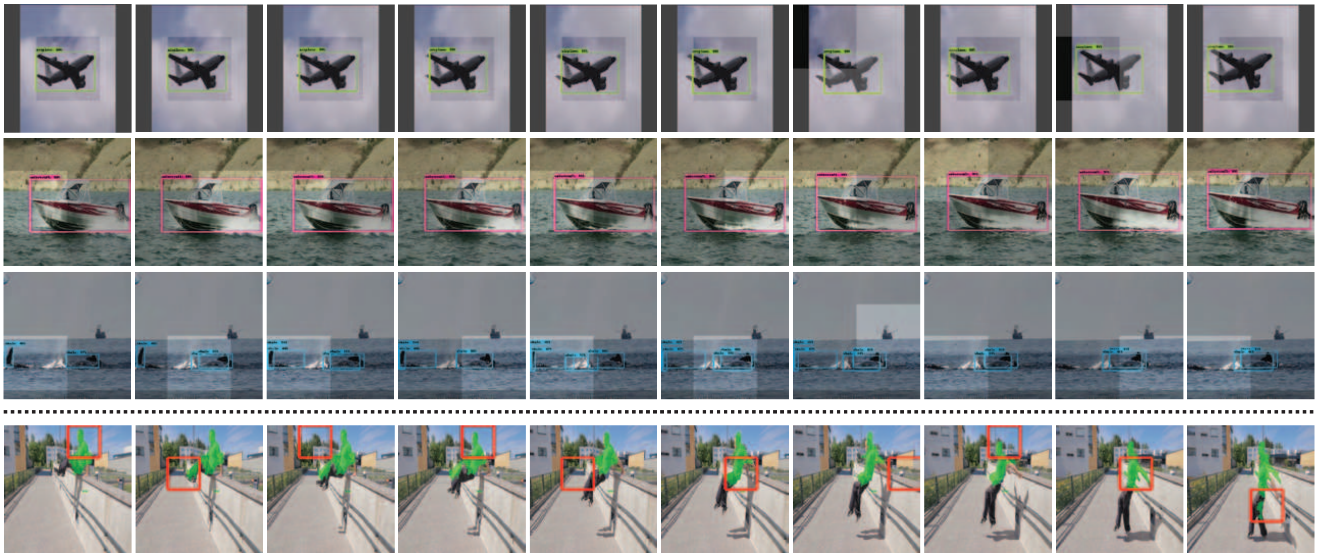


Figure 4: (Best viewed in color) Sample results for the Patchwork network over ten consecutive frames. **First three rows**: object detection results on ImageNet VID. Attention window appears in darker or lighter shading. **Top row**: When the object does not move and can be enclosed within one possible attention window, the window stays until the object away. **2nd row**: When the object is too large for one attention window, the attention window moves around to cover the whole object. **3rd row**: When there are multiple objects of interest spread far apart, the attention window jumps between these objects. **Bottom row**: object segmentation results on DAVIS 2016. Green denotes the segmentation mask, and the red rectangle shows the attention window. Note that the attention window does not simply follow the object of interest. The Q-learning enables a non-greedy attention mechanism that focuses on smaller and potentially new parts while memorizing parts that have been attended before.

frame to form the baseline for the current action. Namely, we define the $TD(0)$ reward R_t as:

$$R_t = \max(0, f(gt_t, p_t) - f(gt_t, p_{t-1})) \tag{2}$$

where gt and p are groundtruth and predictions, $f(gt, p)$ is the metric, and t is the time step. The intuition here is that a reward should only be given if the new attention window has contributed to an increase of a metric f over doing nothing. We force the reward to be non-negative since any negative reward would indicate that having a fresh view of the scene is detrimental in explaining a scene, which should be strictly due to noise.

As for the task-specific metric f, for segmentation, we use the frame-wide mIoU metric (also known as the J-measure in DAVIS [29]). However, for object detection, the mAP measure for object detection is inappropriate as it is quantized and does not reflect incremental improvements. Instead, we draw inspiration from the segmentation task, and define an average box overlap measure as the reward function for object detection:

$$f(gt, p) = \frac{1}{K}\sum_{1}^{K} IoU(gt_k, \tilde{p_k}) \tag{3}$$

where each prediction with a score larger than 0.5 is greedily matched to one of the groundtruth boxes gt_k to yield $\tilde{p_k}$. We then average the IoU scores over all K groundtruth boxes. Note that we omit the class label altogether for simplicity.

We use the DDQN [41] method to train the weights Θ in the Q-function approximation $Q(S_t, A_t; \Theta)$:

$$\Theta \leftarrow \Theta + \alpha(Y_t^Q - Q(S_t, A_t; \Theta))\nabla_\Theta Q(S_t, A_t; \Theta) \tag{4}$$

where

$$Y_t^Q \equiv R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \Theta); \Theta') \tag{5}$$

Θ' is a delayed copy of the weight to make the Q-learning procedure less overly confident. γ is the discount factor and α is the learning rate.

3.4. Training details

There are two significant challenges associated with the training of Patchwork. First, the amount of video training data in the benchmark datasets in the experiments is tiny. ImageNet VID has roughly a thousand unique training videos (technically 3862 but not all unique) whereas DAVIS 2016 only has 30. We decided to make use of the vast amount of single-frame data sources by generating fake videos from them. For a single static image, we randomly sample 2 boxes and build a video sequence by moving the view from one box to the other. We make sure that at least one of the two boxes overlaps with a certain amount of groundtruth detection boxes or segmentation foreground pixels, so that the faked video is not all background. The speed with which the view window moves adheres to a Gaussian distribution. For both the detection and segmentation tasks, we use the COCO and Pascal VOC datasets to augment the original training sets. The usage of these datasets to aid ImageNet VID is not new, as [24] also used them to pre-train their networks.

A second challenge is training with highly correlated samples in mini-batches. This correlation makes the optimization less efficient since multiple samples push gradients roughly in the same direction. More importantly, this makes batch normalization (BN) [16], a critical module to train any deep image model, no longer applicable. To remedy this, we adopt a three-stage training procedure: 1) The video data is split into single frames and mixed with augmentation data to train a single-image model with BN; 2) We place the trained single-image model in a recurrent Patchwork setting and only train the attention network part with Q-learning; 3) With BN frozen, we fine-tune the main network jointly using the task-specific loss and the attention network with Q-learning at a lower learning rate.

4. Experiments

Before diving into the experimental results, we would like to discuss one of the core metrics that is the per-frame latency of the system, which has had different definitions in the past [24, 33].

Average vs. maximum latency. Some algorithms require a variable amount of computation per frame [10, 44]. One notable case is the naive application of a slow model every K frames (keyframes) in a sequence to achieve real-time. Predictions for non-key frames copy from the closest past keyframe. Since copying does not incur any real latency, $K-1$ frames have no latency at all, while one single keyframe takes T amount of time. In this case, the average latency is T/K, which is very different than the maximum latency at T. Both the mean and maximum metrics have their own merits. However, we argue that for most latency-sensitive application, the maximum latency is more critical, as the peak device usage is the bottleneck of the system. Therefore, we report the maximum latency in the experiments section. The average latency appears in Supplementary Sec. C.

Theoretical (FLOPs) vs. empirical latency metric (seconds). The theoretical latency metric measures the latency in the number of floating point operations, which represents the intrinsic complexity of an algorithm. It is independent of the particular implementation and hardware, and therefore, conclusions drawn based on the theoretical measure may remain significant for a long time as implementations are optimized and hardware gets deprecated. The empirical latency metric, on the other hand, measures the latency in absolute time units (e.g., milliseconds). It depends on a particular implementation, so a conclusion can be valid for one hardware, but does not hold for another. We report both metrics in the experiments section as complete as possible. The empirical latency in our experiments is measured using Tensorflow and on a single core Intel Xeon at 3.7GHz.

Model variants. We conducted experiments on multiple variants of the baseline single-frame model and Patchwork for both the object detection and segmentation tasks. The variations include: **depth**: Depth multiplier, with which the number of channels in all layers is multiplied. The latency is roughly proportional to the square of this multiplier. **flip**: Also process the left-right flipped frame and average predictions, and doubles the latency. **resolution**: The default resolution is 384x384. 0.25 resolution has roughly 25% of the original latency. **interval**: The model runs on keyframes that are this many time steps apart, and predictions from keyframes propagate to non-keyframes. While the average latency is proportional to the reciprocal interval, the interval alone has no impact on the maximum latency. **delay**: Predictions are delayed by this many time steps. The delay has no impact on the average latency but can reduce the maximum latency at when combined with a well-chosen interval.

4.1. Object detection

For the object detection task, we conducted our experiments on ImageNet VID 2015 [32], which contains 30 categories with 3862 training and 555 validation video clips. We report our results on the validation set and the mean-average-precision (mAP) at 0.5 IoU, as it is the norm for this dataset.

The system inherits from an off-the-shelf MobileNetV2 + SSD detection architecture [33]. In this work, we do not yet consider more expensive models, e.g., ResNet + Faster RCNN, since these models are too slow to run in real-time. For the baseline, we trained a single-frame detection model by fine-tuning from an ImageNet checkpoint using data sampled from ImageNet VID, ImageNet DET and COCO. This baseline detector achieves 54.7%, which is competitive with other recent works that focus on the low-latency case [24, 45].

The left figure in Fig. 5 shows a theoretical maximum latency vs. accuracy comparison between Patchwork variants and their equivalent single-frame setups. Patchwork is consistently superior than the baseline. Most notably, the M=4,N=2 configuration at roughly 25% of its original computation, only loses 0.4% in mAP (54.7% vs. 54.3%) compared to the baseline single-frame MobileNetV2 model. Moreover, when we reinvest the saved resource in additional channels in the layers and model ensembling, we show that we can boost the results to 58.7% at the origi-

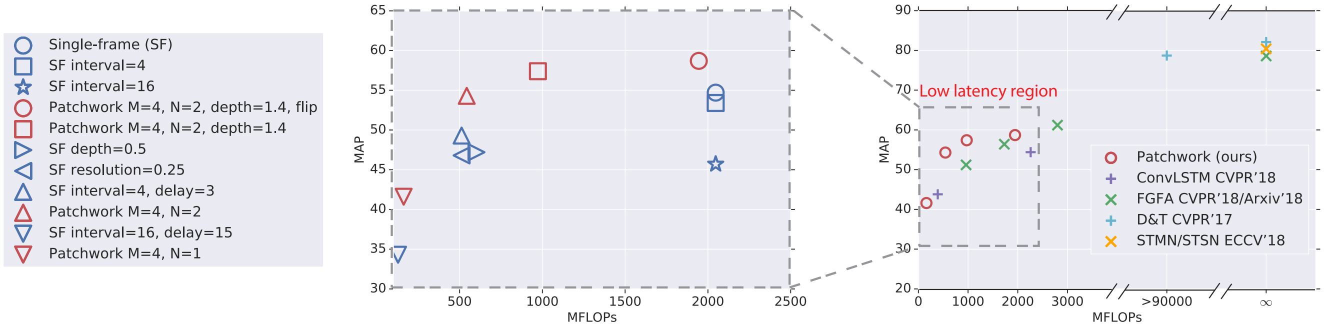


Figure 5: (Best viewed in color) A theoretical maximum latency vs. accuracy comparison on ImageNet VID. **Left figure**: Comparison between single-frame and Patchwork variants (exact values given in Supplementary Tab. 4). **Right figure**: We put Patchwork in perspective with other published works on ImageNet VID: ConvLSTM [24], FGFA [44, 45], D&T [10], STSN [2] and STMN [42]. Most of these use the ResNet-101 as the feature extractor, which by itself has 90 billion FLOPs when applied on a 600x600 image as described in the papers. Methods that require to look ahead for more than ten frames or optimize for the whole video at once are considered to have an infinite latency.

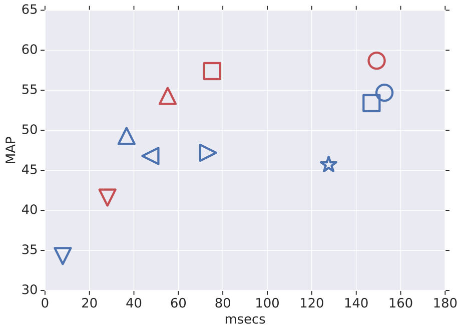


Figure 6: An empirical maximum latency vs. accuracy comparison between single-frame (blue) and Patchwork (red) variants. Same legends as in Fig. 5.

M	N	Policy	MFLOPs	mAP ↑
-	-	Single-frame	512	49.3
4	2	Random	541	52.5
2	1	Scanning	541	51.7
4	2	DQN	543	**54.3**
-	-	Single-frame	128	34.3
4	1	Random	161	37.0
4	1	Scanning	161	37.9
4	1	DQN	162	**41.6**

Table 1: A comparison between different attention policies. Single-frame and DQN results match those in Fig. 5. **Random** picks an action at random. **Scanning** iterates over the action space following a fixed pattern.

Policy	Patchwork Cell?	mAP ↑
Scanning	✗	39.5
	✓	51.7
DQN	✗	43.2
	✓	54.3

Table 2: An ablation study demonstrating the necessity for the proposed Patchwork Cell in recurrent attention models [27]. Experiments are performed using both the reinforced and the scanning policy.

nal computational cost. The empirical maximum latency vs. accuracy figure tells a similar story (Fig. 6) in the high and mid-accuracy range. For the low latency range, the overhead caused by the Patchwork ops starts to dominate. This overhead is mostly because the Patchwork ops are not as optimized as the rest of the network, which mostly consists of highly optimized convolutions. Sampled qualitative results appear in Fig. 4.

Two ablations studies were conducted to highlight the two main contributions of this work: the Q-learned attention mechanism and the Patchwork cell.

Q-learned vs. hand-designed policies. Instead of training the attention mechanism via Q-learning, one could use a manually designed policy. One such policy is the random policy, where we select one of the possible actions at random. Another baseline policy is the scanning policy, where the policy deterministically scans through all possible spatial locations. Tab. 1 shows a comparison on the 1/4 and the 1/16 cost setups. In both cases, the results from the random and scanning policies are fairly comparable. The Q-learned policy is roughly 2-4% better than the manually designed policies, which are in turn 2-3% better than the best baseline single-frame method with a comparable maximum latency. In other words, even without a smart attention mechanism, one can still achieve a reasonable improvement by using Patchwork Cells alone.

Patchwork Cell vs. vanilla recurrent attention network. The Patchwork Cell is a core contribution to previous recurrent attention models [27] and is the key to make them successful on complex object detection and segmentation tasks. Tab. 2 shows the results of the Patchwork net-

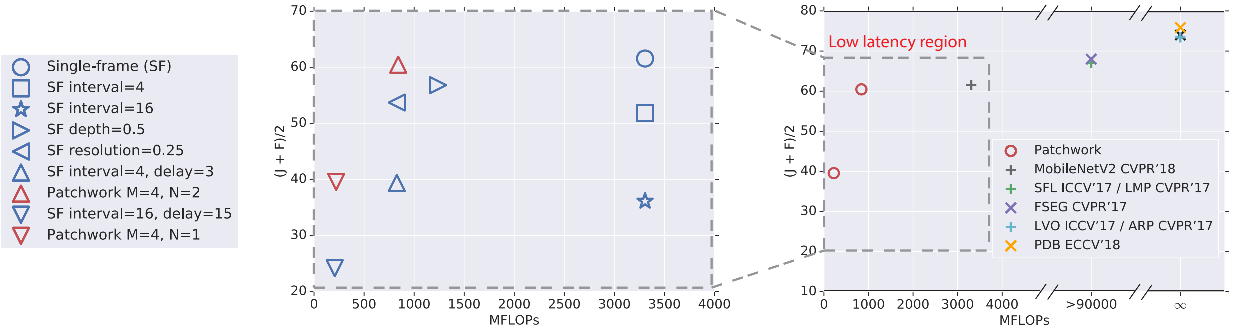


Figure 7: (Best viewed in color) A theoretical maximum latency vs. accuracy comparison on the DAVIS 2016 video object segmentation dataset. $\mathcal{J}$ and $\mathcal{F}$ are the metrics commonly used in DAVIS. **Left figure**: Comparison between single-frame and Patchwork variants (exact values given in Supplementary Tab. 5). **Right figure**: We put Patchwork in perspective with other published works on DAVIS 2016: MobileNetV2 [33], SFL [7], LMP [40], FSEG [18], LVO [39], ARP [20] and PDB [36]. Most works are based on ResNet-101 and has a substantially higher latency than Patchwork.

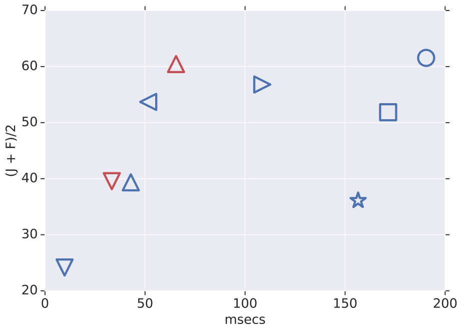


Figure 8: An empirical maximum latency vs. accuracy comparison between single-frame (blue) and Patchwork (red) variants. Same legends as in Fig. 7.

work with or without the Patchwork Cells, by using either the scanning policy or the DQN-learned policy. In both cases, we observe a significant drop in mAP if the Patchwork Cells are not present.

Finally, the right figure in Fig. 5 shows our results with other recent publications on ImageNet Video. Note that most of the other works are based on ResNet-101 and has a significantly higher latency than Patchwork and are not suited for latency-sensitive applications.

4.2. Object segmentation

DAVIS [29] has been instrumental in driving the research in video object segmentation. It has been used in two setups: in the semi-supervised setup where the groundtruth segmentation mask for the first frame is present, or the unsupervised configuration where there is no prior information at all. For the 2016 version of DAVIS, where there is at most one object per video, the unsupervised setting becomes a binary foreground-background segmentation task. For simplicity, we choose to test Patchwork in the unsupervised environment. Note that the task of segmenting the foreground per frame is relatively ambiguous and DAVIS is not exhaustively labeled. However, we hope that an algorithmic improvement is still reflected in an increase in our quantitative measure. DAVIS 2016 contains 30 training and 20 validation sequences, each of which includes a single object, be it a person on a bike, a car in a busy street.

Our baseline single-frame architecture is again directly taken from the MobileNetV2 paper [33]. A maximum latency vs. accuracy comparison is shown in the left figure in Fig. 7 and in Fig. 8 (see Supplementary Sec. C for more details). Patchwork occupies the Pareto points in the mid- and low latency range. Notably, at roughly ¼ of computation, we only observe a minimal loss in the $\mathcal{J}$ (63.6% vs. 62.1%) and the $\mathcal{F}$ (59.5% vs. 58.8%) metrics. Results from other works on the unsupervised DAVIS 2016 appears in the right figure in Fig. 7. Note that most referenced works are based on ResNet-101 and has a substantially higher latency than Patchwork.

5. Conclusion

The lost spatial context presents a fundamental limitation for the hard attention mechanism in deep networks, and we believe it is one of the main reasons why the hard attention idea has not been more popular in real-world applications. As demonstrated in the experiments, Patchwork successfully mitigates this issue. Together with a Q-learned attention mechanism, in the Patchwork architecture, we present a new paradigm to efficient stream processing.

Acknowledgments. We like to thank the anonymous reviewers, Henrik Kretzschmar, Benjamin Sapp and Scott Ettinger for their helpful comments.

References

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015. 2, 3

[2] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *ECCV*, 2018. 7

[3] Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 2

[4] Marisa Carrasco. Visual attention: The past 25 years. *Vision Research*, 2011. 1

[5] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *CVPR*, 2018. 1

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 2018. 1

[7] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *ICCV*, 2017. 8

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 2

[9] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. In *NIPS*, 2016. 1

[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. *ICCV*, 2017. 6, 7

[11] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 1

[12] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015. 2, 3

[13] Junwei Han, Le Yang, Dingwen Zhang, Xiaojun Chang, and Xiaodan Liang. Reinforcement cutting-agent learning for video object segmentation. In *CVPR*, 2018. 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 6

[17] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CVPR*, 2018. 2

[18] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *CVPR*, 2017. 8

[19] Dinesh Jayaraman and Kristen Grauman. End-to-end policy learning for active visual categorization. *PAMI*, 2018. 2

[20] Yeong Jun Koh and Chang-Su Kim. Primary object segmentation in videos based on region augmentation and reduction. In *CVPR*, 2017. 8

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[22] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 2

[23] Yule Li, Jianping Shi, and Dahua Lin. Low-latency video semantic segmentation. In *CVPR*, 2018. 2

[24] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. *CVPR*, 2018. 6, 7

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 1, 2, 11, 12

[26] Julie Markant and Dima Amso. Leveling the playing field: Attention mitigates the effects of intelligence on memory. *Cognition*, 2014. 1

[27] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014. 2, 3, 4, 7

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015. 3

[29] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 1, 5, 8

[30] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016. 2

[31] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *PAMI*, 2017. 1, 2

[32] 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. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1, 6

[33] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *CVPR*, 2018. 2, 3, 6, 8, 11

[34] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *ECCV*, 2016. 2

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[36] Hongmei Song, Wenguan Wang, Sanyuan Zhao, Jianbing Shen, and Kin-Man Lam. Pyramid dilated deeper convlstm for video salient object detection. In *ECCV*, 2018. 8

[37] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998. 4

[38] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1

[39] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning motion patterns in videos. In *CVPR*, 2017. 8

[40] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. In *ICCV*, 2017. 8

[41] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *AAAI*, 2016. 5

[42] Fanyi Xiao and Yong Jae Lee. Video object detection with an aligned spatial-temporal memory. In *ECCV*, 2018. 7

[43] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017. 2

[44] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *CVPR*, 2018. 6, 7

[45] Xizhou Zhu, Jifeng Dai, Xingchi Zhu, Yichen Wei, and Lu Yuan. Towards high performance video object detection for mobiles. *arXiv preprint arXiv:1804.05830*, 2018. 2, 6, 7