Architecture Search of Dynamic Cells for Semantic Video Segmentation

Vladimir Nekrasov Hao Chen Chunhua Shen Ian Reid
The University of Adelaide, Australia
E-mail: {vladimir.nekrasov, hao.chen01, chunhua.shen, ian.reid}@adelaide.edu.au

Abstract

In semantic video segmentation the goal is to acquire consistent dense semantic labelling across image frames. To this end, recent approaches have been reliant on manually arranged operations applied on top of static semantic segmentation networks – with the most prominent building block being the optical flow able to provide information about scene dynamics. Related to that is the line of research concerned with speeding up static networks by approximating expensive parts of them with cheaper alternatives, while propagating information from previous frames. In this work we attempt to come up with generalisation of those methods, and instead of manually designing contextual blocks that connect per-frame outputs, we propose a neural architecture search solution, where the choice of operations together with their sequential arrangement are being predicted by a separate neural network. We showcase that such generalisation leads to stable and accurate results across common benchmarks, such as CityScapes and CamVid datasets. Importantly, the proposed methodology takes only 2 *GPU-days, finds high-performing cells and does not rely on the expensive optical flow computation.*

1. Introduction

Human beings are well-equipped by evolution to quickly observe changes in dynamic environments. From merely few seconds of studying an unknown scene, we are able to coherently map out its main constituents. In contrast, static semantic segmentation networks would perform poorly in such conditions, and may as well produce contradictory predictions across the frames. Therefore, the question arises of how to make the static models suitable for segmenting continuously evolving scenes?

One well-known approach would be to use the optical flow that describes the motion in the scene between adjacent frames [10, 34]. The optical flow calculation tends to be expensive and also comes with several notable disadvantages, among which its inability to deal with occlusions and newly appeared objects. Nevertheless, as shown by Gadde *et al*. [10], a relatively poor estimate of the optical flow may still carry significant benefits, not the least of which lies in computational savings.

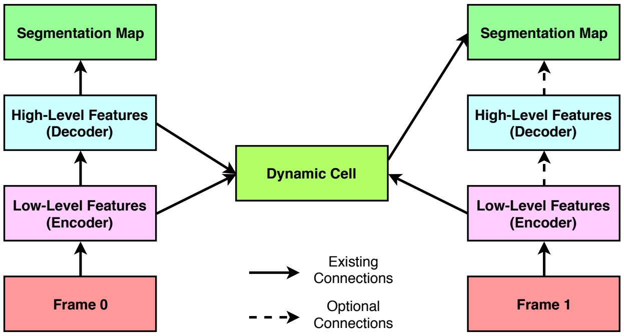

Segmentation Map
High-Level Features (Decoder)
Low-Level Features (Encoder)
Frame 0
Dynamic Cell
Existing Connections
Optional Connections
Segmentation Map
High-Level Features (Decoder)
Low-Level Features (Encoder)
Frame 1

Figure 1. Semantic video segmentation approaches tend to comprise a dynamic cell that takes as inputs the information from the previous and current frames, and outputs the segmentation mask. For example, the dynamic cell can calculate the optical flow [10], or predict convolutional filters [16]. In this work we use NAS to discover novel and high-performing dynamic cells.

Alternatively, one may choose to model which information must be propagated across the frames, *e.g*. with the help of a recurrent neural network with memory units [22]. Even more biologically plausible are the models that compute different features at various time-scales [26], in a vein similar to neural spikes. Naturally, this comes with its own set of disadvantages, most notably the difficulty of choosing an appropriate scheduling regime for updating individual parts of the network.

Yet another complementary line of work focuses on approximating the expensive per-frame forward pass with cheaper alternatives: *e.g*. Li *et al*. [16] predicted local filters to be applied on the segmentation prediction from the previous frame, while Jain *et al*. [14] used a larger network for key frames and directly employed a smaller one for consecutive frames. Such savings may allow to re-use more expensive optical flow methods without a significant slowdown, but the choice of key frames can be crucial and not readily justifiable.

Looking closely at the aforementioned approaches for video semantic segmentation, one may notice an easily discernible pattern: a typical video segmentation network predicts a labelling of the current frame based on the information propagated from the previous one and hidden representations of the current one (Fig. 1). While seemingly obvious, it possesses certain variations depending on the goal – *e.g.* whether efficiency, or real-time performance is desired. Importantly, what we would like to emphasise here is that, while technically sound, all the current approaches have been manually designed and have not considered any interplay between different building blocks.

Starting from that general pattern we instead propose to leverage the neural architecture search (*NAS*) [35] methodology to find contextual blocks that enhance the per-frame segmentation network with dynamic components. This motivation is justified by recent results achieved using NAS on such tasks as image classification [36, 18], language modelling [23] and static semantic segmentation [5, 20], that oftentimes outperform manually designed networks. We build upon those results and adapt current approaches in a way suitable for handling the dynamic nature of dense per-pixel classification. To the best of our knowledge, we are the first to consider the application of NAS to the task of video semantic segmentation.

Our automated approach comes with certain benefits, concretely:

i.) it considers a larger span of initial building blocks than any previous work,

ii.) it empirically evaluates different design structures and finds most promising ones, and

iii.) it requires only few GPU-days to find a set of high-performing structures.

Furthermore, although we do not consider it in this work, the proposed methodology can further be extended to take into account different specific objectives (even non-differentiable), such as runtime [27].

2. Related Work

2.1. Static semantic segmentation

Most recent approaches in static semantic segmentation have been exploiting fully convolutional neural networks [19]. Typical methods are based either on the encoder-decoder structure with skip-connections [19, 17], dilated convolutional layers [30, 32, 6], or the combination of the above [7]. Per-frame instantiations of these networks are usually computationally expensive, hence, several works have considered building light-weight segmentation architectures [31, 21]. Nevertheless, due to the lack of information propagation between frames, these networks perform poorly on videos and are unable to provide consistent results.

2.2. Dynamic semantic segmentation

One of the first lines of work in video segmentation has been built upon the usage of the optical flow [34], in which features extracted from the previous frame are propagated to the current one via warping. This usually results in a slight computational overhead, although as noted by Gadde *et al.* [10] an easily attainable noisy estimate of the optical flow still carries significant benefits. Nevertheless, the optical flow does not fair well in situations when scenes are undergoing substantial changes with novel objects constantly appearing and multiple occlusions being present. Thus, Jain *et al.* [14] have proposed to combine the optical flow estimate with a relatively cheaper approximation of the current frame using a smaller network. Xu *et al.* [28] have chosen to assign different image regions to two different networks to process: while the first one – deep and slow – works on regions that have significantly changed, the second one – shallow – predicts new features based on the optical flow information. In a similar vein, Nilsson and Sminchisescu [22] have propagated labels from the previous frame at only those pixels where the optical flow estimate is reliable.

A seemingly different approach, proposed by Li *et al.* [16], instead predicts local convolutional kernels based on the low-level representation of the current frame that are applied on the prediction from the previous frame. Importantly, while the current estimate is being used for next frame, a more accurate one is being computed in parallel for future re-use.

In yet another line of work, Chandra *et al.* [3] have adapted Deep Gaussian Random Field [4] to handle temporal information by predicting besides unary and spatial pairwise terms also temporal pairwise terms, efficiently propagating features between frames.

2.3. Neural Architecture Search

NAS methods aim to find high-performing architectures in an automated way. Here, we consider the reinforcement learning-based (RL) approach [35], where a separate recurrent neural network (controller) outputs a sequence of tokens describing an architecture that should provide highest score on the holdout validation set.

While there is no prior work on NAS for video segmentation, two results in static segmentation are worth mentioning: Chen *et al.* [5] used a random search to find a single set of operations (so-called 'cell') on the top of the DeepLab architecture [6], while Nekrasov *et al.* [20] exploited RL to find a cell together with the topological structure of the encoder-decoder type of architecture. We borrow one of the architectures found by Nekrasov *et al.* as our static baseline, and extend their NAS approach for video segmentation. Since we are only searching for the dynamic component that connects different instantiations of the already

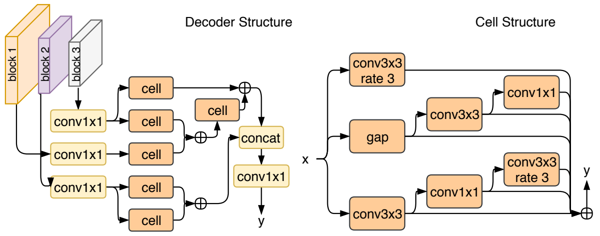


Figure 2. Network structure of *arch2* from [20]. *'gap'* stands for global average pooling.

pre-trained static segmentation network, we are able to train and evaluate each candidate in a short amount of time, the trait that is extremely important for all NAS methods.

3. Methodology

As noted in introduction and depicted in Fig. 1, we attempt to generalise previous solutions for video semantic segmentation in such a way that NAS methods become readily applicable. To this end, we look for a single cell that connects representations from the previous frame and enhances current predictions without a significant overhead. What follows is the description of the input space (Sect. 3.1), the search space (Sect. 3.2), and the search approach (Sect. 3.3).

3.1. Input space

We consider the *arch2* network from the work of Nekrasov *et al*. [20]. It is an encoder-decoder type of the segmentation network with the encoder being a light-weight classifier (MobileNet-v2 [24]), and the decoder being an automatically discovered structure presented in Fig. 2. This architecture strikes a fine balance between accuracy and runtime, both being important characteristics for semantic video segmentation. Here it should be noted that the application of our methodology is not directly tied to a concrete architecture and can be easily adapted to work with other networks.

In the proposed setup, the static network is applied end-to-end on the first frame and three outputs are being recorded: an intermediate representation - in this case, the encoder's output with the resolution of $\frac{1}{32}$ of the input image ($layer$ 4), the decoder's output before (dec) and after the final classifier ($pred$) - both with resolutions of $\frac{1}{8}$ of the input image and with 64 and C numbers of channels, correspondingly, where C is the number of output classes. For the second frame, we record three outputs from the encoder only - two intermediate ones with the resolutions of $\frac{1}{8}$ ($layer$ 2) and $\frac{1}{16}$ ($layer$ 3), respectively, and the final one with the resolution of $\frac{1}{32}$ ($layer$ 4).

We rely on the dynamic cell, the layout of which will be described below, to predict the semantic labelling of the current frame given 5 inputs: $layer$ 4 and dec from the previous frame ($layer$ 4$-prev$ and $dec-prev$, correspondingly), and $layers$ 2$-$3$-$4 from the current one. This way, we do not have to execute the decoder part of the static segmentation network on the current frame (thus decreasing latency), at the same time re-using information from the previous frame. Note also that the output of the dynamic cell can serve as the input dec for the next frame.

3.2. Search space

We rely on an LSTM-based controller to predict a sequence of operations together with locations where they should be applied in order to form a dynamic cell [20]. Concretely, we first choose two layers out of the provided five (with replacement), two corresponding operations that need to be applied on each of them, and an aggregation operation that combines two inputs into a single output. On the next step, we repeat this process, but now we are sampling two layers out of six possible, with the aggregated result being added into the sampling pool. This process can be repeated multiple times, with the final output being formed by the concatenation of all non-sampled aggregated results.

We rely on a similar set of operations as for static segmentation (Table 1), and in order to enable the dynamic cell to apply convolutional filters on irregular grids, we also include deformable 3×3 convolution [33].

ID	Description
0	separable conv 3×3
1	global average pooling followed by upsampling and conv 1×1
2	separable conv 3×3 with dilation rate 3
3	separable conv 5×5 with dilation rate 6
4	skip-connection
5	deformable 3×3 convolution

Table 1. Description of operations used in the search process.

While Nekrasov *et al*. [20] simply summed up two different inputs at each step, here to compensate for the dynamic nature of our problem we consider a set of aggregation operations given in Table 2.

Based on the previous works, we conjecture that this set of operations will be sufficient for the task of video segmentation, and we provide experimental results to support this claim. Please see the full code definitions of each operation in the supplementary material.

3.3. Finding optimal architectures

We assume that there exists a video dataset that comes with segmentation annotations for at least a subset of consecutive frames. From it, we build pairs (or triplets) of

ID	Description
0	summation with per-channel learnable weights per each input
1	channel-wise concatenation of two inputs followed by conv 1×1 to reduce the number of channels to the original size
2	(weight) predictive operation, where the first input becomes a set of spatial convolutional filters (weights) applied on the second one
3	bilinear sampling of the first input, where an affine grid is predicted based on the values of the second input [13]
4	3D-convolution where two inputs are stacked together forming a new dimension with $2 \times 3 \times 3$ convolution applied on top
5	dense attention: *i.e.* element-wise multiplication between the first input and the sigmoid-activated second one

Table 2. Description of aggregation operations used in the search process.

frames such that in each sequence all the frames following the first one are always annotated. As commonly done, we further divide this set into two disjoint parts – meta-train and meta-val. We further assume an existence of the static segmentation network pre-trained on this dataset[1] – in particular, *arch2* from [20]. As mentioned above, we chose this particular architecture due to its compactness and low latency.

The controller samples a structure of the dynamic cell which we train on the meta-train set and evaluate on meta-val. As done in [20], we consider the geometric mean of three metrics as the validation score: mean intersection-over-union (*mIoU*), frequency-weighted IoU (*fwIoU*) and mean-pixel accuracy (*mAcc*). This score is used by the controller to update its weights, and the process is repeated multiple times. After that, one can either sample several cells from the trained controller, or simply choose best found cells that achieved highest results during the search process.

4. Experiments

We conduct all our experiments on two popular video segmentation benchmark datasets – CamVid [2] and CityScapes [8].

The first one, *CamVid*, comprises 701 RGB images of resolution 480×360 densely annotated at 1FPS into 11 categories. Following the previous work [1], we use the dataset splits of 367 images for training and 233 – for testing. We train generated architectures with batches of examples each comprising 3 consecutive frames -- that is capturing 3 seconds of video.

The *CityScapes* dataset contains 5000 high-resolution 2048×1024 images densely labelled with 19 semantic classes - 2975 for training, 500 for validation and 1525 for testing, respectively. In addition to that, raw unannotated frames extracted from videos captured at the frame rate of 17FPS are also provided. For each annotated example, we add an image frame that precedes it (*i.e.* $1/17$ seconds in the past) and train architectures with batches of sequences of length 2, in which the second frame is always annotated.

In each case, we initialise the decoder's output *dec* on the first frame in the sequence using the pre-trained static segmentation network, and rely on the dynamic cell at all following frames for the length of the sequence as described in Sect. 3.1. To update the dynamic cell weights, we sum up cross-entropy loss terms at each frame after the first one and back-propagate the gradients.

For both, search and training, we exploit a single V100 GPU with 32GB of memory.

4.1. Search

For searching we only employ the training splits of each dataset. We further divide each randomly in 2 non-overlapping sets – meta-train (90%) and meta-val (10%). We pre-compute all required outputs from the pre-trained static network and store them in memory. The static network is kept unchanged during the whole search process. Each generated architecture is trained on the meta-train split and evaluated on meta-val. We keep track of average performance and apply early stopping halfway through the training if the generated architecture is un-promising as done in [20].

Our controller is a two-layer LSTM with 100 hidden units randomly initialised from uniform distribution [20]. The controller is trained with PPO [25] with the learning rate of $1e{-}4$. To reduce the size of generated cells, we set the number of emitted layers (each layer is a string of five tokens as described in Sect. 3.2) to 5 on CamVid and to 4 on CityScapes.

For *CamVid*, we train predicted cells on mini-batches of 48 sequences for 20 epochs with the learning rate of $8e{-}3$ and the Adam learning rule [15]. Each image–segmentation mask pair in the sequence is cropped to 350 with the shorter side being mean-padded to 550. No transformations are applied to the validation sequences.

For *CityScapes*, we train for 10 epochs with 48 sequences each cropped to 512×512 with the longer side being resized to 1024.

[1] Please refer to the supplementary material for the details on pre-training of static segmentation networks.

Results

We visualise the progress of each metric together with the reward signal on each dataset in Fig. 3. Although the rewards are not directly comparable between the datasets, the growth dynamics on both datasets signal that the controller is able to discover better architectures throughout the search process across all the metrics.

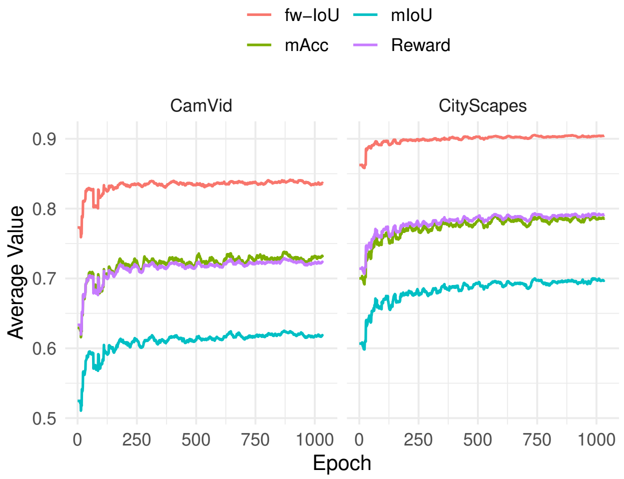


Figure 3. Average search metrics on CamVid and CityScapes datasets.

We further look at the distributions of sampled operations, aggregation operations and input layers plotted on Fig. 4. On both datasets, global average pooling and separable $5{\times}5$ convolution with dilation rate 6 are sampled less frequently than other operations, potentially indicating that these layers could be omitted from the search process. On average, the controller trained on CityScapes prefers sampling deformable convolution (Fig. 4a), while the CamVid one – separable $3{\times}3$ convolution (Fig. 4d).

In terms of aggregation operations, the dynamics between two controllers vary significantly: the CamVid-based controller tend to rely on dense attention, while omitting the predictive operation (Fig. 4e). In contrast, the CityScapes controller is more likely to apply bilinear sampling on an affine grid, and to ignore predictive operation together with dense attention (Fig. 4b).

When sampling the input layers, the controllers behave similarly: in particular, both tend to skip $layer$ 4 from the previous and current frames, which also permits computational savings in the encoder. The CityScapes controller extensively uses information from the previous dec layer (Fig. 4c) (as the difference in frames is very minor), while the CamVid one – from $layer$ 2 of the current frame (Fig. 4f).

Importantly, these observations indicate that two controllers trained on two different datasets exhibit various patterns, capturing dataset-specific attributes (such as frame rate between consecutive frames) in order to discover better performing architectures.

4.2. End-to-end Training

We further select top-2 performing dynamic cells on each dataset to train end-to-end on full training sets for longer.

In particular, for *CamVid*, we pre-train the dynamic cell with Adam and the learning rate of $8e{-}3$ for 10 epochs with the batch size of 16 sequences. Then we decrease the cell's learning rate in half, and fine-tune the whole architecture (i.e., with the per-frame segmentation network) end-to-end for 100 epochs – the static network weights are updated using SGD with momentum of 0.9 and the learning rate of $5e{-}4$. Each sample in the batch is cropped to $600{\times}600$ with the shorter side being padded to 400.

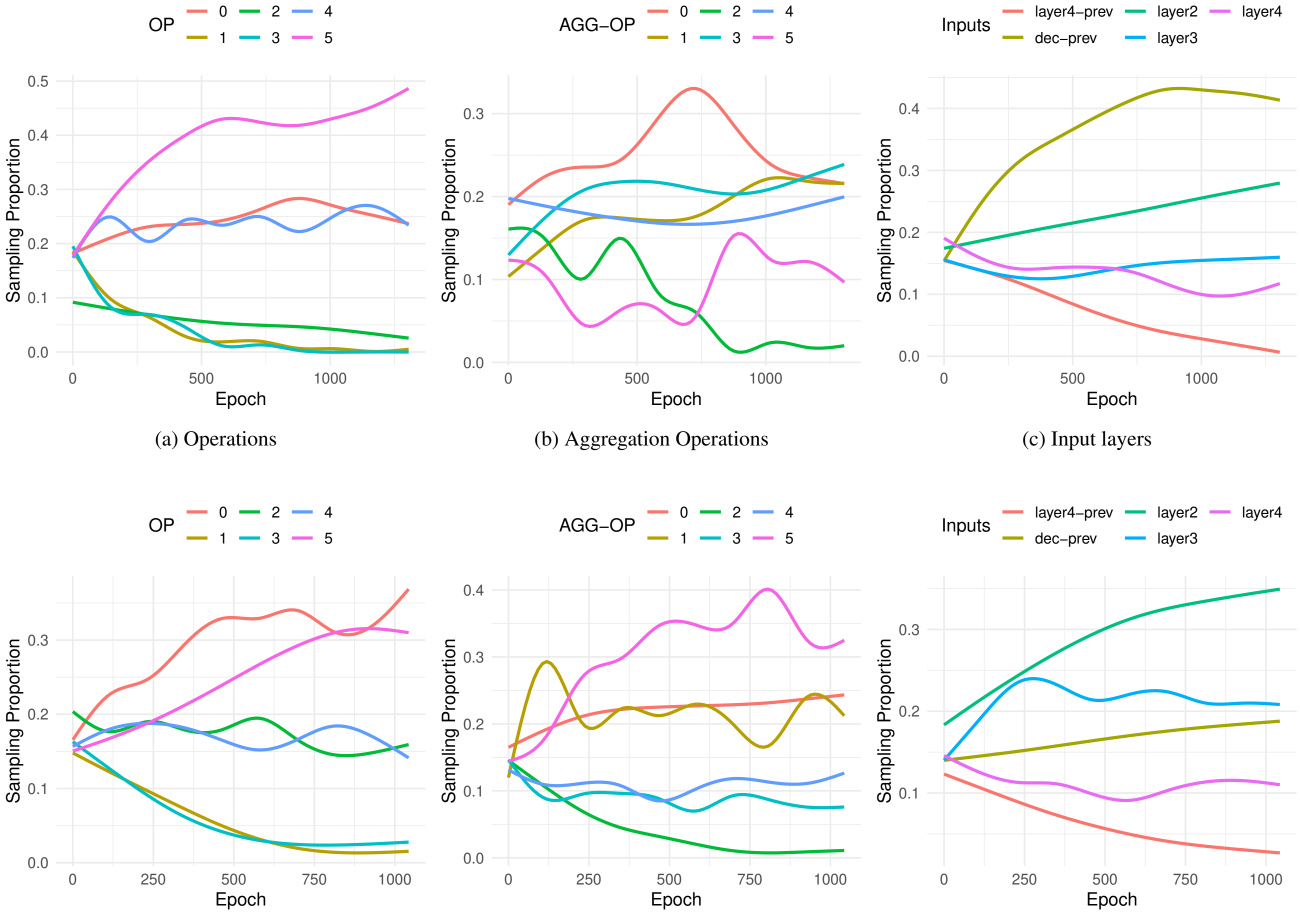
On *CityScapes* we pre-train for 200 epochs with the batch size of 16 sequences and fine-tune end-to-end for 200 epochs. Each example in the batch is cropped to $769{\times}769$.

CamVid Results

We provide quantitative results on CamVid in Table 3. The inclusion of dynamic cells in both cases leads to an improvement over baseline by more than 1%. Importantly, with the exclusion of the first frame in each sequence, we do not rely on expensive computations involving the static decoder.

Both our models perform comparably to other state-of-the-art video segmentation networks even though the backbone that we rely on – MobileNet-v2 [24] – is much smaller in comparison to ResNet-101 [11] exploited by Chandra *et al.* [3], or DilatedNet [29] – by Gadde *et al.* [10] and GRFP [22]. Furthermore, we did not make any use of higher-resolution images of $960{\times}720$ to further improve our scores.

Perhaps surprisingly, we found minuscule changes when removing the dynamic connections between frames in the discovered architectures. This implies that our approach was able to find a smaller and better performing static segmentation network, which is not unexpected as the original baseline was discovered using a completely different dataset (*i.e.* PASCAL VOC [9]). On the other hand, the lack of dynamism can be explained by significant changes in consecutive frames in the environment with fast moving vehicles like cars – remember that adjacent frames are 1 second apart. To confirm this intuition, we conducted another experiment using raw video frames from CamVid: in particular, we re-trained the first architecture, 'cell0', on sequences of size 3, where the last frame is annotated and the first two, $2/30$ and $1/30$ seconds before the last frame, are



(a) Operations

(b) Aggregation Operations

(c) Input layers

(d) Operations

(e) Aggregation Operations

(f) Input layers

Figure 4. Average sampling proportion of operations, aggregation operations and input layers on CityScapes (**a-c**) and CamVid (**d-f**). Please refer to Tables 1 and 2 for the description of operations.

not. In this case, we found that the discovered architecture does exhibit dynamism while still outperforming the static baseline[2].

Method	mIoU,%	mAcc,%	gAcc,%	tIoU,%
per-frame baseline	65.3	76.1	90.8	41.4
w/ cell0	66.6	77.6	91.1	42.6
w/ cell1	66.9	78.5	90.1	42.4
GRFP [22]	66.1	-	-	-
Chandra *et al*. [3]	67.0	-	-	-
Gadde *et al*. [10]	67.1	-	-	36.6

Table 3. Quantitative results on the test set of CamVid. Note that our method uses MobileNet-v2 as the encoder network. For trimap IoU the width is 3.

CityScapes Results

We include the validation results of two discovered cells on CityScapes in Table 4. Once again, both dynamic cells are able to outperform the per-frame baseline by 1.2%. Furthermore, our models achieve favourable results in comparison to other video segmentation methods, all of which employ significantly larger backbones and, with the exclusion of Li *et al*. [16], all rely on the optical flow computation. Note also that Gadde *et al*. [10] improved over their respective static baseline by 1.2%, too, while introducing a non-negligible overhead of 40ms; and Li *et al*. [16] compromised more than 3% of the baseline score in order to reduce the latency. In contrast, we overcame our static baseline, while reducing the average per-frame latency (Table 5). Furthermore, we do witness dynamism in both cells to a varying extent as evident by a significant drop in performance when dynamic connections are removed.

[2] Please refer to the supplementary material for more details on the CamVid experiment with raw video frames.

Method	mIoU,%	mAcc,%	tIoU, %
per-frame baseline	74.4	82.6	40.1
w/ cell2[3]	75.6	84.4	41.5
no dynamism	28.2	39.6	15.5
w/ cell3[4]	75.6	83.7	41.5
no dynamism	44.1	54.4	21.0
GRFP(5) [22]	69.5	-	-
Xu *et al.* [28]	70.4	-	-
Li *et al.* [16]	76.8	-	-
Gadde *et al.* [10]	80.6	-	42.1

Table 4. Comparison with other video segmentation approaches on the val set of CityScapes. Note that our method uses MobileNet-v2 as the encoder network. For tIoU, the trimap width is 3. *No dynamism* implies that there are no connections between adjacent frames.

A few inference examples are visualised in Fig. 5. As can be seen, the dynamic cells enhance the per-frame baseline results and identify partially occluded vehicles more accurately (rows $1-2$, 5), while also avoiding misclassification of traffic signs at pixels with similar texture patterns (rows $2-4$).

4.3. Details of Discovered Architectures

We include characteristics of our networks together with numbers reported by others in Table 5, assuming that the dynamic cell is used on all the frames starting from the second and does not exhibit a significant drift in quality (we discuss ways of overcoming it in the final section).

Concretely, all our architectures contain at most 3.4M parameters while having an average per-frame runtime of 50ms on high-resolution 2048×1024 images. This is possible due to both the network design and the exclusion of the optical flow computation.

Method	GPU	Input Size	Param.,M	Avg. RT,ms
Baseline	1080Ti	2048×1024	**2.85**	92.4±0.3
w/ cell0	1080Ti	2048×1024	3.35	51.5±1.8
w/ cell1	1080Ti	2048×1024	3.19	52.6±1.8
w/ cell2	1080Ti	2048×1024	3.24	51.5±1.9
w/ cell3	1080Ti	2048×1024	3.30	**50.5±1.9**
GRFP [22]	TitanX	512×512	>40	685
Li *et al.* [16]	–	2048×1024	>40	171
Gadde *et al.* [10]	TitanX	2048×1024	>60	3040

Table 5. Number of parameters and average runtime (RT) comparison between different models. To compute average runtime with dynamic cells, we use the baseline on the first frame and the dynamic cell on the rest (1000 frames in total). Where possible, we report same characteristics for other methods.

All the trained cells are visualised in Fig. 6. Notably, layers with deformable convolution are present in all architectures. To propagate information from the previous frame, each cell exploits the dec output instead of $layer\ 4$. All the cells prefer aggregating outputs via channel-wise concatenation with *cell0* also relying on dense attention, and *cell3* – on affine transformation with bilinear sampling. In addition, *cell1* and *cell2* employ 3D convolution in order to capture information between various inputs.

5. Discussion & Conclusions

It is still an open question of what is the optimal way of propagating and extracting information across video frames. While a straightforward solution involving the optical flow allows to achieve solid results, it possesses several disadvantages that stem from the limitations of the optical flow itself and ultimately limit the ability of the network to adapt to novel frames. Furthermore, computations involving the optical flow cause a significant overhead, prohibiting the final system from being deployed in real-time.

In this work, instead of manually enhancing static segmentation networks with dynamic components, we proposed an automatic approach based on neural architecture search methods. Such automation have multiple benefits as it explores a large pool of networks and finds best-performing ones on the given dataset. In a broader sense, starting from a static per-frame segmentation network, we showcased a way of generalising existing solutions without any reliance on the optical flow. More importantly, all previous solutions in semantic video segmentation tend to add new temporal blocks that improve accuracy but deteriorate latency (or vice versa) – in contrast, the designed search space leads us to models that are better (Tables 3, 4) and faster (Table 5) than the respective baseline. In particular, we extended the static baseline with a dynamic cell, the design of which is automatically discovered with the help of reinforcement learning. The best discovered cells improve the baseline by more than 1% at the same time leading to significant memory and latency savings.

Limitations

While the proposed methodology relies on the static baseline, we expect that omitting that requirement and searching for a video segmentation network end-to-end would further boost the results. Another limitation worth mentioning is that at present we do not account for a difference between the pre-classifier output of the dynamic cell and that of the static one (which serves as the input to the dynamic cell on the next step). We believe that adding an appropriate regularisation term (akin to knowledge distillation [12]) would lead to even better results. Furthermore, as shown in the CamVid experiments on 1Hz annotations, the temporal connections might either be not even needed or might need to be extended further in time, for example, requiring an external memory storage.

[3] Test results: https://bit.ly/2FrZ8jM

[4] Test results: https://bit.ly/2HyoVcb

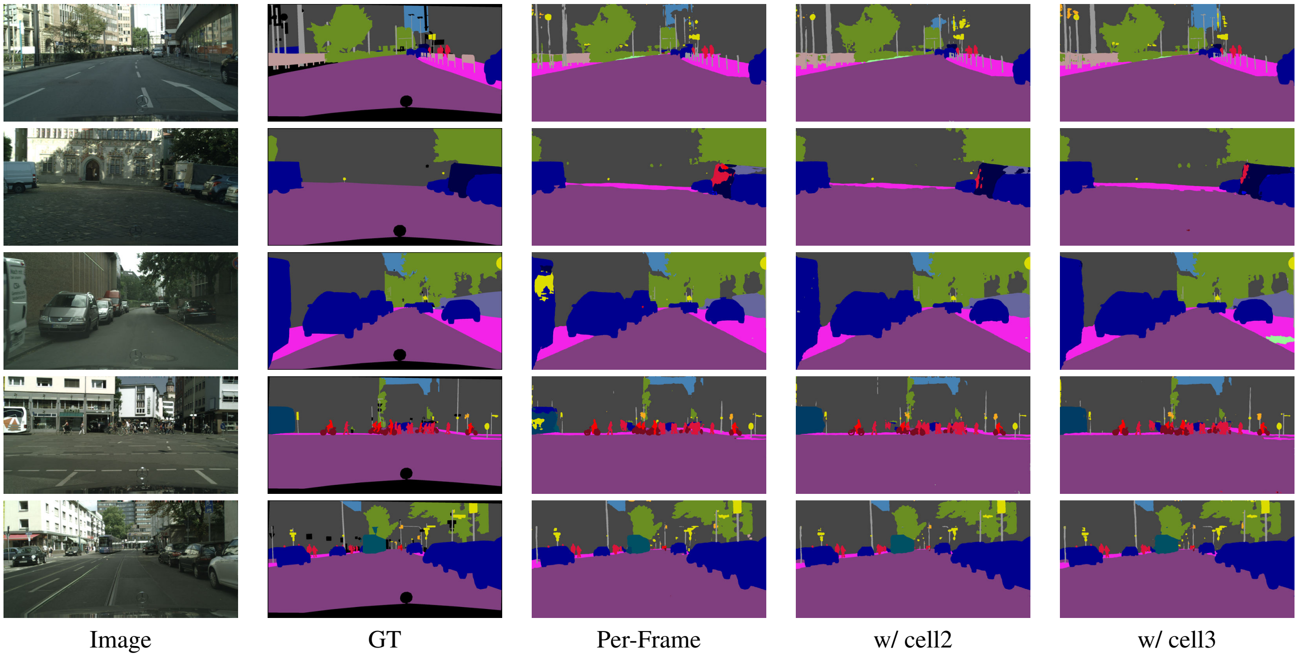


Figure 5. Inference results on the validation set of CityScapes.

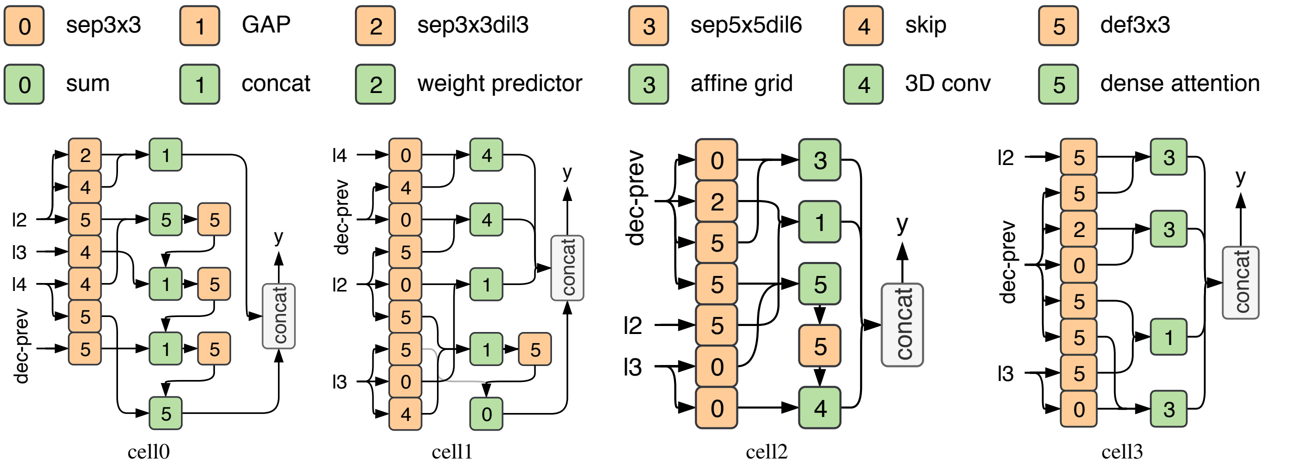


Figure 6. Visualisation of the discovered cells. Orange blocks represent operations and green blocks represent aggregation operations. Numbers inside blocks are operation identifiers as in Tables 1 and 2.

Acknowledgements

VN, CS, IR's participation in this work were in part supported by ARC Centre of Excellence for Robotic Vision. The authors are also grateful to the anonymous reviewers for their comments, and the HPC cluster Phoenix at the University of Adelaide for providing computational resources.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017.

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[3] S. Chandra, C. Couprie, and I. Kokkinos. Deep spatio-temporal random fields for efficient video segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[4] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[5] L. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. *Proc. Advances in Neural Inf. Process. Syst.*, 2018.

[6] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[7] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[9] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vision*, 2010.

[10] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016.

[12] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *Proc. Advances in Neural Inf. Process. Syst.*, 2014.

[13] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2015.

[14] S. Jain, X. Wang, and J. Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. *arXiv: Comp. Res. Repository*, abs/1807.06667, 2018.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv: Comp. Res. Repository*, abs/1412.6980, 2014.

[16] Y. Li, J. Shi, and D. Lin. Low-latency video semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[17] G. Lin, A. Milan, C. Shen, and I. D. Reid. RefineNet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[18] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[20] V. Nekrasov, H. Chen, C. Shen, and I. D. Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019.

[21] V. Nekrasov, C. Shen, and I. D. Reid. Light-weight refinenet for real-time semantic segmentation. In *Proc. British Machine Vis. Conf.*, 2018.

[22] D. Nilsson and C. Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[23] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *Proc. Int. Conf. Mach. Learn.*, 2018.

[24] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv: Comp. Res. Repository*, abs/1707.06347, 2017.

[26] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2016.

[27] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: stochastic neural architecture search. *arXiv: Comp. Res. Repository*, abs/1812.09926, 2018.

[28] Y. Xu, T. Fu, H. Yang, and C. Lee. Dynamic video segmentation network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.

[29] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *Proc. Int. Conf. Learn. Representations*, 2016.

[30] F. Yu, V. Koltun, and T. A. Funkhouser. Dilated residual networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[31] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proc. Eur. Conf. Comp. Vis.*, 2018.

[32] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[33] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. *arXiv: Comp. Res. Repository*, abs/1811.11168, 2018.

[34] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017.

[35] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *Proc. Int. Conf. Learn. Representations*, 2017.

[36] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.