

Survey on fast dense video segmentation techniques

Quentin Monnier^{a,*}, Tania Pouli^a, Kidiyo Kpalma^b

^a b-com, 1219 Av. des Champs Blancs, Cesson-Sevigne 35510, France

^b Vaader, IETR, Bat 11D263 Av General Leclerc, Rennes 35700, France

ARTICLE INFO

Communicated by Sifei Liu

MSC:
41A05
41A10
65D05
65D17

Keywords:
Video semantic segmentation
Matting
Deep learning

ABSTRACT

Semantic segmentation aims at classifying image pixels according to given categories. Deep learning approaches have proven to be very effective for this task. However, extensions to video content are more challenging, typically requiring more complex architectures, given the temporal constraints and the additional data that video introduces. At the same time, video application tend to necessitate real-time, or at least interactive performances: self-driving cars, industrial applications, or live broadcasting to name a few, imposing even stronger constraints to video methods. In recent years, considerable efforts have been made in addressing these somewhat opposing challenges. In this survey, we explore the solutions proposed to improve the quality and accuracy of video segmentation, as well as the different techniques that can be employed to improve the efficiency of such approaches, in particular in terms of inference time. Finally, we briefly describe the datasets related to the semantic video segmentation task and the challenges involved.

1. Introduction

Segmentation in general terms is the process of separating an image into several sets of pixels that have common image features such as color, texture, location, or semantic meaning. This process is a cornerstone of image processing since it provides a simplified version of the image content that can be used in further processing (Subramaniam et al., 2022). In recent years, significant improvements have been made in this field thanks to new deep learning techniques (Hao et al., 2020). However, compared to single image segmentation, segmentation applied to video is still very challenging due to the temporal dimension and the constraints it introduces.

The main challenge brought by video content is that of temporal consistency. The human visual system is particularly sensitive to temporal changes (Borghuis et al., 2019). As such, if the result of the segmentation serves for processing that will be visually observed, inconsistencies over time in the segmentation process must be avoided. To that end, objects need to be accurately tracked over time, handling their motion, occlusions or even objects potentially leaving the frame. Aside from tracking accuracy challenges, video contents also tend to contain a lot of motion blur which complicates accurate segmentation as object boundaries become harder to delineate.

Video sequences contain significantly more data than a single image. The consequences are twofold. First, while image segmentation often relies on complex neural architectures with many parameters to achieve high quality results, adapting these architectures to video content directly would likely lead to networks that would be too

large to train and too time-consuming to use in practical inference scenarios (Mahadevan et al., 2020). Further, in many applications, videos not only contain more data than images but are also compressed using lossy compression methods that can introduce a great variety of artifacts (Unterwiesing, 2012; Lin et al., 2020), which need to be taken into account.

The above reasons show that video segmentation is a complex task, but many applications related to video processing and segmentation in particular require real time performances, without loss of quality: live TV, security applications, self-driving cars to name a few. Considering this set of challenges, in this survey, we mainly focus on two different aspects of video segmentation techniques: quality and efficiency.

The quality criterion can be understood as the ability of a given method to accurately segment videos relative to a ground truth, and can be evaluated subjectively or objectively. Intersection-over-Union is the most commonly used metric in the context of semantic video segmentation, but depending on the goal, other metrics can also be used, namely precision, recall, pixel accuracy, Dice coefficient, and metrics that consider temporal consistency (Varghese et al., 2020; Zhang et al., 2022).

In this paper, the efficiency criterion refers to how fast a given method is able to produce segmentation maps. It can be assessed by measuring the FLOPS (number of basic operations) required to segment a frame, but this metric is not considered as reliable (Ma et al., 2018). Therefore, the speed of a model is usually measured by the number of frames the model can segment in one second (FPS) on a reference platform.

* Corresponding author.

E-mail address: quentin.monnier@b-com.com (Q. Monnier).

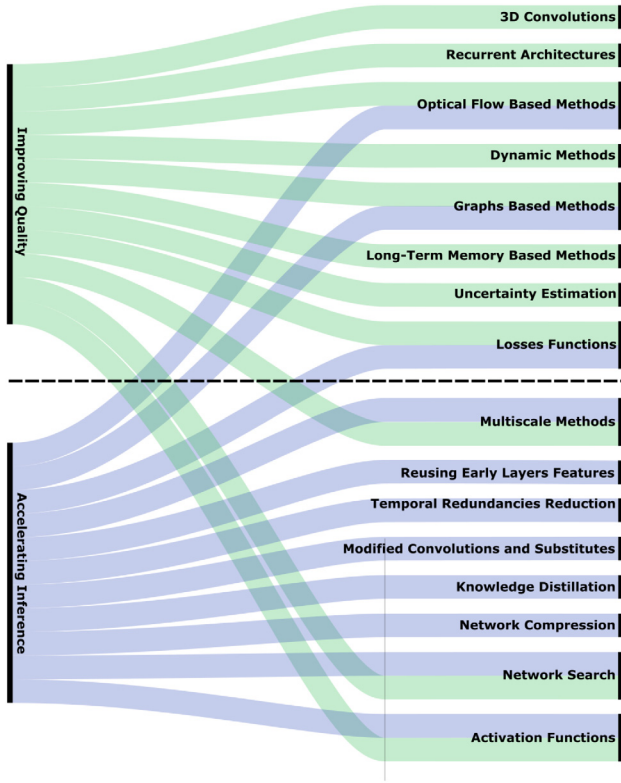


Fig. 1. Diagram showing the two main sections of this paper. The topics covered in the subsections are sometimes related to both main parts to some extent, so in this case they are attached to the main part they cover most.

To provide a basis for the discussion of video segmentation works, Section 3 presents key methods in the context of single-image segmentation. Then, state-of-the-art methods in relation to the two main challenges in semantic video segmentation are presented, as can be seen in Fig. 1. More specifically, approaches focusing on quality are discussed in Section 4, while methods and techniques for improving efficiency are presented in Section 5. Section 6 describes the datasets than can be used for semantic video segmentation, as well as the challenges raised by such data. Finally, the various methods, challenges, and solutions mentioned throughout the survey are summarized and compared in Section 7. It should be noted that the goal of this survey is not to provide an exhaustive list of all video segmentation approaches, nor a detailed benchmark of the state-of-the-art methods (such as Zhou et al., 2023), but rather to explore concepts and ideas from the scientific literature (related to video segmentation, but also from other domains) that can or could be used to improve the performance of semantic segmentation methods, either in terms of the quality of their results or the speed of execution. Since, most real-world applications would need to achieve both of these goals, the approaches described here provide us with useful hints and a basis on which we can build upon in subsequent work towards practical video segmentation methods.

2. Background

In this section, we first formalize the problem and state its links with relative fields. Then we briefly describe its historical background.

2.1. Problem taxonomy

The dense segmentation task applied to images consists of seeking a function that ideally maps any input image to a map of identical height and width, containing pixel-wise simplified information about the input content. The dense segmentation applied to videos follows the same

principle in the sense that, ideally, each frame of each input video must be mapped to a map of identical height and width which contains pixel-wise simplified information about the input content. The nature of the information highlighted in the outputs creates several categories of segmentation:

- **Video Semantic Segmentation** refers to the task of assigning each pixel to a semantic label from a list of predetermined classes. The output of such methods contains a class label for every pixel of the input video. Such methods tend to learn visual representations of the classes they encounter.
- **Video Object Segmentation** or **Video Instance Segmentation** refers to methods whose goal is to detect, segment, separate and track objects that stand out from the background. Such methods do not focus on the semantic meaning of the objects depicted in the input videos but instead focus on their interactions to detect and separate the objects that stand out from the background. To be clear, when multiple objects of same nature are present in the foreground of a scene, VOS methods must assign a unique label to each of these objects. Methods that, instead of separating the prominent objects, simply assign a binary label indicating whether each pixel belongs to the foreground or the background, are called **Video Background/Foreground Segmentation**. It is also interesting to note that “Object Segmentation” in the images domain often refers to tasks that are very similar to “Panoptic Segmentation” and can therefore be easily confused.
- **Video Panoptic Segmentation** is the combination of “Video Semantic Segmentation” and “Video Object Segmentation” in the sense that the goal is to associate each pixel with a semantic label as well as an instance/object index.
- **Matting** differs from classical segmentation tasks in the sense that it considers that each pixel does not necessarily map to a single class but rather to a mixture of classes. Thus, the output map contains alpha values that determine the proportions of the classes for each pixel of the input.

Video segmentation methods can be further categorized based on the amount of human interaction used during the inference:

- **Interactive Video Segmentation** methods are guided by live human hints on how to perform the current segmentation and how to refine it.
- **Semi-interactive Video Segmentation** or **Semi-automatic Video Segmentation** methods only require human input at the beginning of a video to process the entire sequence. The initial cues can be of several types. They can be a textual description of the objects to be segmented, or they can indicate the initial object locations using bounding boxes, a segmentation map, or a sketch of the first frame.
- **Automatic Video Segmentation** methods do not require any human interaction to perform the video segmentation task.

As with other deep learning problems, video segmentation methods can have differences in learning strategy and, more specifically, in the amount of supervision needed to train the method:

- **Supervised** training requires dense ground truth annotations, but makes training easier for most video segmentation methods.
- **Weakly supervised** training also requires ground truth annotations. However, the annotations are sparser and thus much easier to obtain (typically bounding boxes or frame-level labels of the objects present in the frame), but also make the training more challenging and thus only suitable for some methods that are carefully designed for it.
- **Unsupervised** training is dedicated only to methods that are mostly self-supervised, i.e. they generate pseudo-labels based on prior knowledge about the properties of the objects to be distinguished. Thus, there is no need to label any data for these methods.

As we have just described, there is a wide variety of video segmentation methods. Some categories of methods do not fulfill the same goal (e.g., object versus semantic segmentation) or do not have access to the same input information (e.g., interactive versus automatic segmentation). The work of Zhou et al. (2023) provides a complete comparison of the state-of-the-art methods for each category. Our work focuses on another aspect of this diversity: despite important differences, methods developed for distinct segmentation purposes can also share similarities and compatibilities. Thus, a mechanism developed for a specific purpose can sometimes be applied to other segmentation applications and help improve the performance of that other domain. In this work we therefore review innovative methods in various segmentation domains and discuss how they can or could be used to make progress in the specific context of Automatic Video Semantic Segmentation.

2.2. History

Semantic segmentation has been a subject of research since the 1970s. The first works on the subject used basic image operations (Csurka et al., 2023) and focused mainly on two priors: pixels belonging to the same class tend to be similar, while pixels belonging to different classes tend to be different. Thresholding methods (Otsu, 1979; Nock and Nielsen, 2004; Dhanachandra et al., 2015) and edge-based methods (Canny, 1986; Kass et al., 1988) highlight the early use of these two priors which were also combined in other methods using graphs (Boykov et al., 2001; Hochbaum, 2001; Plath et al., 2009). These principles have also been applied to video content (Xu and Corso, 2012; Chang et al., 2013).

The application of segmentation to video also opened up the possibility of using motion as a prior, based on the fact that different objects in the scene may not move in the same way (occlusions, parallax due to camera motion, intrinsic motion), creating temporal edges, and that some motion is typical for certain classes of objects (Brox and Malik, 2010; Fragkiadaki et al., 2012; Ochs et al., 2014). However, these cues only worked for semantic segmentation tasks dealing with simple classes with very distinctive colors or motions, limiting their applications.

To extend semantic segmentation to more complex tasks, some methods started to use shape and texture priors thanks to image descriptors (Lowe, 2004; Schroff et al., 2006; Shotton et al., 2006), statistical algorithms, and early machine learning methods (Lowe, 2004; Schroff et al., 2008; Yu et al., 2011). These techniques were also applicable to video semantic segmentation methods (Jain et al., 2013; Liu and He, 2015), which were computationally intensive, and required selecting the right features depending on the classes at hand to maintain effectiveness.

More recently, the field of semantic segmentation has evolved tremendously with the advent of deep learning models (Long et al., 2015) that replace hand-crafted features with automatic ones. The following work will presents some of the developments that have occurred since then and that can be exploited to create better methods for video semantic segmentation.

2.3. Related research areas

Video semantic segmentation can be used as a cornerstone for many image and video processing techniques such as depth estimation (Wang and Piao, 2023), super-resolution (Aakerberg et al., 2022), and colorization (Xu and Ding, 2021). It can also have more direct applications such as automated diagnosis in the medical field (Krithika alias AnbuDevi and Suganthi, 2022), precise analysis of satellite data (Nepune et al., 2021), or industrial applications such as autonomous driving (Siam et al., 2018) or waste sorting (Wang et al., 2020).

3. Image-based methods

Considering the performance achievements of recent image segmentation methods (Tao et al., 2020; Wang et al., 2022b,a), one

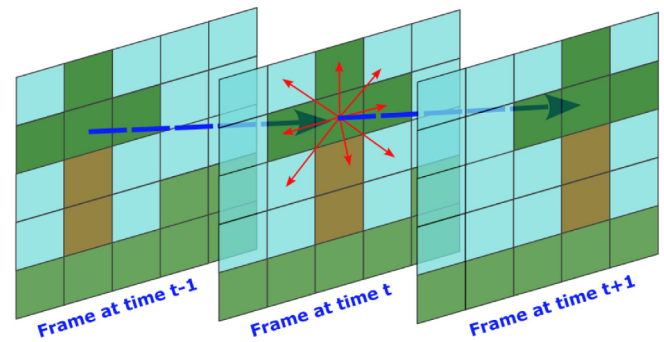


Fig. 2. The spatial neighborhood of a single natural image contains spatial information and redundancy as depicted by the red arrows. This inductive bias is what motivates the design of convolutional neural networks. In natural video sequences, the same can be said about the temporal dimension. However, this information and redundancy cannot be taken into account by frame-wise methods, which is suboptimal both in terms of quality and inference time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

could naively consider applying these same techniques to videos by processing each frame independently. However, both the volume and type of data encountered in video makes a direct application of image-based approaches not straightforward. Yet, a subset of these techniques focuses on improving inference time to make them real-time and therefore applicable to pipelines like self-driving cars. At the time of writing this survey, one of the ways to achieve a good trade-off between quality and inference time is to use parallel branches (Hong et al., 2021), where one branch focuses on rapidly obtaining a global and meaningful context while the other branch gathers low-level details.

An alternative approach can be used by respectively linking the coarse semantics and fine details of those architectures to the proportional (P) and integral (I) parts of controllers used in many industrial control systems (Xu et al., 2022). The authors claim that dual networks face a similar issue as PI controllers, called “overshoot”, which in this context consists of the detailed branch being overwhelmed by the overspread semantic context. Therefore, they propose to add a third branch playing a derivative role to mimic the behavior of a PID controller and tackle the “overshoot” issue.

Other 2D methods (Gao, 2021; Peng et al., 2022) also gather multiscale features by using parallel branches with various strides and dilation coefficients and fuse them using pyramid pooling modules (Zhao et al., 2017). Reusing early features to guide the upsampling of deep semantic features is frequently employed since the introduction of U-net (Ronneberger et al., 2015) and can be enhanced by attention modules (Peng et al., 2022; Xu et al., 2022). To achieve the same quality of results as larger models, but with a more compact architecture, some strategies rely on pruning (Chen et al., 2022) (see Section 5.6.2) or knowledge distillation (Liu et al., 2019; He et al., 2019) (see Section 5.5).

Overall, recent 2D semantic segmentation techniques achieve great results, both in terms of quality and inference time. Yet, they are not suitable for many video-related pipelines because of their lack of temporal consistency. This is particularly problematic when a visual output is produced and further used (e.g. movie colorization, SDR to HDR conversion, style transfer) or in robotics when sudden and large changes can lead to undesirable actions.

Furthermore, as highlighted in Fig. 2, restricting the analysis to the spatial domain while dealing with highly redundant spatiotemporal data is not necessarily the most efficient strategy. Aside for guaranteeing temporal consistency, temporal links between frames can be used in two ways. Either to get a wider context to improve quality, or to get rid of the temporally redundant part of the spatial analysis to accelerate the inference.

4. Improving segmentation quality

State-of-the-art image segmentation methods achieve impressive results for many cases. However, there is still room for improvement when the scenes to be segmented are more challenging, or when considering temporal consistency. Indeed, frame-wise methods have no awareness of the temporal context of the surrounding frames, so they cannot know if their current result is consistent with the surrounding ones, and as such cannot retrieve contextual information from other frames to guide the segmentation result. In the following section, we will focus on methods that take advantage of the video format, and in particular the temporal dimension available, to improve segmentation quality.

4.1. 3D convolutions

To address the temporal inconsistencies of 2D semantic segmentation techniques when applied to video, earlier approaches treated the temporal dimension in the same manner as the two spatial dimensions (Mahadevan et al., 2020). In medical imaging, data covering the three spatial dimensions is common and is processed using 3D convolutions. So pioneering works on semantic video segmentation replaced classic 2D convolutions from existing architectures with 3D convolutions to process the temporal dimension alongside the spatial ones (Ji et al., 2010; Karpathy et al., 2014; Tran et al., 2015; Varol et al., 2018).

Although the increase of the number of parameters due to the transition from 2D to 3D was manageable for the relatively small networks from that time, the consequences in terms of computational cost and training difficulty are more significant when the same transition is applied to more modern and more complex architectures. Even if the amount of video segmentation data is now higher and attenuates the training difficulty problem (Hara et al., 2018), the computational cost remains an issue.

Features extracted from deep layers are of much lower resolution than those of shallow layers, yet contain semantics that are very influential on the final segmentation decision. Thus, to keep a reasonable computational cost, some works propose to keep 2D convolutions in all layers except the ones close to the output, therefore only considering the temporal dimension in one part of the network (Xie et al., 2018; Athar et al., 2020; Grammatikopoulou et al., 2023).

Certain computational and mathematical “tricks”, described in Section 5.4, have made it possible to employ 3D convolutions throughout the complete network. In Hou et al. (2019a) take advantage of that possibility and design a fully 3D convolutional network made of an encoder deeply connected to a decoder by parallel convolutions of different strides, and by convolutional connections between early layers from the encoder to deep layers from the decoder.

Nevertheless, considering the temporal dimension simply as an addition to spatial information comes with certain caveats. First, making no distinction between the computation of the spatial and temporal dimensions is not necessarily a good idea since their nature is very different. Secondly, while dense (i.e.: without stride) convolutions are required to detect spatial details and subtle temporal movements and deformations, they also introduce a lot of redundancy in the computation since the sliding windows overlap a lot. As a network’s spatial receptive field is limited by its depth and the size/stride/dilation of its convolution kernels, the network’s temporal memory is also bounded by those factors (Long et al., 2022).

The distribution of valuable information along the temporal dimension of a video can vary greatly depending on criteria such as camera movement, scene complexity, and the movement of individual objects within the scene. In the following section, we will focus on methods that can adapt to this diversity.

4.2. Recurrent architectures

One way to incorporate temporal consistency into a network without significantly modifying its architecture is to connect the output of the network to a module that can handle sequential data. In deep learning, Recurrent Neural Networks (RNN) are designed to handle such data (Sherstinsky, 2020). Each RNN unit takes a vector as input and outputs a nonlinear transformation of its current hidden state. The current hidden state of the cell results from the nonlinear combination of the previous hidden state and the current input. As each output depends on the previous one, this mechanism creates a sort of dynamic memory that can process temporal information.

However, capturing long-term dependencies with RNN is not efficient because of the vanishing gradient problem (Bengio et al., 1994). For example, when a recurrent unit reaches a stable state for several iterations, the previous state that is responsible for this situation has faded away. Gated architectures were therefore introduced to better control which information should be kept or forgotten at each time step. One historically popular architecture is the Long Short Term Memory (LSTM) unit (Vennerød et al., 2021) that uses tree gates that respectively control for each step what should be forgotten, what should be memorized from the current input, and in what proportion the current hidden state and the current input should be mixed to create the output. More recent work shows that it is possible to achieve similar results without regulating the output with a gate (Chung et al., 2014). Gated Recurrent Units (GRU) (Cho et al., 2014) hence use a lighter architecture that is less computationally expensive and that has a lower memory footprint. Gating units can be used at the output of a 2D CNN backbone to create spatiotemporal features that are then further processed by a CNN and up-sampled to match the input’s dimensions (Fayyaz et al., 2017; Siam et al., 2017). The work from Song et al. (2018) goes a step further by using a multiscale approach (see Section 5.1) in both the spatial CNN and the temporal unit, and by refining the result with a conditional random field (see Section 4.5). Adding a temporal unit at the end of deep Fully Convolutional Networks (FCN) has, among others, the advantage of optimizing computational load by using spatially reduced features. Despite that, classic GRU and LSTM units that work with vectors passing through gates made of fully connected layers of neurons, are still not suited to the spatial dimensions inherent to videos (Siam et al., 2017).

To better represent the inductive bias of local spatial connectivity of natural images and to avoid the parameter waste induced by fully connected layers dealing with such data, convLSTMs and convGRUs (Fig. 3), were introduced. They consist of replacing each fully connected layer from the inner gates by convolutional layers (Shi et al., 2015; Ballas et al., 2016). These convolutional gates can be used for semantic segmentation (Siam et al., 2017; Song et al., 2018). They have the advantage of not necessarily requiring fine-tuning of the 2D backbone to produce temporally coherent results. On the other hand, producing temporal features only at a highly semantic level do not fully account for the richness of temporal cues provided by video content.

The aforementioned issue is taken into account by a similar architecture using a parallel 3D fully convolutional network that considers a few frames at a time to generate spatiotemporal features with short-term dependencies (Qiu et al., 2018). The features generated in this way are then fused with the long-term features given by the convLSTM that follows the 2D backbone. However, this method introduces a lot of redundancy in the computation by processing each frame several times. A similar strategy consists of providing not only the 2D segmentation of the current frame to the recurrent unit but also the previous one warped to the current geometry thanks to optical flow estimation (Nilsson and Sminchisescu, 2018). The use of optical flow help to provide more precise temporal clues but has no influence on the creation of the semantics in this method.

To account for temporal cues at both low and high levels, Tokmakov et al. (2017) suggest creating an architecture consisting of a classical

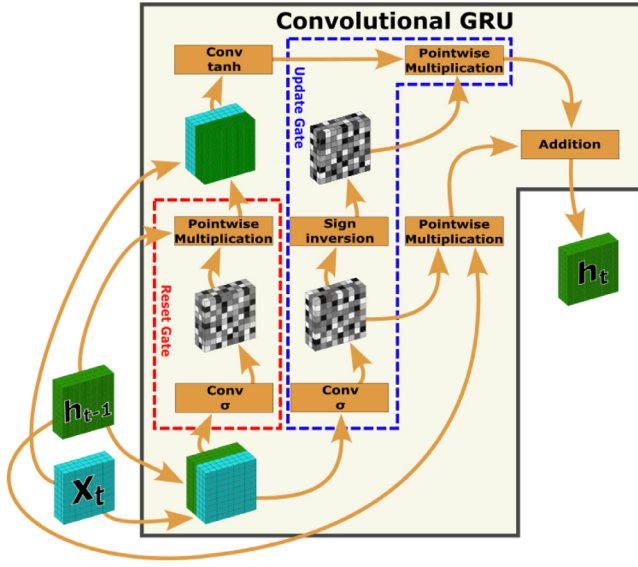


Fig. 3. A diagram representing the convolutional GRU architecture. In the reset gate, the unit decides which information from the past (stored in the last hidden state, namely h_{t-1}) should be discarded to produce the new hidden state candidate by being concatenated with the input value x_t . The update gate then produces the new output (h_t) by pondering between the new information from the candidate and the historical information from the previous hidden state.

CNN connected to stacked convGRU units, so that each convGRU takes as input the feature map produced by the same level CNN layer as well as the output of the shallower convGRU unit. In this way, each layer produces spatiotemporal features at its scale. However, to the best of our knowledge, this idea has not yet been applied to semantic video segmentation. The reason for this may be that, since each convGRU has a significant computational cost, the quality gain of spatiotemporal features provided by adding one in each layer may be a poor trade-off. Another drawback of gating mechanisms and RNNs is that they process data sequentially and therefore cannot be parallelized. Finally, even if gating mechanisms can retain information better than simple RNNs, the problem of long-range information loss is inherent to sequential processing, especially if the processed data change a lot over time. This last drawback can be mitigated by using multiple units working on the same sequence in reverse order, but as noted above, this comes at the cost of increased computation time without addressing the core problem of long-range information decay.

4.3. Optical flow based methods

Optical flow is the result of estimating the apparent motion of pixels describing a scene over time (Zhai et al., 2021). The estimated motion is relative to the scene and to the observer (or camera). This feature can be used in several ways to improve the quality of semantic video segmentation models. Optical flow can be computed using rule-based methods, but to ensure that their method can be fine-tuned in an end-to-end fashion, many methods use deep learning based optical flow in their architecture (Dosovitskiy et al., 2015; Nilsson and Sminchisescu, 2018; Liu et al., 2017; Ilg et al., 2017; Gadde et al., 2017; Xu et al., 2018; Li et al., 2018b; Ding et al., 2020).

4.3.1. Optical flow for warped features aggregation

The first way that optical flow can be used to improve quality is by aggregating the spatial context of multiple frames. The idea is that the content of the frames changes over time, so that an object whose color or texture is difficult to distinguish from its background at one time may have different lighting conditions or appear against a more contrasting background at another time. By using optical flow estimation to warp

features from different times into a common reference frame, one can group together those complementary features that were previously scattered across time. This idea is illustrated Fig. 4a.

There is a 2D CNN architecture in which multiple layers receive warped features from the same layers at the previous time step (Gadde et al., 2017). The warping is obtained through a unique optical flow estimation between the current frame and the previous frame. However, most of the time, consecutive frames are very similar and contain variations that are too small to provide different contexts. Frames that are further away from the current one may have changed more and therefore be more interesting to gather various contexts. On the other hand, optical flow is likely to be more accurate between adjacent frames, and thus more similar frames, than between frames spaced further apart in time. To pick the best of the two, Gadde et al. (2017) first compute a rough segmentation map for the current frame, then estimate the optical flow between the current frame and both the previous frame and a much earlier frame. In this way, the current segmentation map can benefit from a precise but not very distinctive segmentation map from the previous frame and an imprecise but distinctive segmentation map from the much earlier frame. The optical flow results are also used to compute “confidence score” maps, which can then weight the contribution of the corresponding distorted segmentation maps to the current one.

The combination of different frames is further exploited by Li et al. (2018b), which combines optical flow-based warping with LSTM cell processing, as described in Section 4.2. Nilsson and Sminchisescu (2018) use a similar process where previous segmentation maps first pass through a convGRU before being warped to the current frame space thanks to optical flow estimation on the corresponding frames. The warped segmentation frames are then further processed by passing through a second GRU to obtain the final result.

The fact that optical flow can be computed separately from the segmentation architectures allows to use it as a post-processing step for classical 2D architectures. This is emphasized by a method that can make any 2D segmentation model temporally consistent as long as the model produces a class probability distribution as output (Miksik et al., 2013). To do this, they first use optical flow to associate each pixel of the previous frame with the pixels of the current frame. Then, for each pixel of the current frame, they perform a weighted average between itself and the neighboring pixels of its match from the previous frame. The weight coefficients are determined by a learned similarity metric that takes low-level features such as RGB data, gradients, and Local Binary Pattern (LPB) as input.

In all of these methods, optical flow is used as a tool to align spatial contexts from different times but it is never used as a feature itself. Motion, however, can be very effective in distinguishing objects. When the viewer moves, objects of different depths move differently on the screen due to the parallax effect. Independent of camera movement, many objects can make their own movements and thus provide clues as to their nature. It is thus clear that the methods described so far do not exploit the full potential of optical flow. In the next section, we will explore the approaches that use optical flow as a feature.

4.3.2. Optical flow as a feature

Optical flow used as a feature (Fig. 4b) is mostly used by object/saliency segmentation methods. This is because these methods have fewer available assumptions about the shape of the objects they are segmenting and therefore require more resources to infer object boundaries, and also because atypical motion is often correlated with saliency. Many methods compute spatial features and optical flow separately, then combine them to create superpixels or saliency maps in which strong edges (in the spatial or temporal domain) often delineate object boundaries (Fang et al., 2013; Wang et al., 2015b,a; Liu et al., 2016; Koh and Kim, 2017). Temporal cues are then further exploited by adding temporal consistency constraints to the energy functions that are minimized to produce segmentation outputs. Most of them do not

rely on convolutional neural networks to work. However, Jain et al. (2017) compute optical flow and spatial features of the current frame in separate CNN paths to produce saliency prediction maps, which are then multiplied element-wise to produce a third prediction map that takes into account both flows. The final output for each pixel is the maximum of the three prediction maps.

In the preceding paragraph, we have seen that optical flow can be used in object segmentation to compute temporal edges, thereby enhancing the model's ability to create precise boundaries of objects whose implicit shape is unknown to the model. We have also seen that strong or atypical object motion can also be used to detect saliency locations, again for object segmentation. Based on this knowledge, one can understand that these atypical movements can also be used in the context of semantic segmentation as a cue to improve semantic understanding as well as the precision of object boundaries thanks to the motion edges (Saleh et al., 2017). The way to improve boundaries thanks to motion edges starts by using a pre-trained 2D CNN classifier to localize the approximate positions of objects. While the classifiers are not trained to localize objects in a scene, the output of the convolutional backbone (before the densely connected layers) contains highly semantic features that are roughly localized thanks to the inherent structure of CNNs. By designing a deep temporal branch that takes multiple optical flow frames as input, one can obtain the more precise boundary information lacked with the classifier output. Both rough semantics and precise boundary information are then mixed in a sub-convolutional network to produce a semantic segmentation map.

However, optical flow estimation is not a trivial task. There are several methods that try to solve this problem (Tu et al., 2019). Some are based on deep learning (Dosovitskiy et al., 2015; Ilg et al., 2017) and some are not (Memin and Perez, 1998; Brox and Malik, 2011), some are fast (Kong et al., 2021; Young et al., 2020) and some focus more on the quality of the estimations (Huang et al., 2022a; Jiang et al., 2021). Although precise estimations are not necessarily crucial when optical flow is used to improve semantic features that are already coarse, using the estimation to extract precise temporal edges imposes stronger constraints. As such, it is important to choose wisely how to balance the trade-off between precision and complexity depending on the task for which optical flow is used. In addition, even using complex estimation methods is not always enough to obtain very sharp results in some conditions. In fact, aside from the need for textures for the methods to work, such approaches are also inefficient when the scene undergoes large movements, occlusions, and disocclusions (Hu et al., 2020). Finally, one can easily understand that most of the movements of objects are related to their nature. Thus, regular optical flow estimation methods are hampered by their inability to understand scene semantics. In the next section, we will describe methods that overcome this problem.

4.3.3. Joint computation of spatiotemporal features

The methods described thus far either use optical flow to warp features or as a feature itself, and compute it separately from spatial features. The previous section highlighted the fact that optical flow can benefit semantic segmentation at both high and low levels, showing that the two different tasks are in some ways linked. Nevertheless, obtaining an accurate optical flow estimate without significant computational overhead is challenging. Those two facts raise the question whether the optical flow estimation could benefit from the semantic cues given by the segmentation results.

Following this idea, Sevilla-Lara et al. (2016) perform an initial 2D segmentation whose result is then further reduced to three classes of different motion properties: objects that can move independently, objects that are mostly still, uniform, planar, and in the background, and lastly, objects that do not move, but whose shape and texture create complex parallax. The motion cues generated in this way can then be used to guide the computation of an accurate optical flow estimation. Finally, the enhanced optical flow provides the ability to correct errors

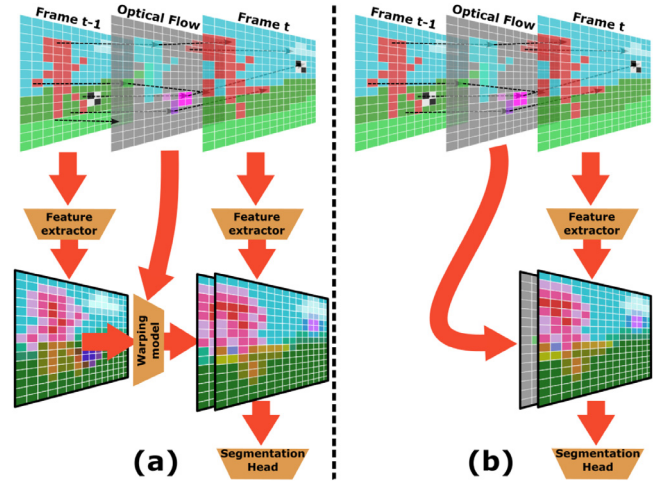


Fig. 4. Optical flow can be used to warp features from different times into a common space to aggregate different contextual information, as in (a). But it can also be used directly as a feature, as in (b), because its motion information is semantically rich and its temporal edges can be complementary to the spatial ones.

and inaccuracies in the initial 2D segmentation to produce better results.

Hur and Roth (2016) define each scene as a three-dimensional space in which there is a set of plans subject to epipolar constraint (principle according to which each line formed by the 3D point of an object and its corresponding point on the projection screen must converge to the focal point of the camera) conditioned by semantics, occlusions, and temporal consistency. At each time step, the current segmentation and flow maps are computed jointly based on the current frame, the previous frame, and the previous refined segmentation map. occlusion detection prevents occluded regions from influencing the semantics. The semantics themselves are used to weight the epipolar constraint to make it more stringent for still objects.

Another way to ensure that the segmentation and optical flow estimation tasks benefit from each other is to create feature interactions at different levels of both models (Ding et al., 2020). In this work, an encoder takes two consecutive frames as input to produce common features, which are then further processed in a first decoder branch that estimates the optical flow and occlusion maps, and a second decoder branch that produces segmentation maps. The two branches interact at different scales through two kinds of modules that take features from the same depth layers of both branches as input (i.e., segmentation, occlusion, and optical flow candidates). The first kind of module is one that computes temporal consistency and returns its output to the segmentation branch, while the goal of the second kind of module is to help the optical flow branch handle occlusions.

We have seen that two different tasks can benefit from each other by alternately refining one task with the result of the other, or by sharing features during the one-pass inference of both tasks. Even with improvements, the usefulness of optical flow estimation for video segmentation remains limited to modeling short to medium-term dependencies, but is not well suited for modeling long-term dependencies (Yang et al., 2019). The reason is that optical flow estimation boils down to finding associations between frames. however, finding associations between distant frames is difficult because much may have changed in that time, and propagating the estimated motion of successive frames over a long period of time accumulates inaccuracies and increases the risk of error.

4.4. Dynamic methods

Traditional semantic segmentation techniques rely heavily on convolutional layers. Each convolution kernel is able to detect how much

each region of the input matches a particular simple pattern. By adding more kernels to a layer, it enables it to recognize different patterns. Thus, by stacking one convolution layer on top of another, the deeper layer can learn to detect patterns in the combined results of the previous layer. The combination of the two can itself be seen as a pattern detector of a larger scale and higher level of abstraction, and so on as one increases the number of stacked layers. Individually, each kernel is just a small regular lattice in which weights can be learned. Therefore, convolutional neural networks are quite easy to train. Nevertheless, they are not without some drawbacks.

The patterns in each convolutional kernel are fixed at inference time, which means that to be robust to object deformations, rotations, or changes in perspective, multiple kernels from the same layer must encode slightly different versions of the same pattern. Using many kernels makes the model heavy and difficult to train. Some robustness can be introduced against small transformations by using handcrafted modules such as pooling layers between convolutional ones. However, this comes at the cost of reduced resolution in the produced features. Further, the abstraction is built by stacking layers. However, due to the regular structure of the convolution kernels, the field of view of a model grows slowly with increasing depth. To capture larger and more complex object representations, the model must be very deep, which makes it slow and heavy. This problem can be mitigated by using strides or dilation rates, but just like pooling, adding those techniques reduces resolution and can cause the loss of important details. Finally, each convolution kernel must process the entire input, even if at the given depth it is not necessarily useful to apply the kernel to some parts of the input feature. In conclusion, to handle various situations, regular convolutional neural networks must have many parameters, which makes them heavy and slow, and they must sacrifice some of their resolution while processing data.

To make 2D CNNs more robust to transformations, Jaderberg et al. (2015) suggest adding a dynamic warping module between convolution layers. This module takes a feature as input and processes it with convolutions followed by a dense layer to produce transformation parameters (one can choose whether non-linear transformations are allowed or not). The given feature is then warped according to the estimated parameters before being passed to the next convolution layer. One of the techniques used by Li et al. (2020) follows the same principle, but is mainly applied to the decoder part of the architecture. Given two features of different resolutions, an optical flow field is dynamically computed to warp the low-resolution feature into an upsampled and more detailed version at the same resolution as the other feature.

Another way to address the challenges mentioned before is to use deformable convolutions (Dai et al., 2017). A deformable convolution layer consists of two regular convolution layers. The first one contains twice more kernels than there are weights in a kernel from the second layer. This is because it is used to generate 2D offsets for the entire convolution window at each pixel location of the input. Then, when processing the convolutions in the second layer, each pixel value is replaced by the value of the pixel at the corresponding offset with respect to the current location. In this way, the receptive field can be quickly expanded as needed, and different spatial transformations can be represented. The principle is further improved in methods which add dynamic weighting coefficients based on the same principle (Wang et al., 2022b). While some methods focus on 2D data, they also note that they can be easily applied to 3D segmentation, i.e. videos (Jaderberg et al., 2015; Su et al., 2023).

These concepts are applied to object segmentation (Schmidt et al., 2022). However, the authors claim that deformable convolutions, as previously presented, lack the inductive bias that regular convolutions have. Thus, instead of computing dynamic offsets for each pixel location, they compute dilation rates for the three axes (two spatial and one temporal), and they also use dynamic weights, as in Wang et al. (2022b), to further dynamically adapt the convolutions to the content.

In Li et al. (2018a) a dynamic method is employed for semantic video segmentation. The principle is not just to compute dynamic

weighting or deformation parameters, but to compute whole kernel coefficients. This means that for each pixel of a given feature, a kernel is created and applied only at that location.

This concept can be pushed even further thanks to the joint use of convolution and attention (see Section 4.4.1), leading to high-performance panoptic segmentation (Zhang et al., 2021). To achieve this, frames are first passed through a backbone and a decoder to produce high resolution and highly abstract features F . Then, N convolutional kernels are randomly initialized and applied to the features F to produce mask predictions for both semantic classes and instances.

From there, an iterative process refines the kernels and thus the predictions. The refinement process is as follows: First, the candidate masks are multiplied by the features F in order to generate instance/class-dependent features. Then, the former kernels as well as the modified features are fed into fully connected layers, followed by batch normalization to create new kernels. The kernels are further modified by passing through a multi-head attention head followed by a feed-forward layer, that allows the kernels to interact in order to model the image context. The updated class prediction can then be obtained through a fully connected layer taking the kernels as input. Meanwhile, the updated masks are obtained by a convolution with the kernels first passing through a fully connected layer.

This refinement process can be repeated several times, with each iteration yielding better kernels and predictions. Furthermore, the method can be adapted to video panoptic segmentation (Li et al., 2022b). The idea is to introduce an embedding constraint on the kernels thanks to a contrastive loss (see Section 4.8), a query system (see Section 4.4.2) that makes connections between current kernels and those from previous timesteps, and a fusion system that adapts kernels from previous timesteps and fits them to the current frame.

4.4.1. Attention mechanisms

Both regular and modified convolutions have been widely used for semantic video segmentation tasks. Recently, however, an alternative called “attention” has replaced convolutions in some of the state-of-the-art segmentation methods. Attention refers to a set of techniques that attempt to mimic the human attention mechanism by adapting to the content rather than working independently of the context.

Attention-based architectures first gained popularity in speech processing tasks (Vaswani et al., 2017). In that work, the encoder part relies on the self-attention mechanism, which tries to guess the strength of word dependencies within a text, as a human would do while reading. This is possible by first converting each word from the text into a vector which is the sum of the word’s semantic embedding vector and the word’s positional embedding vector describing its position in the sentence. The obtained vectors are then transformed into three different vector representations, namely the “query”, the “key” and the “value” vectors, thanks to three linear layers. Then, the dot product of the queries by the keys is computed, producing a self-attention matrix, which is then normalized and multiplied by the value vectors to produce attention-weighted outputs. This type of module is called an “attention head”. The three linear layers are learned to help select a context to focus on, so it is possible to compute multiple attention heads in parallel to capture attention in multiple contexts. The decoder part is similar to the encoder part, except that it guides what the model should currently focus on by introducing attention between the input embeddings and the currently produced output embeddings.

While transformers were originally designed to deal with sequential data, Dosovitskiy et al. (2020) showed that the same principle could be applied to images by using image patches as input embeddings. This form of attention is used in some works focusing on semantic image segmentation, which claim that features from different scales are complementary (Tao et al., 2020; Xie et al., 2021). Tao et al. (2020) therefore propose to use CNNs to generate segmentation maps, then introduce a module that considers two segmentation candidates of adjacent scales as input and use attention to produce a segmentation

candidate that takes the best of both. This system can then be used in a hierarchical fashion, taking the result of a module of a given scale and another segmentation candidate of adjacent but lower scale as inputs to another such module, and so on. Since then, several attention-based semantic segmentation methods have been developed (Thisanke et al., 2023).

Attention for image processing can be spatial (Tao et al., 2020; Xie et al., 2021), but it can also be applied to feature channels. Hu et al. (2018) introduce channel-wise attention where they consider the importance of the produced features of a given layer given the current data-relative context. To do this, a set of k features is transformed into a vector of size k by global max pooling. Then, a small fully connected network takes the vector as input and produces another vector of the same size in which each element is used to weight the associated feature map to increase or decrease its importance.

Peng et al. (2022) introduce another 2D semantic segmentation method similar to the one of Tao et al. (2020) in the sense that attention is used to decide what to propagate between the high-level but coarse features of the decoder and the low-level but detailed features of the encoder. The main difference from this other paper is that attention is not applied to segmentation maps, but directly to sets of features, thus requiring both spatial and feature-wise attention.

Both channel-wise and spatial attention are used in the context of video object segmentation. For example, Zhen et al. (2020) use a variant of channel-wise attention to select the most relevant features to describe the scene for the entire video. Specifically, a backbone computes features for each frame of the video, which are then clustered into k groups. The k clusters are then weighted in a manner similar to channel-wise attention. Once the k -weighted features are computed, their currently most relevant pixels in each frame are selected by computing the spatial attention between them and the frame features. For each frame, both the current frame features and the selected portions of the k -features are then passed to a CNN to perform the final segmentation.

The above mentioned methods model attention mostly in the spatial domain and do not consider temporal information. This is because attention has several drawbacks that limit its use in video applications. Notably, it lacks inductive bias and it has a large computational overhead as the amount of data increases. To address these issues, other video object segmentation methods (Duke et al., 2021; Yu et al., 2022) propose to use spatiotemporal attention, but with a limited scope, in contrast to global attention mechanisms. The work from Duke et al. (2021) introduce “grid attention” and “strided attention”, which enable to jointly compute, with a CNN, the features from the current frame and the features from previous frames selected by the spatiotemporal attention process. Yu et al. (2022) do not use a fixed window to compute local attention, considering both low-level features and optical flow estimation to choose in which spatial patches spatiotemporal attention should be computed to collect features from previous frames.

Finally, attention is also used in some semantic video segmentation methods (Hu et al., 2020; Wang et al., 2021b; Li et al., 2021; Subramaniam et al., 2022; Wu et al., 2022; Li et al., 2023; Su et al., 2023; Zhang et al., 2023; Athar et al., 2023). In the work of Hu et al. (2020), frames are preprocessed by different shallow CNNs that alternate in a circular fashion. Thus, in a sequence of consecutive frames, features of different nature are produced. These features are then propagated in a hierarchical way to the current space domain thanks to “attention propagation” modules that can associate features from one frame to the space of the next frame. In this way, the segmentation output benefits from information provided by previous contexts and from features of different types. In a different approach, attention is employed to build a kind of retrieval system (Wang et al., 2021b). The current frame is preprocessed by a backbone and the features are processed in parallel branches, each consisting of two convolutional layers. The first branch produces “query” features encoding the semantics of the current frame, while the other branch produces “current value” features with more

channels and storing visual details. On the other hand, each previous frame in a fixed temporal window with respect to the current frame has previously gone through a similar process, producing “key” and “value” features, respectively, which are stored in memory. Thus, the matrix multiplication of the query by the keys produces a matrix that indicates which “value” information should be retrieved from the memory of past frames. The result of the values multiplied by this matrix is then concatenated with the “value” features of the current frame, and given to a segmentation head. The segmentation head thus benefits from both the details of the current frame and relevant long-term information. In the next section, we will focus on methods that use graph architectures to improve their quality.

4.4.2. Query-based architectures

DETR (Carion et al., 2020) is an object detection method that relies on a novel “query-based” approach. DETR uses a CNN backbone to produce features that are further processed by a transformer encoder that takes positional encoding as a side input to induce the ability to make global links between feature elements. The improved features coming out of the encoder are then fed to a transformer decoder along with learned vectors called “queries”. These “queries”, which are independent of the input, help the decoder interact with the encoder features and manage potential object locations.

DETR has inspired methods beyond the field of object detection, in particular, some have proposed to apply this concept to instance segmentation (Fang et al., 2021; He et al., 2023), and others even aim to produce an architecture that could be used for any kind of image segmentation (Cheng et al., 2022). Instead of a transformer encoder, the latter work, includes a pixel decoder that creates a feature pyramid by gradually upsampling the low-resolution features produced by the backbone CNN, thus preventing the model from ignoring small objects. The transformer decoder also uses masked attention instead of cross attention, which restricts the attention to the surroundings of the predicted mask for each decoder query.

Query-based models can also be applied to Video Instance Segmentation (Cheng et al., 2021; Huang et al., 2022b), but also to Panoptic Video Segmentation (Wu et al., 2022; Li et al., 2023). The latter work notes that video segmentation methods produce either frame-wise or clip-wise results: the methods following the first option (such as Wu et al., 2022; Huang et al., 2022b) often lack of understanding of the temporal cues, while methods following the second option (such as Cheng et al., 2021) are unable to segment long videos consistently. To address this problem, this work splits videos into sub-clips that are first processed independently using the backbone and decoders of Cheng et al. (2021), and then the temporal consistency between adjacent clips is ensured by applying a contrastive loss to the output of self-attention layers that take the clip-wise decoder output embeddings as input.

The model of Athar et al. (2023) can be applied to various video segmentation tasks (including panoptic segmentation) simply by learning different queries. To do this, the input frames are fed to a 2D backbone, then the resulting features are transformed into spatio-temporal features thanks to a subnetwork made of alternating deformable attention layers (Zhu et al., 2021), which are spatially global but limited to the current frame, and temporal attention layers, which are spatially local but temporally global. The features produced at multiple scales by the spatio-temporal module are then fed to a transformer decoder along with the target queries in order to refine the queries through multiple layers of self and masked cross-attention. Finally, the refined queries are multiplied with the high-resolution output of the spatio-temporal module to produce the predictions.

4.5. Graphs based methods

A graph is a powerful tool that can be used to model the relationship between data instances. Unlike many methods that deal with temporal continuity only at a short range, relationships modeled through graphs

can be long-range in both spatial and temporal domains. Moreover, the graph construction can take into account priors and constraints, which make it easy to adapt to a given task. These features are especially useful in object segmentation methods. Wang et al. (2015a), use a graph to model the long-range relationships between superpixels (i.e., groups of pixels separated by strong edges in the spatial or temporal domain). The spatial connections of the graph are weighted according to the color distance, and the temporal connections are made only between superpixels that overlap, taking into account the displacement of the objects and the camera motion.

Wang et al. (2019a) rely on graphs to aid the exchange of information among frames. To do this, each frame is converted into an embedding by passing through an FCN. Then, the embeddings are used as nodes to construct a graph neural network. In this graph, each node is connected to itself and to every other node. Thus, the embedding nodes can update their information in multiple iterations, during which each node is modified by the sum of its own attention and that of its neighbors. Once all the iterations are done, each frame is segmented in a coherent way thanks to the shared information. However, such a method is not suitable for real-time applications.

Conditional Random Fields (CRF)

A special type of graph called “Conditional Random Field” or CRF is used in many segmentation techniques. This discriminative model is a generalization of Markov Random Fields that takes the form of a probabilistic graph. It models the dependencies between nodes so that the state of a given node is affected by the state of its neighbors.

Notably, CRFs are used in many segmentation methods because they can moderate the independence of the pixel-wise annotations, thus producing smoother and more consistent results. They can be used in video object segmentation networks (Song et al., 2018; Zhen et al., 2020) or in semantic video segmentation networks (Tripathi et al., 2015; Kundu et al., 2016; Saleh et al., 2017; Chandra et al., 2018). Sometimes the CRF module can be used as a 2D post-processing module that only spatially refines the segmentation maps (Song et al., 2018), but 3D CRF can also be integrated into a network to improve both spatial smoothness and temporal consistency in an end-to-end fashion (Zhen et al., 2020; Tripathi et al., 2015; Kundu et al., 2016; Chandra et al., 2018). In contrast, in Saleh et al. (2017), the CRF is integrated into the training pipeline so that the segmentation head learns to mimic its behavior and no longer needs it at inference time. To use CRFs efficiently, some peculiarities should be taken into account. In particular, due to motions such as rotations that produce complex pixel displacements, the resulting spatiotemporal space is non-Euclidean and should therefore be transformed into Euclidean space before passing through a spatiotemporal CRF (Kundu et al., 2016). A drawback of CRFs trained jointly with the network is that the gradient can be hard to compute if their loss function is non-convex. Chandra et al. (2018) solve this problem by using a special kind of CRF called “Gaussian Conditional Random Field”, which presents the peculiarity of having a convex loss function that is therefore easier to compute and optimize.

4.6. Long term memory based methods

Semantic segmentation and object segmentation in video are two closely related but distinct areas of research. Video object segmentation methods are often developed for a semi-supervised setting, which consists of reliably propagating an initial segmentation mask over time. For this task, the ability of a given network to rely on object appearance priors is very limited. Thus, the network focuses on adapting the segmentation mask to new frames. As described in Section 4.3, some methods rely on short-term cues to propagate the prediction. However, this type of strategy tends to accumulate errors over time and is generally unable to recover from large changes such as temporary occlusions. In contrast, some suggest to focus only on long-term temporal dependencies (Yang et al., 2019). In this sense, their method

consists in directly linking the original frame and its segmentation map to the current frame. To do this, first the embeddings of both frames are computed using a feature extractor. Then, the network is split into three branches. The first branch is a skip connection that feeds the current frame’s embedding to the segmentation head. The second branch computes the long-range spatial connections within the current frame embedding before feeding the result to the segmentation head. The third branch computes a correspondence map between the two frames using non-local operations that are closely related to the attention mechanism, and then feeds the result to the segmentation head as well. The segmentation head concatenates the features and performs a segmentation of the current frame guided by a long-range dependency.

However, one could argue that both short and long range dependencies matter. While early frames provide more reliable segmentation information, the current content is increasingly likely to be different from the original frame as the video length increases, thus requiring knowledge from intermediate frames to link the two contexts. It is therefore easy to understand the importance of memory in this domain. On the other hand, the storage and complexity requirements increase the more frames any memory mechanism employed considers. As such, finding the right trade-off is crucial.

Cheng and Schwing (2022) aim to achieve the best information selection for arbitrarily long videos. To do this, the authors designed a memory network, which stores data associated with keys that can be retrieved with specific requests. In this work, three types of memory are implemented: a short-term memory, a working memory, and a long-term memory. For each frame, the network gathers relevant information from the three types of memory to produce the corresponding segmentation output using a decoder. Each type of memory works differently and has a different purpose. The short-term memory, which is simply a GRU, is used to smooth the results at each frame. Retrieving information from both working memory and long-term memory works similarly: an encoder takes the current frame as input and produces a set of “Query” features. The query is then multiplied by the concatenation of “keys” features, which come from both working memory and long-term memory. The operation produces an affinity matrix that aims to select the most appropriate “value” features from both memories. At regular intervals, key and value features are generated from the current query and segmentation map and stored in the working memory. The oldest features from the working memory are regularly discarded, but to keep the most valuable ones longer, features whose ponderated affinity scores cumulated over time are the highest are transferred to long-term memory. When the long-term memory is full, the less relevant key/value pairs are discarded using the same selection process.

4.7. Uncertainty estimation

Traditionally, the segmentation task maps each input pixel to a unique hard label. However, when applied to real-world tasks, this may not be sufficient. For example, when the segmentation task is applied to medical diagnostic applications, clinicians need to know the confidence of the class predictions for each pixel. The same is true for systems that make critical decisions, such as autonomous vehicles. Also, in reality, a given pixel is not always best described by a single label, but rather by a mixture of labels, due to sampling quality, blur effect, or the intrinsic fuzziness of some objects.

To account for uncertainty, one approach is to apply a model several times for each frame with small variations such as dropout, and then average the results (Gal and Ghahramani, 2016). However, this type of strategy is slow because it requires each frame to be processed multiple times. To overcome this problem, Huang et al. (2018a) propose to take advantage of the properties of video content. The idea is that in semantic video segmentation, each frame must be processed anyway to produce its segmentation map. However, neighboring frames are often very similar except for some small displacements. These small

shifts can be enough to produce inconsistent results if the network is not confident in some of its estimations. By using optical flow to model the small displacements, one can warp the segmentation maps of neighboring frames to a common space, thus allowing the possibility to average the segmentation results and thus obtain an uncertainty estimation.

However, when a method aggregates estimations of the same frame with a network that has the same goal, it only takes into account the model uncertainty, but not the intrinsic fuzziness of some input data. Considering this problem, Wang et al. (2021a) develop a pipeline that can produce accurate matting maps of intrinsically fuzzy images. To do this, the ground truth continuous annotation is first used to generate several pseudo-masks using random thresholds. Then, these pseudo-masks are used to train a probabilistic UNet (Kohl et al., 2018), which is a combination of UNet and a conditional variational autoencoder (CVAE) that can learn to generate multiple segmentation hypotheses, each associated with a position in a low-dimensional latent space. For a given input image, the probabilistic UNet thus produces several prediction masks corresponding to different confidence levels. These prediction masks are then converted into an uncertainty map using the entropy of the pixel-wise annotations. Finally, the input image, the latent features from the probabilistic UNet, and the uncertainty map are fed to a matting network, which produces a smooth output based on the cues it has been given.

Finally, to perform video matting with high quality and consistency while remaining fast, Wu et al. (2018b) exploit the temporal information inherent in the video format thanks to a recurrent architecture. This work demonstrates the effectiveness of training such a network to perform both the matting and segmentation tasks, especially when dealing with synthetic training data. It also uses a trainable guided filter, which can efficiently upsample a coarse segmentation output to match the details of high-resolution input frames.

4.8. Loss functions

Most of the improvements described so far involve changes to the model architecture. However, it is not the only thing that can be improved. The loss function describes how well the model achieves its objective. However, defining a metric for a task as complex as semantic video segmentation is not trivial. Thus, the way a given model will perform depends heavily on its loss function. Jadon (2020) gives an overview of loss functions for image semantic segmentation and identify what is best to deal with unbalanced datasets, hard to segment boundaries, focus on hard examples, or focus on shapes and structures.

Even if these loss functions are not directly designed for video content, some of them can be easily adapted to semantic video segmentation because of their independence from model architectures. For example, an “inverse transformation network” that learns to guess the homographic transformation between two edge maps can be used to train a model to compensate for the limitations of pixel-wise losses (Borse et al., 2021). To train a network using this method, the student network produces high-level features that are fed to a segmentation head, which produces a segmentation map. This segmentation map is compared to the ground truth using a pixel-wise cross-entropy loss. In parallel, the high-level features are also fed to a boundary detection head, which outputs an edge map. The result is compared to the edges of the ground truth with a balanced cross-entropy loss to train the edge detection head. Then, both edge maps are divided into tiles whose size and number depend on the trade-off between the importance of local and global context. Finally, the “inverse transformation network” is used to estimate deformation parameters between corresponding tiles. This last step allows to measure the total deformations between the segmentation candidate boundaries and the ground truth boundaries. This is interesting because most of the loss functions are pixel-wise and therefore cannot measure the spatial variations such as translation, rotation, and scaling. Similarly, “contrastive losses” (Wu et al., 2022;

Li et al., 2023) can be used to ensure temporal consistency between consecutive frames or sub-clips that are computed independently. It can also improve robustness to phenomena such as occlusion. The key idea is to put successive outputs into a common embedding space and add a constraint that encourages vectors representing the same object or class to be closer together while encouraging vectors representing different objects or classes to be further apart.

To train deep networks, it can be inefficient to apply a loss function only to the final output layer. Therefore, one can add a second segmentation head in a shallower part of the network and apply a loss function to it in order to guide the previous layers to learn meaningful features in a less indirect way than deep back-propagation (Zhao et al., 2017). The additional segmentation head and its associated loss function are only useful during the training phase and can be dropped afterward, thus having no impact on the inference time during testing.

The loss function can even help to adapt image semantic segmentation methods to video content by using knowledge distillation (see Section 5.5) and a temporal loss function to train a frame-by-frame student network to be temporally consistent (Liu et al., 2020).

5. Accelerating the inference

Video content provides access to the temporal dimension. The variation between frames can be analyzed and used to create spatio-temporal features, aggregate different contexts, and provide semantic clues about object semantics through motion analysis. All of these are useful for improving segmentation quality. However, the architectural improvements that take advantage of the temporal dimension often come at a significant increase in computational cost if nothing is done to compensate. Moreover, such methods do not take into account the fact that, just as the spatial neighborhood of a pixel in a natural image is often similar, it is also the same for images that are close in time. Therefore, in this section we will focus on methods that accelerate the inference to compensate for the data increase of video content or to account for the redundancy of data.

5.1. Multiscale methods

The goal of semantic segmentation is to assign a label to each pixel of the input. As a model trained for classification, a semantic segmentation model must generate high-level semantic features to distinguish the different classes. To do this, a convolutional neural network stacks layers that produce features of increasing complexity and receptive field. The assumption is that the more layers, the better the semantics. However, deep networks suffer from several drawbacks. While they can produce meaningful features, these semantics are not well localized, and thus are not suitable for precise determination of object shapes. Further, deep networks contain many weights, but increasing the number of parameters makes the model harder to train, slower to perform inference, and prone to overfitting.

The receptive field of a model can be increased using pooling, stride and dilation strategies, requiring fewer layers to produce meaningful semantics. However, while this can be effective for classifiers, it is not suitable for segmentation task because all of these techniques affect the resolution of the features, making them unsuitable for precise pixel labeling. To address this challenge, Szegedy et al. (2015) introduce “inception layers”, which is a module that breaks the idea of using convolutions in a sequential way by proposing to use them in parallel. More specifically, the idea is to use a single-layer convolution kernel of several sizes in parallel, and then concatenate their outputs, thus obtaining features of both large and small scale. Originally, one of these modules contained convolutions of size 1×1 , 3×3 , and 5×5 , as well as a parallel pooling of size 3×3 . Recently, the design has been made even more temporally efficient by adding 1×1 convolutions, as discussed in Section 5.4.2. Individually, an inception layer is more computationally intensive than a regular convolutional layer, but by

increasing the richness of features produced in each layer, fewer layers are necessary to build meaningful yet precise features, resulting in shallower and thus faster networks. Nevertheless, even if this approach offers an improvement for semantic segmentation models, it may not be sufficiently fast to be applied to video segmentation models because it still uses 5×5 convolution kernels that are slow to compute.

An alternative to inception layers is introduced by Zhao et al. (2017). The module, which is called “pyramid pooling module”, relies on the use of pooling kernels of varying size instead of convolution ones. Pooling is indeed faster to compute than convolutions and makes it possible to gather information from a much larger scale through global pooling for instance. In such a module, each pooling operation is followed by a 1×1 convolution that reduces the number of channels. At the end of the pyramid pooling module, each output is upsampled to match the spatial dimensions of the input feature maps, and then concatenated with them to form a tensor containing information at multiple scales. The pyramid pooling module is used in semantic segmentation methods adapted to images (Li et al., 2020; Xu et al., 2022; Peng et al., 2022) and videos (Hou et al., 2019b; Zhao et al., 2021). Although the pyramid pooling module allows the model to be shallower, it is itself computationally intensive. Therefore, to further reduce the computations, the module is used in the bottleneck part of the network, at the interface between the encoder and decoder parts of the network, where the dimensions of the input features are smaller and thus require fewer operations to be processed.

Another way to quickly increase the receptive field of a multiscale layer without increasing the computation time is to use convolutions with multiple stride or dilation values. Parallel convolution kernels with different dilation rates have been employed for semantic image segmentation (Gao, 2021) and video object segmentation (Song et al., 2018). Alternatively, different stride values can be used to obtain local and precise features as well as more global but coarse features (Hou et al., 2019a).

We have seen that using multiscale paths can help build strong features without the need for deep networks. Such methods therefore reduce the number of parameters to learn and the computations needed to perform inference. However, such parallel architectures can paradoxically reduce the GPU’s ability to parallelize operations in the model (Ma et al., 2018). To ensure that the multiscale strategies employed are not counterproductive in terms of inference time, they must be carefully designed, for example by avoiding parallel paths with too heterogeneous complexity.

5.2. Reusing early layers features

In a regular convolutional network, early layers are able to extract local patterns that form detailed but simple feature maps, while deeper convolutional layers learn more complex features. However, by definition, high-level features retain semantic information rather than visual details that can help create an accurate segmentation map. And since each convolutional layer is only connected to adjacent layers, the only way to access early low-level features that could be helpful is to force their regeneration in each layer. Because the number of generated features in each layer is limited, this phenomenon creates a competition between the generation of new high-level features and the preservation of low-level features, which is detrimental to network performance (Huang et al., 2017). This phenomenon is illustrated Fig. 5a.

As mentioned in the previous sections, another problem with deep networks is that they are difficult to train. The reason for this is called the “vanishing gradient problem” and is related to the way back-propagation works (i.e., the mechanism that calculates the error contribution of each neuron in the network to update its weight). In the last layer of a model, errors can be directly assigned to the neurons that produced that output, so the strength of their responsibility is high and their weight is updated strongly. However, those neurons were

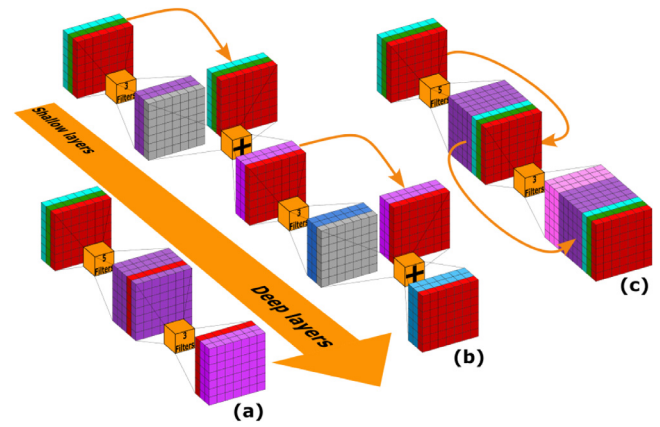


Fig. 5. If a shallow feature is useful for deeper processing, a classical CNN like (a) has to propagate it with a filter through each convolution layer along the way. With residual networks like (b), the task is simpler because the network only has to learn to create a zero residue at the right location for each convolution layer. However, with densely connected architectures like (c), no filters are needed to propagate early features. However, the number of input features increases with the depth of the layer.

themselves influenced by the output of a larger number of neurons from the previous layer, so some of those previous neurons are indirectly responsible for the errors and are updated. But, because there are many of them, their responsibility is diluted and the updates of their weights will be weaker than those of the last layer. We can repeat this reasoning for each previous layer in the trace chain until we reach the input layer. There, the responsibility for the errors is so diluted that it will be hard for the neurons to learn anything useful, the gradient has “vanished”. This problem is easily illustrated by the fact that it is hard to teach a very deep network to learn a simple function (i.e., the identity function) (He et al., 2016).

Instead of building slow and hard to train networks with many features for each layer to regenerate the early features, it is possible to reuse already produced features without further processing them. Following this idea, He et al. (2016) introduce “skip connexions”, which allow early features to be passed directly to the next layer, bypassing the convolutions. Architectures using these skip connections are called “residual networks” because, unlike regular networks that try to approximate a target function, each layer learns the “residue” to add to the input to produce the layer’s output. Because there are ways to skip each layer, the shallow neurons have connections that are relatively close to the output, making back-propagation more efficient and thus solving the vanishing gradient problem. Returning to the example from the previous paragraph of a network learning the identity function, we can see that it would now be much simpler, since all the model has to do is learn to pass the input through the skip connections all the way to the output, as illustrated Fig. 5b.

In residual networks, the output of each skip connection is added to the output of the skipped layer. Thus, deep layers still cannot access the unaltered early features (unless the layers produce a zero residue). To improve the ability of layers to access early features, Huang et al. (2017) suggest creating connections so that each layer has access to every feature that precedes it. This type of network performs better than ResNets. However, as illustrated Fig. 5c, the number of features provided to the layers grows rapidly with depth. Thus, such networks are limited in depth. To address this, the authors improve these architectures in subsequent work by pruning the connections during training (Huang et al., 2018b). The idea is to start with all the connections and gradually remove those that are associated with small weights compared to other weights. In this way, only the most important features are transmitted through the network, which is more memory and time efficient.

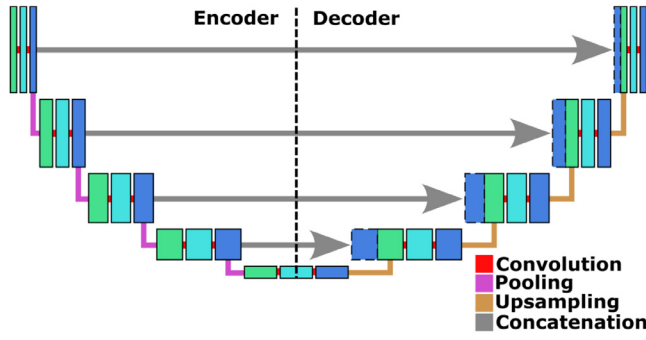


Fig. 6. A diagram showing the principle of the U-Net architecture (Ronneberger et al., 2015). The encoder progressively reduces the resolution of the features as it improves their semantics. Then, the decoder progressively upsamples these semantics and reconstructs the details, guided by the concatenated high-resolution features from the encoder.

As shown by Huang et al. (2018b), not all features are equally important at every step of the network. For example, some early features that contain details are not particularly important when creating high-level, coarse semantics. Also, dense networks like the ones discussed in the previous paragraph are not compatible with the progressive reduction of feature sizes in deeper layers. In contrast, feature downsampling helps to generalize, to gain robustness to small variations, and also to improve inference time. Thus, dense architectures are not necessarily the best way to produce efficient and fast networks. In fact, reusing early features to improve segmentation quality remains a good idea, but can be done after the semantic production part. Ronneberger et al. (2015) divides the model into two main components, namely the encoder and the decoder, as illustrated Fig. 6. The encoder consists of blocks of two successive convolutions followed by a max-pooling layer. Each block produces features of reduced resolution compared to its input. The output of the last block of the encoder is then provided to the decoder, which is made up of blocks of two successive convolutions followed by an up-convolution layer. The role of these blocks is opposite to that of the encoder, since it is to increase the resolution of the features. To achieve this, the decoder blocks also receive the features from the encoder whose resolutions match the target resolution. To summarize, the semantics are built classically, without accessing any features other than those of the previous layer, and then the semantics are upsampled with the guidance of early features. This architecture is therefore much more time efficient while maintaining very high precision. In a similar vein, Pinheiro et al. (2016) replace the simple link between early features and their corresponding refinement module with a convolution layer to adapt the features to the new context. Other 2D segmentation techniques use this principle (Lin et al., 2017) or some attention-based variants (Peng et al., 2022) including in the Video domain (Su et al., 2023). The detail level of segmentation methods can be greatly improved by using guided filters, which upsample a coarse output using the high-resolution frame directly as a guide (Wu et al., 2018b). Finally, the concept of guided upsampling can be applied to video segmentation by linking layers of different scales with 3D separable convolutions (Hou et al., 2019a). In many architectures, the transition between encoder and decoder also has a multiscale mechanism, as discussed in Section 5.1.

To produce qualitative semantic segmentation results, most encoder-decoder networks must first produce high-level semantics with their encoder. Typically, the deeper the layer, the higher the level of semantics produced, and the lower the spatial resolution. In a sense, the bottom of the decoder produces a low-resolution proto-segmentation map. Thus, the quality of a network output also depends on its ability to progressively upsample this proto-segmentation map with the decoder.

As discussed above, some methods do this by progressively fusing the interpolated proto-segmentation map with higher-layer features

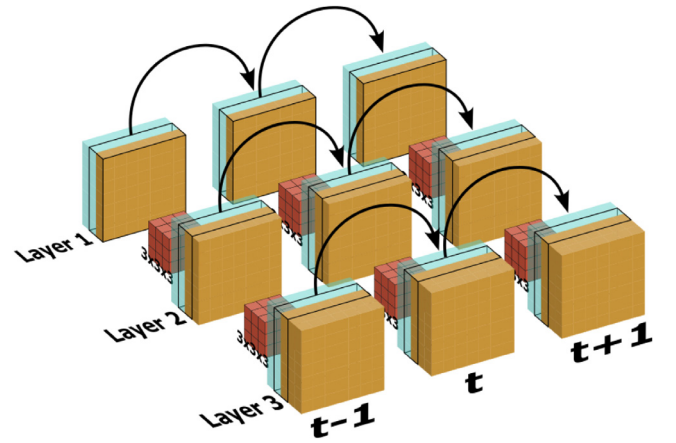


Fig. 7. The principle of “temporal skip connections” (Köpüklü et al., 2022). In this illustration, the parts of the tensors that are not involved in a recalculation process are not represented. New content is represented in orange, and temporal skip connections are represented by black arrows that transfer the bold framed features from the previous time step to the current time step at the position represented in transparent blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

that contain more spatial detail. However, according to Elhassan et al. (2022), if the network is effective at preserving the semantics from the initial proto-segmentation map, the interactions with the feature maps from shallower layers have an increasing semantic gap, and thus, making associations between the semantics and the spatial details becomes more difficult, leading to a degraded result. To address this issue, this work proposes a network with three parts: one is the backbone encoder, the second one is a multi-scale feature fusion module, and the third is a semantic upsampler module. In such an architecture, the multi-scale feature fusion module can learn the associations between features from multiple layers of the encoder without worrying about the loss of high-level semantics. Meanwhile, the semantic upsampler, guided by the multi-scale feature fusion module, can effectively upsample the proto-segmentation map without suffering from semantic gaps with the features it relies on.

Köpüklü et al. (2022) make the observation that 3D convolutions are a powerful tool for video processing because such operators can process the temporal dimension along with the spatial dimensions. However, to process two consecutive frames, such a network performs a lot of redundant computations in the temporal dimension, because both frames share common previous frames. Inspired by the skip connections described in this section, the authors propose to create “temporal skip connections” that propagate a part of the tensor of a given layer at the previous timestamp to the same layer at the current timestamp. More precisely, the part of the tensor that is propagated corresponds to the previous frames that are not processed together with the current frame in the given layer and therefore do not need to be recomputed. This principle, illustrated Fig. 7, provides a good transition to the next section, where we will describe methods to further reduce temporal redundancy.

5.3. Temporal redundancy reduction

Natural video content has a lot of temporal redundancy, meaning that successive frames often share most of their content. So far, the mechanisms we have described do not take this into account and thus perform some redundant computations. Hopefully, there are ways to exploit this redundancy to reduce inference time.

5.3.1. Segmentation propagation to reduce redundancy

The simplest way to exploit temporal redundancy is to reuse the segmentation map of the previous frame for the current one. To propagate a segmentation map between consecutive frames and account for changes between them, one can use optical flow estimation. As explained in Section 4.3, optical flow estimation, which can be computed with neural networks (Dosovitskiy et al., 2015; Ilg et al., 2017), allows, for example, to warp one frame to the next. Consequently, it is possible to apply the same transformation to warp the corresponding segmentation map to the next frame, thus correcting the small deformations due to motion. However, optical flow estimation is not perfect, and neither is the resulting warping. If the warping operation is applied many times in a row, errors can accumulate and produce a poor quality result. Similarly, estimating the motion/distortion between two distant frames also leads to poor results because the content between the two frames is too different to accurately match the corresponding pixels.

To address this, Jain et al. (2018) propose a compromise between segmentation map reusability and error accumulation by selecting reference keyframes at regular intervals and propagating the segmentation for the other frames. Specifically, keyframes are segmented by a deep convolutional network, which is slow but powerful. Then, until another keyframe is reached, the generated segmentation map is warped to the current frame thanks to optical flow estimation. In parallel, a much shallower network computes low-level features of the current frame and concatenates them with the warped segmentation map to guide a 1×1 convolutional layer that outputs the final current segmentation. Because this model treats a video sequentially, it cannot use future keyframes to guide the estimation of the current frame. Also, the fact that the keyframes are determined by a regular time interval makes them not adaptable to sudden changes in the scene. To account for this, an LSTM cell can be employed to select keyframes (Mahasseni et al., 2017), which are then segmented using a deep 2D segmentation method. Finally, the segmentation of the remaining frames is done by interpolating the neighboring keyframes with a single 5×5 CNN layer.

5.3.2. Reusing high level features

Segmentation maps are the end product of segmentation techniques. Ideally, the segmentation head that produces them takes the semantics from high-level features and reframes them in a precise way using low-level details, discarding the general neighborhood knowledge that high-level features typically have. The result is highly meaningful, but also very specific and localized, so it does not lend itself well to further modifications or corrections. Thus, a middle ground is to propagate high-level features instead of just the final segmentation map. Propagating high-level features instead of segmentation maps also has the advantage that such methods are easily adaptable to tasks other than semantic video segmentation.

One such approach is proposed by Zhu et al. (2017), where keyframes are selected in a regular schedule and processed by a deep CNN to obtain high-level features. Each non-keyframe is fed together with the last associated keyframe into a shallow network, which determines a deformation flow map and an uncertainty field. The flow map is then used to warp the high-level features to the current frame. The warped features are then multiplied term by term by the uncertainty field to account for potential warping errors due to phenomena such as occlusion. Each high-level feature is then processed by the same segmentation head, regardless of whether the features are directly generated or warped from a keyframe. Similarly, a lightweight detail enhancement network can be applied to each frame after feature propagation to account for inter-frame distortions and occlusions (Li et al., 2019).

Li et al. (2018a) note that simply warping features in which each pixel contains cues about spatial neighborhood is not optimal. Also, as mentioned above, regularly spaced keyframes do not account for significant scene changes. To address this, Li et al. (2018a) suggest that each frame passes through a low-level feature extractor. Then the

result is fed to a small network along with the low-level features from the last keyframe. The small network then decides whether the current frame should become a keyframe or not. If so, the low-level features are fed into a deeper network that creates high-level features. If not, the low-level features of both the keyframe and the current frame are fed into a convolution layer that outputs convolution kernels for each pixel. These convolution kernels are used to perform a dynamic convolution (as explained in Section 4.4) on the high-level features from the last keyframe, warping them to the current frame space with more freedom and reliability than optical flow-based methods. Finally, for each frame, the high-level features are processed by the same segmentation head, regardless of whether they are directly generated or not.

Some work claims that most methods that warp past features do not take into account the intra-frame correlation between pixels within the current frame, and thus reduce the accuracy (An et al., 2023). To address this problem, a dual local and global correlation network can be constructed. First, such a network has a heavy branch that is used to process the first frame as well as keyframes. The heavy branch starts with a low-level feature encoder that takes any frame as input, then it is followed by a heavy encoder that produces high-level features, and finally, the branch ends with a segmentation head. Any frame that is not the first in the sequence passes through the low-level encoder. Then, corresponding relations on the low-level features of both the current frame and the last key frame are processed thanks to a local attention module (see Section 4.4.1).

To determine whether the current frame will become the new keyframe, the inputs and outputs of the local attention module are fed into a decision network that is trained to predict the error between the results of the heavy and light branches respectively. If the predicted error is above a predefined threshold, the frame becomes the keyframe, otherwise the frame computation continues with the light branch. The next step in the light branch is simply to multiply the output of the local attention module by the high-level features from the last keyframe so that the derived high-level features are pondered by the local correspondence between the two compared frames low-level features. Meanwhile, the low-level features from the current frame are further processed by three convolutional layers and the output is added to the propagated high-level features described above. The resulting tensor, which is subject to a temporal consistency constraint, is finally transformed into a segmentation map by passing through the same segmentation head as in the heavy branch.

5.3.3. Clock networks

In the previous sections, we saw that segmentation maps, and more generally high-level features, change slowly (Wiskott and Sejnowski, 2002). More specifically, low-level signals (or features) of a given video content change faster than high-level ones. In a classical CNN, however, each level of features is computed at each temporal step. Methods that reuse segmentation maps or high-level features for multiple frames provide an incomplete solution to this problem because redundancy reduction is only considered at one scale. To minimize redundancy, a refresh rate adapted to the speed of feature changes can be applied at multiple scales within the network. This is the idea behind clock convnets where low-level features are refreshed every time, while deeper features are refreshed less frequently depending on their depth (Shelhamer et al., 2016). The work rely on networks that use skip connections to form a residual network. Thus, changes in early layers are propagated to the end of the network even if the outputs of some layers are frozen. In this method, a layer update can be triggered by scheduling, but also by detecting important changes in the features of the previous layer. A similar concept is exploited by Carreira et al. (2018), which also focus on improving the parallelization of each CNN layer. This method also uses other tricks such as knowledge distillation (see Section 5.5) and feedback, which is the fact that the segmentation result of the previous frame is given as input along with the current frame. One of the drawbacks of methods that use uneven layer

activations over time is that their computational load is inconsistent over time and can be equivalent to frame-by-frame methods during computational peaks. Thus, while fast on average, these peak delays make such methods incompatible with real-time applications.

5.3.4. Reusing unchanged regions

Instead of dynamically adapting the computation of the network to its depth, some semantic video segmentation methods propose to spatially adapt the feature computation depending on the amount of changes in that area (Xu et al., 2018). In concrete terms, there are no keyframes, but rather regional keys. In fact, each input frame is first divided into several fixed regions. Then, for each region, the current input and the last corresponding key region are fed into an optical flow network followed by a decision network that determines the amount of motion between the two. If the amount is small, the segmentation map from the key region is warped to the current space, thus avoiding the computation of the current one. Otherwise, the warping would be imprecise, so the segmentation map of the current region is computed with a deep CNN and the result serves as the regional key for the next frames.

Rhee et al. (2022) use a different approach where they create a network that can learn to compute only features that help correct the segmentation evolution relative to the current and previous frames. To do this, the current frame first goes through a low-level feature extractor. Then, both outputs of the current and the previous frame are fed to a network that determines a pruning mask for the blocks of a deeper ResNet. At the same time, the low-level features of both frames are partitioned into patches, and the similarity between corresponding patches is estimated using the cosine similarity metric. As a result, in addition to the previously computed mask, we obtain a 2D mask that contains the similarity between adjacent frames for each patch. The current frame is then fed into a deep residual network, some parts of which have been frozen by the first mask. Since the high-level features from the previous frame have already been computed, they can be mixed with the current frame by using the 2D mask to weight the contributions from both sources. Finally, the blended features pass through a segmentation head to create the current segmentation map. In the next section, we will see that it is possible to go even further in terms of feature calculation savings.

5.3.5. Removing high-level features

As we discussed earlier, high-level features are built by aggregating low-level ones. Computing and aggregating various low-level features is a slow process. Thus, if high-level features could be approximated by low-level features from different time steps, it would be possible to distribute their computation over time, thus reducing inference time. To achieve this, frames can be alternately processed by different shallow networks in a circular fashion (Hu et al., 2020). These shallow networks, trained with knowledge distillation, produce complementary low-level features. Because they are from different time steps, the features produced by a full cycle are not aligned with the current frame. Thus, instead of using optical flow to align previous features to the current space, the authors propose to use a sequence of “attention propagation modules” that dynamically select what to take from each of two successive feature sources and propagate it to the latest one. At the end of the chain, the output corresponds to the segmentation of the current frame.

5.4. Modified convolutions and substitutes

Convolution is a computationally intensive operation. While large kernels increase the receptive field, they require more computations than small kernels. A large number of kernels is also preferred when the network needs to be able to produce various features. However, this is another factor of computational overhead. Finally, 3D convolutions, which are often used in the context of semantic video segmentation, are also a cause of increased inference time. To compensate for this computational overhead, several modifications have been applied to convolutions. We summarize the most common ones in the following.

5.4.1. Stride and dilation

One of the most common ways to reduce computation time is to increase the stride hyper-parameter, as depicted in Fig. 8a. This increases the offset step between convolution operations, thus preventing successive operations from overlapping too much on the same pixels. Another effect is that the output tensor size is reduced compared to convolutions with smaller strides, thus reducing the number of operations on the next layers. However, stride does not increase the effective size of the convolution kernel. Hence, the nature of the features produced by 3×3 convolutions with or without a large stride is identical, the ones with a large stride are just sparser, thus missing some intermediate features.

To create features of a larger scale without increasing the number of parameters or without having inconsistent outputs, one can use dilation coefficients instead. The idea is to make the kernel larger without changing the number of neurons by adding space between each of them. In this way, the features produced represent a larger but less detailed portion of the input features, as shown in Fig. 8b. Dilated convolutions are notably used in a semantic video segmentation technique based on 3D convolutions (Qiu et al., 2018). Since 3D convolutions are by nature very computationally expensive, the use of dilation in this case helps to have large kernels without unbearable complexity.

However, both stride and dilation coefficients are associated with the loss of some input detail. A high stride coefficient results in the creation of highly detailed features that are sampled more sparsely in the input tensor, while a high dilation coefficient results in larger scale features produced without increasing the kernel size (and thus the resolution). To take advantage of the computational reduction offered by such convolutions without suffering from their shortcomings, it is possible to combine a multiscale approach (see Section 5.1) with the use of dilated convolutions (Song et al., 2018; Gao, 2021). In this way, paths using regular convolution kernels of small size produce detailed local features without much computation, while parallel paths using convolutions with dilated coefficients of different scales can produce coarser but more general features at about the same time.

Intuitively, one can understand that it would be great if a convolution layer dealing with natural images could dynamically adapt its behavior depending on the nature of the area currently being computed. Some parts of a given input may indeed contain important details, while other parts may contain structural information on a larger scale. Schmidt et al. (2022) propose a strategy that makes this possible. This method falls into the category of dynamic methods described in Section 4.4 and is based on the frame-wise segmentation technique of Dai et al. (2017). Specifically, they introduce “dynamic convolution modules”. Such a module first contains a regular 3D convolution layer with small-scale kernels, whose goal is to determine for each position whether it is more useful to extract details or larger-scale structural information. To achieve this, the layer is trained to produce three dilation values and one emphasis value at each position. The three dilation values are then used by a second 3D convolution layer within the module. For each location of the module’s input, the second layer takes the previously computed dilation values and applies them to the three dimensions of its kernels before performing the convolutions. Each local result is then multiplied by the corresponding emphasis value to produce the module’s output. By using such modules, the model is able to dynamically adapt to its content while maintaining the inductive bias given by the grid of convolutional kernels. Furthermore, such a network learns the trade-off between detail and structural information while maintaining an identical computational time for each case.

5.4.2. Kernel factorization

A 2D convolution kernel has two spatial dimensions. However, regular convolution kernels also have a third dimension whose size is equal to the number of channels in the input tensor. This is because regular convolutions always compute the full depth of the input tensor for each spatial location. A 2D convolution kernel is represented Fig. 10a and a 2D convolution layer of three kernels is illustrated Fig. 9a. In most

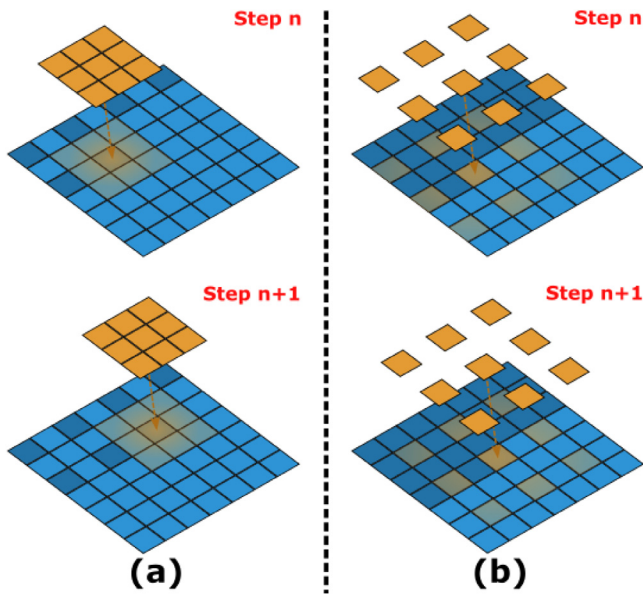


Fig. 8. The behavior of two different convolution kernels (in orange) on an input feature map (in blue): (a) A 3×3 convolution with same padding and a stride of 2. (b) A 3×3 convolution with same padding and a dilation factor of 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CNN-based models treating images, the first input has three channels corresponding to the three color components. Then, each layer contains multiple convolution kernels allowing the model to capture different features and contexts. Thus, as we go deeper into a CNN, the spatial size of the input features typically gets smaller, but the number of channels increases. Since regular convolutions consider each channel from the previous tensor, most of the computation time of the deep layers is due to the large number of channels.

To address this problem, Krizhevsky et al. (2012) propose “group convolutions”, illustrated Fig. 9b. The idea is to perform parallel convolutions on subsets of the input channels to reduce computation. Xie et al. (2017) show that this not only helps to reduce inference time but also helps to produce better representations, thus increasing quality. Sifre and Mallat (2014) go further and introduce “depthwise convolutions”, which are an extreme case of group convolution where each channel is computed by an independent kernel. In both cases, the channel interactions are greatly reduced. Thus, to ensure that the network retains links between features without increasing the computation time, one can use $1 \times 1 \times d$ pointwise convolutions that apply on the whole channel depth. Group convolutions and depthwise separable convolutions can then be used to create lightweight yet well performing architectures (Ioannou et al., 2017; Howard et al., 2017).

More specifically, group convolutions and depthwise convolutions are used in various video-related applications such as classification (Tran et al., 2019), object segmentation (Mahadevan et al., 2020), and semantic segmentation (Jin et al., 2017). However, Ma et al. (2018) suggest that while using group convolutions and 1×1 convolutions reduces the number of operations, it also increases the storage cost, which can reduce the speed of the network. Therefore, the number of such modules must be chosen properly to avoid the opposite effect of what is intended.

The idea of reducing the number of neurons of convolution kernels without changing their size can be pushed further by stating that just as the channel dimension can be used to factorize regular convolutions, the same is true for the spatial dimensions (Jaderberg et al., 2014). A regular 2D convolution kernel can be factorized in several ways other than group, depth-wise, and point-wise convolutions. Spatial dimensions are first processed using a 1D convolution, and then, the other

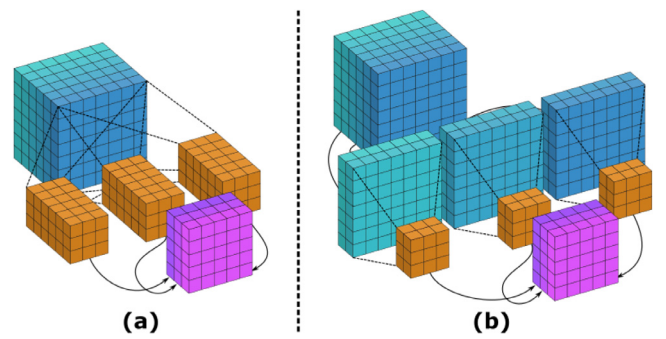


Fig. 9. The difference between regular convolutional layers and group convolutional layers. (a) A regular 2D convolutional layer of three 3×3 kernels, each with a depth of 6 channels, for a total of 162 neurons. (b) The same input tensor is divided into 3 groups, so that each convolutional kernel computes only two layers. This reduces the number of neurons to 54.

spatial dimension along with the channel dimension are processed using a 2D “flat” convolution. Another solution, which is suggested in the same work, is similar to depth-wise convolutions but goes even further by replacing each 2D depth-wise kernel by two successive channel-wise 1D convolutions. This is illustrated Fig. 10b. Interestingly, the larger the original kernel, the more the kernel factorization reduces the computational complexity.

Since 3D convolutions are even more computationally intensive than 2D ones, Tran et al. (2018) suggest that instead of using regular 3D convolutions to process videos, each could be approximated by a factorization of a regular 2D convolution followed by a temporal 1D convolution. Theoretically, such a module offers less descriptive power than regular 3D convolutions. However, due to the computational overhead of regular 3D convolutions, it is rarely possible to replace every 2D convolutional layer with its 3D alternative, so video processing methods have to choose which layers to replace 2D convolutions in and which not to. Such a choice is not trivial and often depends on the task at hand and the network architecture. Nevertheless, as factorized 3D convolutions are much lighter, they can be used in many layers of a given network (Mahadevan et al., 2020), thus eliminating the need to make a difficult choice.

Qu et al. (2020) propose to further reduce the computational overhead of 3D volume computation by factorizing each 3D convolution kernel into three perpendicular 1D convolutions interleaved with dense connections. On the other hand, Gonda et al. (2018) offer a trade-off between factorized convolutions, which are fast but approximate, and regular 3D convolutions, which are more precise but slow. In this latter work, three modules like those from Tran et al. (2018) are parallelized in such a way that the 2D convolutions of the three paths are orthogonal to each other. In this way, rich features can be generated in each dimension, thus replacing several successive regular 3D convolutions. However, to the best of our knowledge, these two methods have not yet been used for semantic video segmentation.

5.4.3. Modified convolution modules

The modified convolutions described so far can be used as building blocks to construct modules that make CNN architectures even more efficient. Sandler et al. (2018) explain that the information contained in a regular set of feature maps can be efficiently embedded in a lower-dimensional subspace. On the other hand, 1D convolutions can be used to change the number of channels of a given tensor, including compressing some feature information into a lower-dimensional subspace. However, the authors also tell us that compressed subspaces are not suitable for transformations followed by ReLU activation because they either result in information loss or poorly descriptive linear transformations. Therefore, the classical ResNet architectures (illustrated in Fig. 11a) that use skip connections between uncompressed features

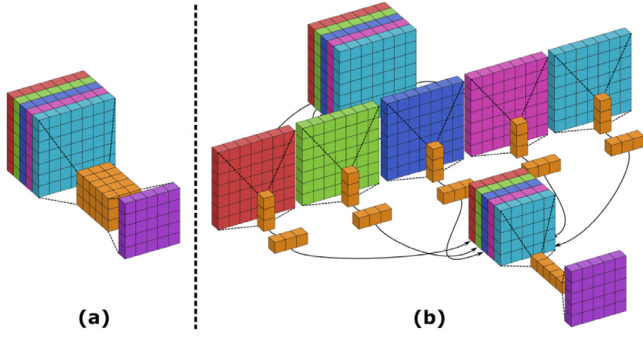


Fig. 10. (a) Features maps being processed by a 3×3 2D convolution. (b) Features maps being processed by a separable 3×3 2D convolution that uses less neurons (represented in orange). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

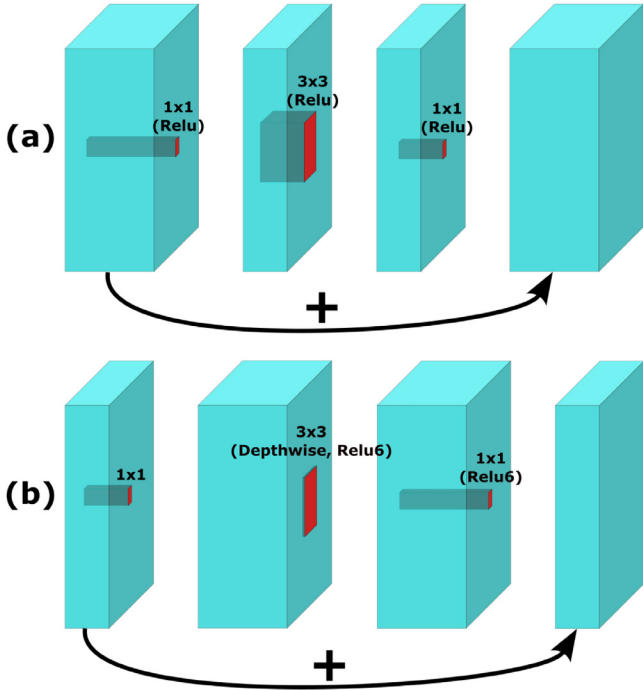


Fig. 11. (a) A regular residual block. The skip connection is made between channels with many layers, and the Relu activation is applied at each step, even if the information is in a compressed form. (b) An inverted residual block, which uses less computational resources and uses Relu activation only where it does not cause information loss (Sandler et al., 2018).

and that use slow 2D convolutions and destructive activation functions between the intermediate compressed features are poorly efficient. On the contrary, this paper explains how to construct “inverted residual blocks” (illustrated in Fig. 11b) that consist of having skip connections between dimensionality reduced tensors and using 1×1 convolutions to construct wide intermediate tensors whose redundant information can then be pruned by depthwise convolutions followed by ReLU6 activation (see Section 5.8).

Zhang et al. (2017) explain that in a model that uses group convolutions followed by 1×1 convolutions, the pointwise operations represent most of the computational overhead. In fact, each group produces multiple channels as output, which are then mixed with all the other channels from the other groups thanks to 1×1 convolutions that take the entire set of channels as input. The authors then state that while channel interaction between different groups is essential to building strong representations, the way it is done can be more

efficient than that. The idea of the improvement is to avoid the use of 1×1 convolutions that take all channels as input. Instead, they use 1×1 convolutions that take only one channel from each group as input. Thus, several small 1×1 convolutions can be used in parallel to create new groups based on mixed features from the previous groups. To ensure that all previous features are used to create the new groups, each 1×1 convolution takes a different channel from a given group. This preserves the group interaction high with minimal computational overhead. Even though it is not random at all, the authors call this process “channel shuffle”.

Moreover, a study by Ma et al. (2018) highlights several architecture-related practices that are counterproductive for inference speed. First, it is not advisable to have convolution layers with a large difference in the number of input and output channels. Second, using many group convolutions creates too many channels, which has a negative impact on memory consumption and thus on inference speed. Splitting the network into many parallel paths that are not identity paths can also reduce the ability to parallelize. Finally, element-wise operations such as “ReLU”, “AddTensor”, and “AddBias” also affect memory consumption and thus inference speed. Considering all these aspects, the authors design two types of blocks that use both “channel shuffle” and another type of operator called “channel split”. This new operator simply splits the input channels into two separate paths. The architectures mentioned so far in this section can be modified to be efficiently applied to video-related tasks (Kopuklu et al., 2019).

While many works propose to apply some modifications to spatial convolutions, Wu et al. (2018a) propose a different approach by replacing spatial convolutions with another operation called “shift”. The principle is that each channel of the input tensor is shifted in an independent and learned spatial direction. Then, the shifted information is mixed across channels by means of 1×1 convolutions. This alternative has the advantage of mixing spatial information without additional computation, as opposed to spatial convolutions, and it is orthogonal to most other model optimizations, so it can be used in many circumstances. Even though it does not seem to have been used for semantic video segmentation, it might be interesting to try it in this context.

Fast transformers

We have already discussed the contribution of transformer-based architectures to improve model outputs quality in Section 4.4.1. However, such architectures, if designed efficiently, can also help improve quality for frame-wise tasks (Li et al., 2022a). A recent work on fast transformers for video segmentation presents an architecture based on sparse spatiotemporal transformers for video object segmentation, and claims that the method is also applicable to other tasks such as semantic segmentation (Duke et al., 2021). Just as convolutions can be made easier to compute using stride or dilation, self-attention layers can also be modified in this way. To achieve good results and to be robust to important temporal changes while remaining fast, the proposed model takes as input a fixed-size sequence consisting of the current frame and the previous frames as well as their respective segmentation map. This input is then fed into a 2D ResNet to produce embeddings, which are optionally summed to a positional embedding. The resulting tensor is then fed into a series of spatiotemporal self-attention modules with sparse connectivity. These modules consist of multi-head attention followed by a feed-forward layer, the two being interleaved with skip connections and normalization layers. The role of such modules is twofold. First, the final output of the sequence of modules gives features that incorporate temporal information. Second, in each layer, the “object affinity” tensors produced by the multi-head attention component are collected, so that the information about how to propagate segmentation information across frames is passed directly to the decoder part of the network, along with the final output of the sequence of modules and the current frame. The decoder part of the network, which is also a CNN, then computes the final segmentation

map for the current frame. The reason this architecture is fast is that the self-attention components of multi-heads are sparse. A given pixel does not have direct access to all other pixels, but only to a small fraction determined by a spatiotemporal pattern. However, because these self-attention layers are stacked, a dense connectivity between all pixels is built indirectly as depth increases, allowing long-range dependencies in the spatiotemporal domain at low computational cost.

The approach of Li et al. (2021) also aims at reducing the inference time using transformers, but in the context of semantic segmentation. This work uses a classical 2D CNN architecture as encoder and decoder, and the same kind of self-attention modules as the approach of Duke et al. (2021). However, instead of adding sparsity directly in the self-attention modules as in the previous paper, this time the sparsity intervenes at the point where the attention module is applied (similar to the methods discussed in Section 5.3.4). The idea is that in a typical scene, the segmentation difficulty is spatially inconsistent: there are some large regions that contain a single class, while other regions contain multiple objects of different nature. Therefore, the time-consuming operation of aggregating content from previous frames does not significantly help to improve quality in simple regions, while it can be useful to distinguish object boundaries in complex regions. Specifically, the features from the current frame are used to determine the complex regions. Then, each complex region detected in the current tensor is used as a query to find cues in the previous frames. Since each previous frame has produced features, they can be used as keys and values to perform the attention mechanism. However, to further avoid unnecessary computations, the attention between a given query region and a previous frame is not performed globally, but in a local region centered around the spatial position of the query and whose radius increases as the frames are temporally distant relative to the current time step. These mechanisms, once combined, allow the model to incorporate temporal information where it is most needed while remaining fast.

As we saw in Section 5.4.3, 3D convolutions can be factorized into simpler units to reduce the computation. The same can be said for spatio-temporal transformers as seen in the approach of Su et al. (2023). In this work, each frame is preprocessed independently by a CNN backbone. Then, the extracted features of adjacent frames must communicate to produce spatio-temporal features. However, creating an attention matrix to link pixels from features coming from multiple frames is very computationally intensive. To address this problem, “decoupled transformers” are introduced. The idea is to first take features belonging to frames of a fixed time window and gradually downsize the spatial resolution of the oldest features. Then, multi-head self-attention can be performed independently on the modified features, the process being faster on the oldest and thus smallest features. The low-resolution results are then upsampled to match the current frame results. To handle feature misalignments due to motion, the past features are passed through deformable convolutions (see Section 4.4). Once the features from different times are aligned in a common space, a 1D temporal transformer can be applied at each pixel location, mixing information across time. This process results in much less computation than directly applying attention to features from different frames. The method described above can be combined with another transformer branch whose goal is to aggregate current frame features from different backbone stages (concatenated with the result of the temporal branch corresponding to the current frame) and process them in a 1D fashion along the channel dimension. An object detection method has introduced “deformable transformers” (Zhu et al., 2021), which are very similar to the way “deformable convolutions” work (see Section 5.4). Notably, this principle has been used in a video panoptic segmentation method (Athar et al., 2023).

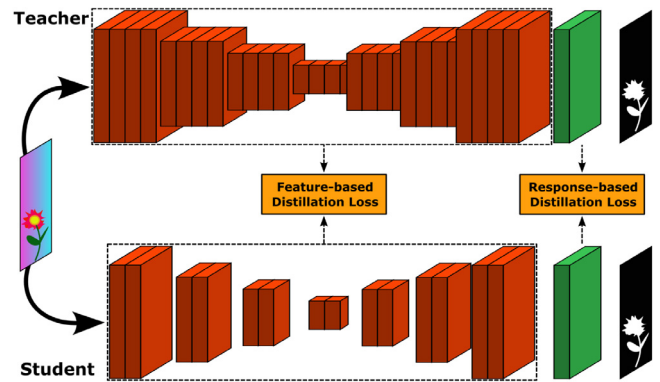


Fig. 12. A diagram illustrating the principle of feature-based and response-based knowledge distillation techniques.

5.5. Knowledge distillation

It is generally accepted that the more parameters a deep learning model has, the more knowledge capacity it has. With insufficient training, large models tend to learn the data “by heart” and thus overfit. However, given a sufficient amount of quality data, a large model can learn more complex functions than a small model. Even so, the complexity of the functions that a model can learn with a regular training strategy may be limited to a fraction of its theoretical knowledge capacity (Ba and Caruana, 2014).

To overcome this challenge, the first step is to create a model that is large enough so that it can solve the task at hand by learning complex features despite the lack of inductive bias of the training samples. The large capacity of this initial model improves its ability to learn complex features, but once they are learned, such capacity is no longer needed to store them. The second step is therefore to create a smaller “student” model that will receive the knowledge gathered by the larger model, called the “teacher” in this context. By setting the goal to mimic the behavior of the teacher, the student learns the same task as the teacher, but the training has a much stronger inductive bias and thus requires much less capacity to build up interesting features. This process is called “knowledge distillation” and allows us to create compact networks that behave the same way (or even better, if the downscaling has allowed some generalization) as their large teacher model, while having considerably fewer weights and thus being faster to compute (Phuong and Lampert, 2019).

There are many different knowledge distillation techniques (Gou et al., 2021). The first element that helps distinguish them is their type, referring to the material the student network is trying to copy. For example, in the 2D segmentation technique of Chen et al. (2020), the student learns to copy the output (or response) of the teacher network, making it a response-based knowledge distillation technique. There are also feature-based techniques where the student learns to copy some of the teacher’s features (Liu et al., 2019) or a latent representation of its features, created by an auto-encoder, which allows to overcome the models architecture differences (He et al., 2019). Those two types of knowledge distillation are illustrated Fig. 12. Differently, in relation-based knowledge distillation techniques (Yang et al., 2022), the student network does not learn to independently mimic the teacher’s output logits or intermediate features from single data samples, but rather to copy the relationship between the inputs and the network states across multiple data samples as shown in Fig. 13. Finally, Liu et al. (2019) show that a single method can fuse several types of distillation types.

In addition to the type, what can differ is the mode of distillation. Most methods use an offline distillation scheme, meaning that the teacher network is fully trained before knowledge transfer and its weights are frozen during the transfer. However, the training of

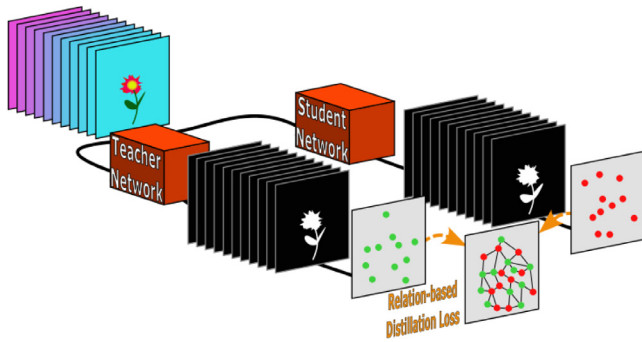


Fig. 13. A diagram illustrating the principle of the relation-based knowledge distillation techniques.

both models can be performed simultaneously, which is called “on-line distillation”. Finally, a single network can teach itself by transferring knowledge from deep layers to early layers, which is called “self-distillation”.

Knowledge distillation is a very powerful method that is used in many domains where fast inference is required. Thus, there are several examples of semantic video segmentation techniques that rely on it in different ways. Liu et al. (2020) illustrate the thin boundary between frame-wise and video segmentation techniques. Indeed, the goal of this work is to create a compact network that can perform temporally consistent semantic segmentation at inference time without processing more than one frame at a time. To achieve this, the model is intensively trained with multiple temporal constraints. However, there is a gap between small and large models in terms of the temporal consistency that the model can learn by itself. To narrow this gap, the compact model learns to mimic the behavior of a deeper model thanks to knowledge distillation.

Knowledge distillation can be applied not only to models with similar architectures, but also to models with significant structural differences (Carreira et al., 2018; Shimoda et al., 2019; Bai et al., 2022), which can help to train efficient but difficult-to-train designs. Holliday et al. (2017) shows that knowledge from multiple semantic segmentation architectures can be transferred to a unified network, resulting in a lightweight model that combines the diverse strengths of its teachers while being significantly faster than the sum of them. Differently, Hu et al. (2020) explain that the output of multiple shallow networks used in parallel can have as much representational power as one deeper network. In the case of semantic video segmentation, the idea is not to compute the multiple networks simultaneously, but to exploit the redundancy between successive frames by computing one shallow network per frame in a circular fashion (as explained in Section 5.3.5). To be efficient, this process requires that the features produced by each shallow network are complementary. To achieve this, and to build a strong representation across frames, knowledge distillation is used. Here, the teacher is a deep feature extractor model that the shallow students will try to replace. In order for them to learn complementary features, the deep features produced by the teacher are divided into as many groups as there are shallow networks, so that each student focuses on learning a different subset of features.

5.6. Network compression

One approach for reducing the memory footprint of a network is by modifying its parameters. The two most common approaches are either to reduce the number of parameters by pruning or to reduce the number of bits used to store each parameter, which is called quantization. In both cases, compressing a network is useful to make it suitable for devices with limited memory, such as mobile devices. For example, Iandola et al. (2016) take Alexnet, a deep image classification

model, and reduce its memory size by significantly pruning the model weights and quantizing the resulting weights from 32 bits each to 6 bits. At the end, they manage to reduce the number of parameters by 50 and the size of the model by 510 without affecting its performance. Moreover, Paupamah et al. (2020) show that reducing the memory size of models has other positive effects on image classification tasks, such as reducing inference time and increasing robustness to overfitting. In this section, we will discuss how these strategies can be applied to semantic video segmentation.

5.6.1. Weights quantization

Weight quantization seems to be very effective when applied to rather simple tasks such as classification. Some works design networks that achieve an astonishing memory reduction, using one bit per parameter (namely “binary networks”) (Courbariaux and Bengio, 2016; Rastegari et al., 2016) that can be orders of magnitude faster than their unquantized version while maintaining good quality. However, segmentation is usually considered a more challenging task and sometimes requires more complex layers with, for example, upsampling or dilated convolutions. Nevertheless, some work has been done in the area of quantization of image segmentation networks (Vogel et al., 2019; Miyama, 2021; Ahamad et al., 2021). The results show that it is possible to reduce the size of the original 32-bit parameters to 8 or 3 bits, depending on the models, without losing much of the original quality.

Unfortunately, to the best of our knowledge, there are no studies on the quantization of semantic video segmentation models, which would be even more challenging than its 2D alternative. Nevertheless, the quantization of a 3D UNet model designed for volumetric image segmentation demonstrates the plausibility of doing so on models adapted to video content, since this format is also three-dimensional (Askari-Hemmat et al., 2019).

5.6.2. Pruning

Pruning is particularly well suited to networks containing fully connected layers. The first reason is that these layers contain the most parameters compared to convolutional layers. Second, as demonstrated by Iandola et al. (2016), these layers have such densely connected architectures that many neurons can be removed without degrading the most important connections. This is illustrated Fig. 14a.

In the case of fully convolutional networks, this is more complex because removing a neuron has a more significant impact on the behavior of the network. A convolutional neuron is part of a feature extractor that is applied to each position of the input tensor. Moreover, even though pruning reduces the memory footprint significantly, most hardware resources are not designed to efficiently load the resulting unbalanced convolutional filters. Consequently, the reduction in computation time is limited. Nevertheless, Shimoda et al. (2019) propose a way to prune a FCNN network with efficient inference time reduction. To do this, the pruned network is guided by a dense network via knowledge distillation (see Section 5.5). More importantly, the pruning strategy (which is illustrated Fig. 14c) consists of taking each filter and sorting its weights by their absolute value, then pruning a fixed percentage of the smallest ones. As such, since each filter is affected by the pruning in the same way, there is no imbalance in the load and the inference time can be improved.

In a convolutional layer, there are typically multiple filters, each of which produces a channel output of variable importance compared to others. Thus, instead of pruning individual weights, one could remove entire channels/filters as depicted in Fig. 14b. This principle is applied in several methods such as the one from Chen et al. (2022) where a network is optimized for both classification and 2D segmentation tasks. The authors use the scaling factor of the batch normalization layers to determine the importance of each channel. By applying an L1 normalization to the scaling factors, they obtain sparse results and can therefore drop the filters/channels that have an L1 normalized scaling

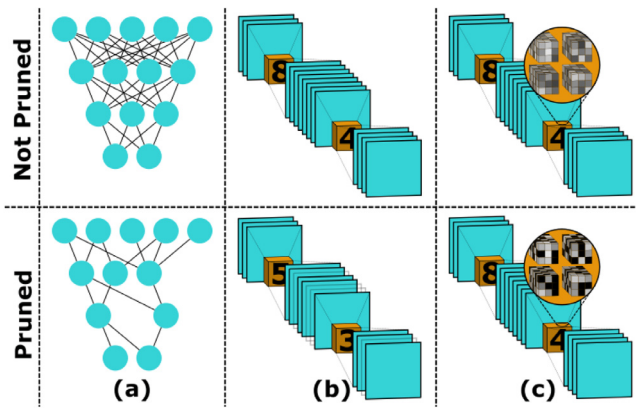


Fig. 14. There are several ways to prune a network. In networks with hidden layers (a) such as [Iandola et al. \(2016\)](#), pruning can be done in those layers by removing the least useful neurons and connections. In fully convolutional networks, pruning can be done in two main ways: The first (b), is to remove entire feature maps and their respective kernels instead of removing individual neurons as in [Chen et al. \(2022\)](#), [Huang et al. \(2018b\)](#). The other method (c), is to set a fixed fraction of the weights in each convolutional filter to zero as in [Shimoda et al. \(2019\)](#).

factor close to 0. A similar procedure is used by [Huang et al. \(2018b\)](#) to accelerate the classification task. There, the importance of an input channel is determined by the filters that use it: if the L1 norm of the weights treating that channel is small compared to the weights treating the other channels, then that input channel is no longer needed and the filter that produces it can be dropped. To reinforce this phenomenon, the authors use group convolutions (see Section 5.4.2) and encourage the filters belonging to the same group to use the same input channels thanks to a group-lasso regularizer ([Zhang et al., 2020](#)). This way, each group takes only a subset of the remaining channels and thus performs less intensive computations.

Finally, [Bai et al. \(2022\)](#) present a pruning mechanism that does not lighten the memory footprint of the network, but is specifically designed to reduce the inference time of the segformer architecture ([Xie et al., 2021](#)), a transformer-based 2D semantic segmentation network. First, the authors observed the magnitude variation of the segformer's neurons across different instance inferences. The results show that while some neurons are informative most of the time, others are highly informative only in some cases, and finally there are neurons that are almost never informative regardless of the context they are in. This motivates the development of a gating mechanism that uses both the current input and the current linear layer parameters to create a mask of which neurons to keep and which to discard. The mask is then applied to both the current linear layer parameters and the input of the next linear layer. This saves time by avoiding the computation of currently unnecessary neurons and by telling the next layer which output to ignore. The results show that the neurons previously identified as never informative are indeed always masked, while those whose usefulness depends on the context are dynamically activated only when necessary.

Similar to quantization, so far there is not much work on pruning applied to semantic video segmentation. Given the effectiveness of pruning on 2D tasks, its application to semantic video segmentation is an interesting avenue for future work.

5.7. Network search

The most suitable neural architecture can be very different depending on the task. The construction of efficient architectures often relies on manual trial and error, which can be very slow and tedious, especially given the possible options available. A field of study called "Network Search" focuses on designing better search strategies to determine the optimal architecture for a given task. There are several

search strategies for exploring the space of possible networks. [Yang et al. \(2018\)](#) suggest a method that starts the search with a pre-trained network whose complexity is above the desired budget. To optimize the latency of the model while maintaining the quality of its results, the method works iteratively. At each iteration, several candidate networks of identical complexity are generated by removing filters from the current model. Each candidate is then briefly fine-tuned and evaluated. The candidate with the best accuracy is then defined as the new current model for the next iteration. Once the target complexity is reached, the iterative process stops and the resulting model is further fine-tuned.

In contrast, [Tan et al. \(2019\)](#) follow a reinforcement learning strategy with an RNN as the controller (i.e., the candidate architecture generator). It also has the peculiarity of factorizing the search space into modules in a hierarchical way. The idea is that if each layer of a network is considered independently, the search space defined by the set of possible combinations of each layer variation is too large to be explored efficiently. Thus, by grouping some layers together so that they adopt the same variations at the same time, one can greatly reduce the search space. However, depending on their position in the network, layers have different roles and data flows, so it is not necessarily a good idea to group layers with these differences together. In this work, the term "hierarchical" describes the fact that the groups formed by the factorization are made in such a way that the layers inside them share a close role and data flow.

Yet another approach is based on a gradient optimization method ([Liu et al., 2018](#)). The strategy trains a one-shot model (also called a supermodel) that contains all variations of the search space and jointly relaxes this space in a continuous manner so that the network progressively selects the wisest operations. This method takes advantage of parameter sharing to greatly reduce redundant training, in contrast to strategies that independently train nearly identical models. Finally, network search can be used directly to obtain a model specifically designed for semantic video segmentation ([Nekrasov et al., 2020](#)). The authors use a reinforcement-based strategy with a two-layer LSTM network as the controller, as suggested by [Tan et al. \(2019\)](#). Here, only a part of the network is modified, which reduces the search space. The complexity of the search is further reduced by using factorization into cells that share the same internal architecture. At the end, the method converges to architectures that do not need to perform slow optical flow estimation and are thus much faster.

5.8. Activation functions

Activation functions are an essential part of modern neural architectures because they introduce nonlinearities in the behavior of the network. Depending on the task at hand, the architecture of the model, but also on the network layer, the more appropriate activation function can vary. The choice of activation function for each network layer has an impact on the quality of the network results, but can also have an impact on its inference time. Some of the most popular activation functions are the Sigmoid function and the Rectified Linear Unit (ReLU) function. Since its introduction, ReLU has led to many variants such as Relu6 ([Krizhevsky, 2010](#)), Leaky ReLU, Exponential Linear Unit (ELU) ([Clevert et al., 2016](#)), Parametric ReLU ([He et al., 2015](#)), Gaussian Error Linear Unit (GELU) ([Hendrycks and Gimpel, 2016](#)), and Swish ([Ramachandran et al., 2017](#)). These variants can fix some drawbacks of ReLU, such as the dead neuron problem or the fact that the function is not continuously differentiable. However, while the original function is computationally simple, these improvements also come at the cost of increased complexity. To address this, [Avenash and Viswanath \(2019\)](#) propose a low-complexity version of the swish activation function that outperforms ReLU in some image- and video-related tasks. Similarly, [Courbariaux et al. \(2015\)](#) introduce the Hard Sigmoid function, which is a discrete and thus faster version of the Sigmoid.

6. Available datasets

Semantic video segmentation is a complex task that ideally requires a large amount of training data. In this section, we will describe the existing datasets for such a goal. The description of the datasets is summarized in Table 1.

- **Camvid (Cambridge-driving Labeled Video Database)** (Bros-tow et al., 2008) contains five video sequences from the dash-board viewpoint of a driving car. In four of the five sequences, one frame per second of this 30fps footage is densely annotated with 32 semantic classes. In the remaining sequence, one out of every two frames is annotated with semantic classes. In total, the dataset contains over ten minutes of video and up to 701 annotated frames.
- **Cityscapes** (Cordts et al., 2016) is a dataset containing videos of urban scenes from 50 cities, taken during daytime, in good or average weather, and in different situations and seasons. These videos are annotated with 30 semantic classes belonging to 8 groups. The first part of the dataset contains a selection of 5000 highly diverse video snippets taken in 27 of the aforementioned cities. The 20th frame of each of these 30 frame (1.8s) snippets is annotated with precise pixel-wise annotations. The second part of this dataset contains the remaining footage from the 23 other cities, with roughly annotated frames every 20 s or every time the vehicle moves 20 meters forward, for a total of 20,000 coarsely annotated frames. The annotations are made using polygons, and overlaps are not tolerated, even when the footage contains transparent or sparse objects such as windows or tree leaves.
- **Cityscapes-VPS** (Kim et al., 2020) is an extension of the Cityscapes dataset for panoptic segmentation. In fact, among the 5000 finely annotated video snippets from the original dataset, 500 of them also contain instance labels that make them usable for panoptic segmentation purposes. This extension further enriches these 500 clips by providing for each snippet a panoptic annotation of the 5th, 10th, 15th, 25th, and 30th frames in addition to the original annotated 20th frame. This provides additional useful material for semantic segmentation as well.
- **NYUDepth** (Silberman and Fergus, 2011) is a panoptic video dataset of indoor scenes recorded by a Microsoft Kinect camera. 64 scenes, classified into 7 scene types, were recorded at a frame rate of 20 to 30fps. Out of a total of 110,964 frames recorded, including both RGB and depth estimation, 2347 are associated with dense labels belonging to a set of more than 1000 classes, which translates to approximately one labeled frame every 2 to 3 s.
- **NYUDepth V2** (Nathan Silberman and Fergus, 2012) is the same kind of dataset as NYUDepth and contains 464 additional scenes of 26 types. This makes up to 408,473 new frames, including 1449 densely labeled ones.
- **Gatech** (Raza et al., 2013) is a dataset originally designed for understanding the 3D geometric structure of outdoor video scenes, but it can also be used for semantic segmentation of videos. It contains 160 videos obtained from YouTube or obtained by filming while walking and driving in urban areas. The videos have different aspect ratios and resolutions, and each contains 60 to 400 frames. More than a hundred of them (20,000 frames) are fully annotated with 6 general labels (mix, sky, ground, solid, porous, and object). However, because the manual ground truth labeling was done on a superpixel scale rather than a pixel-wise scale, the annotations contain under-segmentation errors.
- **Freiburg Forest** (Valada et al., 2016) is a multispectral and multimodal video dataset for semantic segmentation of unstructured environments. It contains sequences captured by an autonomous robot that traveled 4.7 km per day in the forest during 3 different days. Thus, the dataset contains 15,000 frames obtained by

subsampling the original 20 Hz content at 1 Hz. Among these frames, 366 were manually annotated at the pixel level with 6 labels (obstacle, trail, sky, grass, vegetation, and void).

- **Indian Driving Dataset (IDD)** (Varma et al., 2019) is similar to Cityscapes, containing 182 driving sequences that take place in two Indian cities and their respective suburbs. It provides scenes that are much less structured and contain very different traffic than Cityscapes. To account for the scene diversity and label ambiguity, additional classes were created, resulting in a total of 34 labels organized in a 4-level hierarchy. The videos, which are mostly at 1024p resolution, are annotated at a rate that depends on the interest of the scene. In order to densely annotate the 10,004 selected frames, the team first replicated some of the Cityscapes annotations to reduce the domain gap between the two datasets.
- **GTA5** (Richter et al., 2016) is a synthetic dataset for semantic segmentation. The content represents a photorealistic view of a driving car perspective in urban scenes from the video game “Grand Theft Auto 5”. To select the 24,966 frames to densely label, the team recorded one frame for every 40 frames generated by the software during a game session. The 19 Cityscapes-compatible labels were then applied to the frames in a semi-automatic process using data extracted from the game’s shaders.
- **SYNTHIA-Seqs** (Ros et al., 2016) is a synthetic dataset containing four photorealistic video sequences in which each frame is densely annotated with 13 potential semantic labels. More specifically, the 200,000 annotated frames represent the multi-view perspective of a car driving through a Unity-rendered city that contains dynamic objects in addition to static elements.
- **VSPW** (Miao et al., 2021) is a semantic segmentation dataset containing 3337 videos from YouTube, ranging from 2 to 10 s in length, with high resolution (720P to 4K), describing a wide range of real-world scenes taking place either indoors or outdoors. The selected videos contain moderate object and camera motion and are sampled at 15fps, resulting in 239,934 densely annotated frames with 124 labels. This dataset also has a panoptic extension called VIPSeg.
- **KITTI-STEP** (Weber et al., 2021) is a panoptic segmentation dataset containing 50 videos of streets from the perspective of a driving car. The 1280 × 384 videos are sampled at 10 fps (19,103 frames in total) and densely annotated with 19 semantic classes.

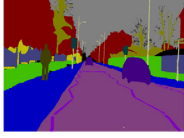







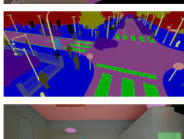

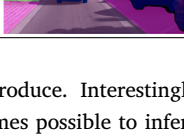
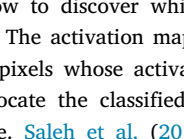
6.1. Annotation sparsity

Semantic segmentation is a task that aims to create precise boundaries between objects of different nature. Learning such a task usually requires a large amount of training data with dense and precise annotations. Annotating a 2024×2048 frame takes on average 1.5 h (Cordts et al., 2016). However, with semantic video segmentation, the volume of data to annotate is very high, as one second of content represents dozens of frames. To address this problem, one solution could be to create synthetic data to simultaneously generate the segmentation maps (Richter et al., 2016; Ros et al., 2016). However, models trained on such data may struggle to adapt to real examples due to the domain gap. As we can see, natural annotated data is quite necessary, so the solution commonly used to produce such data is to annotate only one frame at a regular interval (Cordts et al., 2016). Among the annotated frames, only a small fraction is densely annotated and the rest is only coarsely annotated. Because of this reality, semantic video segmentation methods must adopt strategies to effectively train on this sparsely annotated data.

Although the training of the segmentation task requires dense pixel-wise annotations, this is not the case for every vision task. For example, the multi-class video classification task requires only video-wise or frame-wise annotations, which are much more common because they

Table 1

A table summarizing key information about the available datasets.

| Dataset | Example | Year | Fields | Semantic classes | Videos | Format | FPS | Frames | Labeled frames | Label per second |
|-----------------|---|------|--------------------|------------------|----------|-----------------------------|----------|---------|----------------|------------------|
| Camvid |  | 2008 | Semantic | 32 | 5 | 960×720 RGB | 30 | 18 202 | 701 | 1 to 15 |
| Cityscapes |  | 2016 | Semantic, Panoptic | 30 | 25,000 | 1024×2048 RGB | 17 | 150,000 | 20,000 + 5,000 | 0.55 |
| Cityscapes-VPS |  | 2019 | Panoptic | 30 | 500 | 1024×2048 RGB | 17 | 15 000 | 3000 | 3.3 |
| NYUDepth |  | 2011 | Panoptic | 1000+ | 64 | 640×480 RGBD | 20 to 30 | 108,617 | 2347 | 0.33 to 0.5 |
| NYUDepth V2 |  | 2012 | Panoptic | 1000+ | 464 | 640×480 RGBD | 20 to 30 | 408,473 | 1449 | 0.33 to 0.5 |
| Gatech |  | 2013 | Semantic | 6 | 100 + 60 | 320×480 to 600×800 RGB | >10 | 34 000 | 20,000 | >10 |
| Freiburg Forest |  | 2016 | Semantic | 6 | 3+ | 1024×768 RGBD + others | 1 | 15,000 | 366 | ≈0.02 |
| IDD |  | 2018 | Panoptic | 34 | 182 | 1920×1080 RGB or less | ≈15 | 10,004 | 10,004 | ≈0.88 |
| GTA5 |  | 2016 | Semantic | 19 | unknown | 1914×1052 RGB | 1.5 - 3 | 24,966 | 24,966 | 1.5–3 |
| SYNTHIA-Seqs |  | 2016 | Semantic | 13 | 5 | 960×720 RGBD | 5 | 200,000 | 200,000 | 5 |
| VSPW |  | 2021 | Semantic | 124 | 3,536 | 1280×720 to 3,840×2,160 RGB | 15 | 239,934 | 251,633 | 15 |
| KITTI-STEP |  | 2021 | Panoptic | 19 | 50 | 1280×384 RGB | 10 | 19,103 | 19,103 | 10 |

are easier to produce. Interestingly, once a classification model is trained, it becomes possible to infer it on an example and then follow the gradient flow to discover which features influence the classification decision. The activation maps corresponding to these features then show the pixels whose activation is the strongest. This makes it possible to locate the classified objects and determine their approximate shape. [Saleh et al. \(2017\)](#) exploit this and combine the obtained spatial information of the current frame with the temporal information provided by the optical flow estimations of the previous

frames. The obtained spatio-temporal features are then further combined with deeper spatial features to produce a semantic segmentation map of the current frame. Similarly, [Shimoda and Yanai \(2020\)](#) suggest up-sampling frame-wise classifier features of different scales and subtracting features corresponding to different objects in order to create maps corresponding to specific semantics. These enhanced features are then fed to a CRF to produce precise segmentation maps.

Another approach is to use knowledge distillation, as described in Section 5.5 ([Chen et al., 2020](#)). There, a teacher network is trained on the available labeled frames. Then, the same network is used to produce

segmentation trials of the unlabeled frames. To improve the robustness of the predictions, each frame is segmented several times using data augmentation and the different results are then averaged. The improved segmentation candidates of the unlabeled frames can then be added to the training set of the teacher network for further training. By training on this additional data, which has been made more robust than the initial naive predictions, the teacher can improve and thus produce new segmentation maps of better quality for the unlabeled frames. This process is repeated for several iterations, with each iteration producing both a more powerful teacher network and better quality pseudo-labels. At the same time, a student network learns the segmentation task using both the labeled frames and the frames with pseudo-labels generated by the teacher. Finally, the student is fine-tuned using only the labeled frames.

End-to-end video segmentation methods typically require larger amounts of task-specific training data than their image counterparts. On the other hand, semi-automatic video instance segmentation techniques are designed to efficiently propagate an initial mask to the next frames of a video sequence (Badrinarayanan et al., 2010; Wang et al., 2019b; Lu et al., 2020; Cheng and Schwing, 2022; Qin et al., 2023), and we have seen in the Sections 4.2 and 5.3.1 that mask propagation is also possible for video semantic segmentation tasks. Therefore, it may be advantageous to adapt 2D segmentation methods to video (Miksik et al., 2013; Cheng et al., 2023), especially with the use of VIS methods, as demonstrated by Cheng et al. (2023).

The latter work indeed presents a universal temporal propagation model that can convert any 2D segmentation method to work with video without the need for additional task-specific training data. This propagation model, which produces more accurate results than applying the 2D segmentation model to each frame independently, is bidirectional and makes use of several tools. First, XMem (Cheng and Schwing, 2022), a model described in Section 4.6, is slightly modified and trained so that it can propose a class-agnostic segmentation candidate for a given frame based on the segmentation results of past frames. Another tool used by this method is the “in-clip consensus”, which takes as input a fixed number of future frames in addition to the current one. These input frames are then segmented using the 2D model and aligned to the current frame space using the modified XMem. The aligned set of segmentation candidates can then be compared by measuring the mIoU between all possible pairs of candidates. The segments with the most and highest mIoU ratings are selected as the consensus candidate. In summary, to segment a given frame, the model computes a mask from past frames with the modified Xmem network and merges it with the result of the in-clip consensus from future frames.

6.1.1. Frames prediction

As described in Section 4.3.1, some semisupervised object segmentation methods use optical flow estimation to propagate a ground-truth segmentation map to the next frames. Therefore, one might think that this technique could be used to generate new training examples to help train semantic segmentation methods that do not rely on optical flow. However, the quality of the training examples generated in this way is directly related to the quality of the optical flow estimation. If the estimation is not sufficiently precise, the generated examples can become counterproductive for training due to misalignments between the propagated labels and the associated video content. Furthermore, classical optical flow methods do not handle occlusions and disocclusions. Hopefully, there is an area of research that focuses on predicting the next frame of a given sequence. More specifically, it can compute the likely deformations of the current frame that will produce the next frame while handling occlusions and disocclusions. The same deformations can then be applied to the current segmentation map to generate the next one. Using this technique for training purposes is interesting because there is no risk of misalignment between the movements in the video and the deformations of the annotation sequence (Zhu

et al., 2019). However, the warping done on the frames can introduce interpolation-induced approximations at the edges of the objects. To reduce the impact of these small errors on the training performance of the produced examples, the authors propose to reduce the label weights where the destructive deformations are applied. Finally, by training their model with the synthetic examples in addition to the ground truth examples, they achieve better results than by training it with the ground truth examples alone.

The frame prediction technique can also be used directly for a semantic video segmentation task, as demonstrated by Jin et al. (2017). Indeed, a frame prediction network does not need label annotations to be trained. Once the network is trained, it can be used to provide spatiotemporal features from the previous frames and therefore work as a better performing alternative to optical flow estimation.

7. Discussion

Throughout this survey, we have discussed about various deep learning methods that have been used or could be used to improve the quality or the efficiency of video semantic segmentation. As we saw in Section 3, although semantic segmentation methods achieve impressive performance on image datasets, they suffer from three main limitations when dealing with video:

- **Computational complexity:** Computationally intensive methods can be used in many image-based practical applications, but the same cannot be said for video-based practical applications because the associated data flow can rarely be processed in a reasonable time. This issue is particularly important since most of the induced computations are temporally redundant due to the nature of video content.
- **Temporal consistence:** Video semantic segmentation results must be temporally consistent to be applicable to most real-world applications, but this is not possible with framewise methods.
- **Temporal cues:** Image methods rely only on spatial cues to perform segmentation. However, video content contains temporal cues that should be exploited to improve the segmentation results.

To address these challenge, some methods have proposed to use sequence-specific tools to deal with the temporal dimension (see Section 4.2). This was done with RNN, then with gated units such as LSTM and GRU. Improvements have been made to adapt these units to image processing (convGRU, convLSM), but they are decreasingly used in recent work on video semantic segmentation. The reason may be that they cannot be parallelized, but more importantly, that they have a limited ability to retain long-range information, in contrast to newer and more dynamic architectures such as transformers.

Optical flow estimation has been widely used to support the development of video segmentation methods (see Section 4.3). It can be used to improve segmentation accuracy and consistency by warping features from different time steps, or it can help reduce the overall inference time and redundant computation by propagating features or segmentation masks from keyframes. Furthermore, optical flow itself can be used as a feature, since it provides spatiotemporal edges that are complementary to spatial ones, and it provides semantic cues by revealing the way objects move. Finally, optical flow can be computed with hand-crafted methods, and if it is based on deep learning, it can be trained beforehand when the segmentation training data is scarce, or it can be trained together with the segmentation in order to make both methods benefit from each other. However, optical flow also has some drawbacks. For example, it is typically not robust to occlusion and complex/fast motion. Also, when trained separately from the segmentation, it can suffer from a domain gap that reduces the accuracy. As for keyframe propagation, some semi-supervised instance segmentation methods are more efficient than optical flow warping (see Section 6.1). Finally, keyframe methods are not ideal because although they reduce the overall inference time, the inference time

is inconsistent between keyframes and non-keyframes, which is not suitable for many real-world applications.

On another hand, some early work adapted image models to video by converting 2D convolution kernels to 3D convolution kernels (see Section 4.1). However, a limit was soon reached due to the induced computational overhead. Depending on the task at hand, it has been shown that using 3D convolutions only in some parts of the network can sometimes be sufficient to gain accuracy and/or temporal consistency. In addition, as we saw in Section 5.4, it is possible to reduce the computation of 3D convolutions by factorizing kernels or by using group-wise convolutions. However, convolutions in general suffer from a dilemma that becomes even more pronounced when it comes to 3D convolutions: to perform well, a network must first capture abstract semantics thanks to a large receptive field, which can be built by stacking many convolutional layers at the cost of heavy computation. To enlarge the receptive field at reasonable computational cost, researchers have introduced techniques such as pooling, stride, and dilation rates, which work by downsampling the features fed to each layer. However, low-resolution features lack the details required by the network to produce accurate, and thus qualitative, results. A large branch of research has therefore focused on solving this crucial dilemma: Multiscale methods (see Section 5.1) provide ways to generate features of different scales in parallel, to capture both features containing details and large scale features that increase the receptive field (pooling pyramid, inception layers). Alternatively, most state-of-the-art methods employ ways to make deep features communicate with features from early layers (see Section 5.2). In this way, shallow features are used to produce deeper features, but also later to restore some spatial detail to deep features that have become coarse in the process of acquiring meaningful semantics. Finding ways to create efficient convolutions (including 3D convolutions) is a promising research direction, since convolutions are still used in the vast majority of state-of-the-art video segmentation methods, at least in the backbones. The same can be said about finding links between features of different scales, because no matter how they are created (convolutions, transformers), mixing them in an effective way is still a challenge, while being at the core of the segmentation task.

For some time, graphs have been used either for post-processing or as part of the networks themselves (see Section 4.5). They are useful for imposing priors on the input data, constraints on the output, and for modeling the structure of a video. However, they are usually carefully designed for specific applications, which does not follow the trend of creating architectures that can be easily adapted to multiple tasks. Nevertheless, they continue to be used and improved, especially in recent work introducing graph neural networks.

The advent of dynamic methods and attention has led to major advances in the field of video semantic segmentation (see Section 4.4). There is a wealth of new tools (deformable convolutions, dynamic convolutional kernels, transformers) that can be applied to whole feature tensors or to parts of them (channel attention, spatial attention, temporal attention). Attention is a powerful tool for creating links between distant feature elements, allowing for large receptive fields and clever multi-scale feature mixing. Such discoveries have also led to the rise of query-based methods, which aim to become universal methods that can perform different tasks without changing the architecture. These attention-based methods have proven their effectiveness and offer promising prospects, so work should continue to find what are the most effective tools among all the new ones. Recent methods, while providing high accuracy results, often face some limitations in terms of network size, training difficulty, and computation required to perform inference. Great improvements have already been made (sparse attention, local attention, factorized attention), but the efforts should continue in order to obtain effective models that could be used for various real-world applications.

In general, recent methods focus improving the quality of the segmentation results. However, rare are the works that consider other

constraints. In fact, there are several challenges and limitations other than accuracy that limit the real-world applications of such methods. First, some methods claim to be applicable to video without evaluating the temporal consistency of the segmentation results, which is essential for some applications. Inference time is obviously a major drawback of many current methods, but is quite rarely mentioned, and is difficult to compare when methods are not evaluated on identical platforms. The consistency of the inference time may also affect the possibility of using a method for a given task. The model's memory footprint, memory usage, computational load, and energy consumption during inference can be a hindrance for applications on embedded systems or frugal projects. As some models progress toward becoming universal architectures, it becomes increasingly important to simplify the ability to train or tune a given model for a specific task. This includes the complexity of the structural changes required to adapt the architecture to a given task, the simplicity of finding the right combination of hyperparameters, the computational power and time required to train the network, the amount of annotated training data required, and the ability to handle unbalanced data. Binary segmentation maps may not be sufficient for some real-world applications that require knowing the uncertainty of the predictions. Finally, while almost all recent methods test their effectiveness on short videos, some applications may require consistent results on longer sequences.

Work has been done and should be continued to assess these many challenges. Loss functions are important because they define the goals and constraints associated with a given task (see Section 4.8). Some commonly used losses have shown their limitations and could be replaced by one or more losses that better formalize the goals and lead to more effective training. While losses are often associated with the last layer of a model, it has also been shown that applying losses to shallower layers can help guide the network towards better training. Moreover, since losses are only applied during training, they can improve results without adding any delay to inference. As discussed above, many methods achieve great results at the cost of heavy architectures, so the usefulness of knowledge distillation will only increase as it offers the possibility of transferring capabilities that emerged in heavy models to compact architectures (see Section 5.5). In the same vein, quantization (see Section 5.6.1) and pruning (see Section 5.6.2) allow to lighten trained models to some extent with minimal impact on their performance. Pruning is quite difficult to apply to fully convolutional architectures, but may make a comeback with the rise of transformer-based models, although this is limited by the need for retraining.

Activation functions are at the core of the effectiveness of deep learning architectures (see Section 5.8). However, some activation functions can be computationally intensive and could be replaced by lighter alternatives. In addition, not every activation function is adapted to every part of the network. In order to use them to their full potential and to find the right balance between accuracy and computational overhead, work in this field should be pursued. Finally, there is some work on long-term memory (Section 4.6) and uncertainty estimation (Section 4.7), but much remains to be done. It is interesting to note that most of these research fields are orthogonal to methods designed to improve accuracy, and also orthogonal to each other, suggesting that they could be used together to improve the applicability of models to real-world constraints.

On another note, some recent architectures include many hyperparameters that can be very difficult to tune for a given task. In Section 5.7) certain works addressing this issue were discussed and offer promising results. Nevertheless, in our opinion, such tools should only be used to improve the results of real-world applications. When used in a research context to improve the performance of a network that introduces a new method unrelated to hyperparameter tuning, network search can make comparisons with other methods unfair and affect reproducibility.

Finally, irrespective of the network architecture used, the importance of datasets should not be understated. In the context of video

semantic segmentation, datasets are still somewhat lacking. Dense labeling is often scarce, which affects the training capability. This problem is beginning to be addressed with the advent of better labeling workflows and the emergence of synthetic datasets. However, methods trained on synthetic data currently suffer from a domain gap when applied to non-synthetic examples, showing that there is still room for improvement in this area. Furthermore, most of the datasets for video semantic segmentation currently contain urban driving content, which, albeit well-suited to autonomous driving applications, represents a poor diversity for different use cases. Therefore, a future goal would be to create more diverse and accessible datasets.

8. Conclusion

In this survey, we have described the different mechanisms that can help improve the performance of semantic video segmentation approaches, and in particular deep learning approaches adapted for this task, either by reducing its inference time or by improving the quality of its results. The nature of these mechanisms is very diverse: some are based on modified operators, others on structural changes on the network architecture, or on specific training strategies.

In some cases, described improvements are orthogonal to others and could be combined to achieve even better results. Moreover, the respective goals of improving inference speed and the output quality may be complementary. Indeed, to be usable for real-world applications, deep learning approaches are often constrained by a minimum inference speed budget, which inevitably limits the ability to incorporate computationally intensive mechanisms to improve quality. Thus, finding ways to speed up a given network may open the possibility to add the aforementioned quality improving methods while respecting the inference speed budget.

We have also highlighted that while some techniques have been proven for years, the new state-of-the-art methods also show promising results that should be further investigated. Nevertheless, several challenges remain and semantic video segmentation is still a complex task with a growing application area, and research in this area has room for improvement.

CRedit authorship contribution statement

Quentin Monnier: Conceptualization, Resources, Writing – original draft, Visualization. **Tania Pouli:** Conceptualization, Writing – review & editing. **Kidiyo Kpalma:** Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

Responsible for ensuring that the descriptions are accurate and agreed by all authors. This work is supported in part by Region Bretagne under grant ARED No. 2853

References

- Aakerberg, A., Johansen, A.S., Nasrollahi, K., Moeslund, T.B., 2022. Semantic segmentation guided real-world super-resolution. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*. pp. 449–458.
- Ahamad, A., Sun, C.C., Nguyen, H.M., Kuo, W.K., 2021. Q-SegNet: Quantized deep convolutional neural network for image segmentation on FPGA. In: 2021 International Symposium on Intelligent Signal Processing and Communication Systems. ISPACS, pp. 1–2. <http://dx.doi.org/10.1109/ISPACS51563.2021.9650929>.
- An, S., Liao, Q., Lu, Z., Xue, J.H., 2023. Dual correlation network for efficient video semantic segmentation. *IEEE Trans. Circuits Syst. Video Technol.* 1. <http://dx.doi.org/10.1109/TCSVT.2023.3298644>.
- AskariHemmat, M., Honari, S., Rouhier, L., Perone, C.S., Cohen-Adad, J., Savaria, Y., David, J.P., 2019. U-net fixed-point quantization for medical image segmentation. In: Zhou, L., Heller, N., Shi, Y., Xiao, Y., Sznitman, R., Cheplygina, V., Mateus, D., Trucco, E., Hu, X.S., Chen, D., Chabanas, M., Rivaz, H., Reinertsen, I. (Eds.), *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis and Hardware Aware Learning for Medical Imaging and Computer Assisted Intervention*. Springer International Publishing, Cham, pp. 115–124.
- Athar, A., Hermans, A., Luiten, J., Ramanan, D., Leibe, B., 2023. TarViS: A unified approach for target-based video segmentation. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 18738–18748. <http://dx.doi.org/10.1109/CVPR52729.2023.01797>, URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.01797>.
- Athar, A., Mahadevan, S., Ošep, A., Leal-Taixé, L., Leibe, B., 2020. SEm-seg: Spatio-temporal embeddings for instance segmentation in videos. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, pp. 158–177.
- Avenash, R., Viswanath, P., 2019. Semantic segmentation of satellite images using a modified CNN with hard-swish activation function. In: VISIGRAPP.
- Ba, J., Caruana, R., 2014. Do deep nets really need to be deep? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (Eds.), *In: Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc.
- Badrinarayanan, V., Galasso, F., Cipolla, R., 2010. Label propagation in video sequences. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 3265–3272. <http://dx.doi.org/10.1109/CVPR.2010.5540054>.
- Bai, H., Mao, H., Nair, D., 2022. Dynamically pruning segformer for efficient semantic segmentation. In: ICASSP 2022 – 2022 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 3298–3302. <http://dx.doi.org/10.1109/ICASSP43922.2022.9747634>.
- Ballas, N., Yao, L., Pal, C., Courville, A.C., 2016. Delving deeper into convolutional networks for learning video representations. In: Bengio, Y., LeCun, Y. (Eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*.
- Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5 (2), 157–166. <http://dx.doi.org/10.1109/72.279181>.
- Borghuis, B.G., Tadin, D., Lankheet, M.J., Lappin, J.S., van de Grind, W.A., 2019. Temporal limits of visual motion processing: Psychophysics and neurophysiology. *Vision* 3 (1), <http://dx.doi.org/10.3390/vision3010005>.
- Borse, S., Wang, Y., Zhang, Y., Porikli, F., 2021. InverseForm: A loss function for structured boundary-aware segmentation. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 5897–5907. <http://dx.doi.org/10.1109/CVPR46437.2021.00584>.
- Boykov, Y., Veksler, O., Zabih, R., 2001. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (11), 1222–1239. <http://dx.doi.org/10.1109/34.969114>.
- Brostow, G.J., Fauqueur, J., Cipolla, R., 2008. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.*
- Brox, T., Malik, J., 2010. Object segmentation by long term analysis of point trajectories. In: *European Conference on Computer Vision*. URL: <https://api.semanticscholar.org/CorpusID:16608752>.
- Brox, T., Malik, J., 2011. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33 (3), 500–513. <http://dx.doi.org/10.1109/TPAMI.2010.143>.
- Canny, J., 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8 (6), 679–698. <http://dx.doi.org/10.1109/TPAMI.1986.4767851>.
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, pp. 213–229.

- Carreira, J., Pătrăucean, V., Mazare, L., Zisserman, A., Osindero, S., 2018. Massively parallel video networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, pp. 680–697.
- Chandra, S., Couprie, C., Kokkinos, I., 2018. Deep spatio-temporal random fields for efficient video segmentation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 8915–8924. <http://dx.doi.org/10.1109/CVPR.2018.00929>.
- Chang, J., Wei, D., Fisher III, J.W., 2013. A video representation using temporal superpixels. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2051–2058. <http://dx.doi.org/10.1109/CVPR.2013.267>.
- Chen, L.C., Lopes, R.G., Cheng, B., Collins, M.D., Cubuk, E.D., Zoph, B., Adam, H., Shlens, J., 2020. Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, pp. 695–714.
- Chen, X., Zhang, Y., Wang, Y., 2022. MTP: Multi-task pruning for efficient semantic segmentation networks. In: 2022 IEEE International Conference on Multimedia and Expo. ICME, pp. 1–6. <http://dx.doi.org/10.1109/ICME52920.2022.9859583>.
- Cheng, B., Choudhuri, A., Misra, I., Kirillov, A., Girdhar, R., Schwing, A.G., 2021. Mask2Former for video instance segmentation. *arXiv abs/2112.10764*.
- Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R., 2022. Masked-attention mask transformer for universal image segmentation. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1280–1289. <http://dx.doi.org/10.1109/CVPR52688.2022.00135>.
- Cheng, H.K., Oh, S.W., Price, B.L., Schwing, A., Lee, J.Y., 2023. Tracking anything with decoupled video segmentation. *arXiv abs/2309.03903*.
- Cheng, H.K., Schwing, A.G., 2022. XMem: Long-term video object segmentation with an atkinson-shiffrin memory model. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (Eds.), *Computer Vision – ECCV 2022*. Springer Nature Switzerland, Cham, pp. 640–658.
- Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y., 2014. On the properties of neural machine translation: Encoder-decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. Association for Computational Linguistics, Doha, Qatar, pp. 103–111. <http://dx.doi.org/10.3115/v1/W14-4012>.
- Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014.
- Clevert, D., Unterthiner, T., Hochreiter, S., 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In: Bengio, Y., LeCun, Y. (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings.
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 3213–3223. <http://dx.doi.org/10.1109/CVPR.2016.350>.
- Courbariaux, M., Bengio, Y., 2016. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or –1. *CoRR abs/1602.02830*.
- Courbariaux, M., Bengio, Y., David, J.P., 2015. BinaryConnect: Training deep neural networks with binary weights during propagations. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2. NIPS '15, MIT Press, pp. 3123–3131.
- Csurka, G., Volpi, R., Chidlovskii, B., 2023. Semantic image segmentation: Two decades of research. *Found. Trends Comput. Graph. Vis.* 14, 1–162, URL: <https://api.semanticscholar.org/CorpusID:253028117>.
- Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y., 2017. Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision. ICCV, pp. 764–773. <http://dx.doi.org/10.1109/ICCV.2017.89>.
- Dhanachandra, N., Manglem, K., Chanu, Y.J., 2015. Image segmentation using K-means clustering algorithm and subtractive clustering algorithm. *Procedia Comput. Sci.* 54, 764–771. <http://dx.doi.org/10.1016/j.procs.2015.06.090>, Eleventh International Conference on Communication Networks, ICN 2015, August 21–23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21–23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21–23, 2015, Bangalore, India. URL: <https://www.sciencedirect.com/science/article/pii/S1877050915014143>.
- Ding, M., Wang, Z., Zhou, B., Shi, J., Lu, Z., Luo, P., 2020. Every frame counts: Joint learning of video segmentation and optical flow. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 10713–10720. <http://dx.doi.org/10.1609/aaai.v34i07.6699>.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR abs/2010.11929*.
- Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P., Cremers, D., Brox, T., 2015. FlowNet: Learning optical flow with convolutional networks. In: 2015 IEEE International Conference on Computer Vision. ICCV, IEEE Computer Society, Los Alamitos, CA, USA, pp. 2758–2766. <http://dx.doi.org/10.1109/ICCV.2015.316>.
- Duke, B., Ahmed, A., Wolf, C., Aarabi, P., Taylor, G.W., 2021. SSTVOS: Sparse spatiotemporal transformers for video object segmentation. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 5908–5917.
- Elhassan, M.E.A., Yang, C., Huang, C., Mune, T.L., 2022. SPNet: Subspace pyramid fusion network for semantic segmentation. *arXiv abs/2204.01278*.
- Fang, Y., Wang, Z., Lin, W., 2013. Video saliency incorporating spatiotemporal cues and uncertainty weighting. *IEEE Trans. Image Process. Publ. IEEE Signal Process. Soc.* 23, 1–6. <http://dx.doi.org/10.1109/ICME.2013.6607572>.
- Fang, Y., Yang, S., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., Liu, W., 2021. Instances as queries. pp. 6890–6899. <http://dx.doi.org/10.1109/ICCV48922.2021.00683>.
- Fayyaz, M., Saffar, M.H., Sabokrou, M., Fathy, M., Huang, F., Klette, R., 2017. STFCN: Spatio-temporal fully convolutional neural network for semantic segmentation of street scenes. In: Chen, C.S., Lu, J., Ma, K.K. (Eds.), *Computer Vision – ACCV 2016 Workshops*. Springer International Publishing, Cham, pp. 493–509.
- Fragkiadaki, K., Zhang, G., Shi, J., 2012. Video segmentation by tracing discontinuities in a trajectory embedding. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1846–1853. <http://dx.doi.org/10.1109/CVPR.2012.6247883>.
- Gadde, R., Jampani, V., Gehler, P.V., 2017. Semantic video CNNs through representation warping. In: 2017 IEEE International Conference on Computer Vision. ICCV, IEEE Computer Society, Los Alamitos, CA, USA, pp. 4463–4472. <http://dx.doi.org/10.1109/ICCV.2017.477>.
- Gal, Y., Ghahramani, Z., 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: Balcan, M.F., Weinberger, K.Q. (Eds.), *Proceedings of the 33rd International Conference on Machine Learning*. In: Proceedings of Machine Learning Research, vol. 48, PMLR, New York, New York, USA pp. 1050–1059.
- Gao, R., 2021. Rethink dilated convolution for real-time semantic segmentation. *CoRR abs/2111.09957*.
- Gonda, F., Wei, D., Parag, T., Pfister, H., 2018. Parallel separable 3D convolution for video and volumetric data understanding. *CoRR abs/1809.04096*.
- Gou, J., Yu, B., Maybank, S.J., Tao, D., 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.* 129 (6), 1789–1819. <http://dx.doi.org/10.1007/s11263-021-01453-z>.
- Grammatikopoulou, M., Sanchez-Matilla, R., Bragman, F., Owen, D., Culshaw, L., Kerr, K., Stoyanov, D., Luengo, I., 2023. A spatio-temporal network for video semantic segmentation in surgical videos. *Int. J. Comput. Assist. Radiol. Surg.*
- Hao, S., Zhou, Y., Guo, Y., 2020. A brief survey on semantic segmentation with deep learning. *Neurocomputing* 406, 302–321. <http://dx.doi.org/10.1016/j.neucom.2019.11.118>.
- Hara, K., Kataoka, H., Satoh, Y., 2018. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6546–6555. <http://dx.doi.org/10.1109/CVPR.2018.00685>.
- He, J., Li, P., Geng, Y., Xie, X., 2023. FastInst: A simple query-based model for real-time instance segmentation. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 23663–23672. <http://dx.doi.org/10.1109/CVPR52729.2023.02266>, URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.02266>.
- He, T., Shen, C., Tian, Z., Gong, D., Sun, C., Yan, Y., 2019. Knowledge adaptation for efficient semantic segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 578–587. <http://dx.doi.org/10.1109/CVPR.2019.00067>.
- He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: 2015 IEEE International Conference on Computer Vision. ICCV, pp. 1026–1034. <http://dx.doi.org/10.1109/ICCV.2015.123>.
- He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR pp. 770–778. <http://dx.doi.org/10.1109/CVPR.2016.90>.
- Hendrycks, D., Gimpel, K., 2016. Bridging nonlinearities and stochastic regularizers with Gaussian error linear units. *CoRR abs/1606.08415*.
- Hochbaum, D.S., 2001. An efficient algorithm for image segmentation, Markov random fields and related problems. *J. ACM* 48 (4), 686–701. <http://dx.doi.org/10.1145/502090.502093>.
- Holliday, A., Barekatin, M., Laurmaa, J., Kandaswamy, C., Prendinger, H., 2017. Speedup of deep learning ensembles for semantic segmentation using a model compression technique. *Comput. Vis. Image Underst.* 164, 16–26. <http://dx.doi.org/10.1016/j.cviu.2017.05.004>, Deep Learning for Computer Vision.
- Hong, Y., Pan, H., Sun, W., Jia, Y., 2021. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *CoRR abs/2101.06085*.
- Hou, R., Chen, C., Sukthankar, R., Shah, M., 2019a. An efficient 3D CNN for action/object segmentation in video. In: 30th British Machine Vision Conference 2019. BMVC 2019, Cardiff, UK, September 9–12, 2019, BMVA Press, p. 170.

- Hou, R., Chen, C., Sukthankar, R., Shah, M., 2019b. An efficient 3D CNN for action/object segmentation in video. In: 30th British Machine Vision Conference 2019. BMVC 2019, Cardiff, UK, September 9-12, 2019, BMVA Press, p. 170.
- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *CoRR* abs/1704.04861.
- Hu, P., Caba, F., Wang, O., Lin, Z., Sclaroff, S., Perazzi, F., 2020. Temporally distributed networks for fast video semantic segmentation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8815–8824. <http://dx.doi.org/10.1109/CVPR42600.2020.00884>.
- Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7132–7141. <http://dx.doi.org/10.1109/CVPR.2018.00745>.
- Huang, P.Y., Hsu, W.T., Chiu, C.Y., Wu, T.F., Sun, M., 2018a. Efficient uncertainty estimation for semantic segmentation in videos. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, pp. 536–552.
- Huang, G., Liu, Z., Maaten, L.V.D., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 2261–2269. <http://dx.doi.org/10.1109/CVPR.2017.243>.
- Huang, G., Liu, S., Maaten, L., Weinberger, K.Q., 2018b. CondenseNet: An efficient DenseNet using learned group convolutions. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 2752–2761. <http://dx.doi.org/10.1109/CVPR.2018.00291>.
- Huang, Z., Shi, X., Zhang, C., Wang, Q., Cheung, K.C., Qin, H., Dai, J., Li, H., 2022a. FlowFormer: A transformer architecture for optical flow. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (Eds.), *Computer Vision – ECCV 2022*. Springer Nature Switzerland, Cham, pp. 668–685.
- Huang, D.A., Yu, Z., Anandkumar, A., 2022b. MinVIS: A minimal video instance segmentation framework without video-based training. <http://dx.doi.org/10.48550/arXiv.2208.02245>.
- Hur, J., Roth, S., 2016. Joint optical flow and temporally consistent semantic segmentation. In: Hua, G., Jégou, H. (Eds.), *Computer Vision – ECCV 2016 Workshops*. Springer International Publishing, Cham, pp. 163–177.
- Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR* abs/1602.07360.
- Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T., 2017. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1647–1655. <http://dx.doi.org/10.1109/CVPR.2017.179>.
- Ioannou, Y., Robertson, D., Cipolla, R., Criminisi, A., 2017. Deep roots: Improving CNN efficiency with hierarchical filter groups. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 5977–5986. <http://dx.doi.org/10.1109/CVPR.2017.633>.
- Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K., 2015. Spatial transformer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. pp. 2017–2025.
- Jaderberg, M., Vedaldi, A., Zisserman, A., 2014. Speeding up convolutional neural networks with low rank expansions. In: BMVC 2014 - Proceedings of the British Machine Vision Conference 2014. <http://dx.doi.org/10.5244/C.28.88>.
- Jadon, S., 2020. A survey of loss functions for semantic segmentation. In: 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology. CIBCB, pp. 1–7. <http://dx.doi.org/10.1109/CIBCB48159.2020.9277638>.
- Jain, A., Chatterjee, S., Vidal, R., 2013. Coarse-to-fine semantic video segmentation using supervoxel trees. In: *Proceedings of the IEEE International Conference on Computer Vision. ICCV*.
- Jain, S., Wang, X., Gonzalez, J.E., 2018. Accel: A corrective fusion network for efficient semantic segmentation on video. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8858–8867.
- Jain, S., Xiong, B., Grauman, K., 2017. FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 2117–2126. <http://dx.doi.org/10.1109/CVPR.2017.228>.
- Ji, S., Xu, W., Yang, M., Yu, K., 2010. 3D convolutional neural networks for human action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 221–231.
- Jiang, S., Campbell, D., Lu, Y., Li, H., Hartley, R., 2021. Learning to estimate hidden motions with global motion aggregation. In: 2021 IEEE/CVF International Conference on Computer Vision. ICCV, pp. 9752–9761. <http://dx.doi.org/10.1109/ICCV48922.2021.00963>.
- Jin, X., Li, X., Xiao, H., Shen, X., Lin, Z., Yang, J., Chen, Y., Dong, J., Liu, L., Jie, Z., Feng, J., Yan, S., 2017. Video scene parsing with predictive feature learning. In: 2017 IEEE International Conference on Computer Vision. ICCV, pp. 5581–5589. <http://dx.doi.org/10.1109/ICCV.2017.595>.
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1725–1732. <http://dx.doi.org/10.1109/CVPR.2014.223>.
- Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. *Int. J. Comput. Vis.* 1 (4), 321–331. <http://dx.doi.org/10.1007/BF00133570>.
- Kim, D., Woo, S., Lee, J.Y., Kweon, I.S., 2020. Video panoptic segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR*.
- Koh, Y.J., Kim, C.S., 2017. Primary object segmentation in videos based on region augmentation and reduction. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 7417–7425. <http://dx.doi.org/10.1109/CVPR.2017.784>.
- Kohl, S.A.A., Romera-Paredes, B., Meyer, C., Fauw, J.D., Ledsam, J.R., Maier-Hein, K.H., Eslami, S.M.A., Rezende, D.J., Ronneberger, O., 2018. A probabilistic U-net for segmentation of ambiguous images. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS '18*, Curran Associates Inc., pp. 6965–6975.
- Kong, L., Shen, C., Yang, J., 2021. FastFlowNet: A lightweight network for fast optical flow estimation. In: 2021 IEEE International Conference on Robotics and Automation. ICRA, pp. 10310–10316. <http://dx.doi.org/10.1109/ICRA48506.2021.9560800>.
- Köpkülü, O., Hörmann, S., Herzog, F., Cevikalp, H., Rigoll, G., 2022. Dissected 3D CNNs: Temporal skip connections for efficient online video processing. *Comput. Vis. Image Underst.* 215, 103318. <http://dx.doi.org/10.1016/j.cviu.2021.103318>.
- Kopuklu, O., Kose, N., Gunduz, A., Rigoll, G., 2019. Resource efficient 3D convolutional neural networks. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop. ICCVW, IEEE Computer Society, Los Alamitos, CA, USA, pp. 1910–1919. <http://dx.doi.org/10.1109/ICCVW.2019.00240>.
- Krithika alias AnbuDevi, M., Suganthi, K., 2022. Review of semantic segmentation of medical images using modified architectures of UNET. *Diagnostics* 12 (12), <http://dx.doi.org/10.3390/diagnostics12123064>, URL: <https://www.mdpi.com/2075-4418/12/12/3064>.
- Krizhevsky, A., 2010. Convolutional deep belief networks on CIFAR-10. Not Published.
- Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), *In: Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc.
- Kundu, A., Vineet, V., Koltun, V., 2016. Feature space optimization for semantic video segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR*.
- Li, Y., Shi, J., Lin, D., 2018a. Low-latency video semantic segmentation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5997–6005. <http://dx.doi.org/10.1109/CVPR.2018.00628>.
- Li, M., Sun, L., Huo, Q., 2019. Flow-guided feature propagation with occlusion aware detail enhancement for hand segmentation in egocentric videos. *Comput. Vis. Image Underst.* 187, 102785. <http://dx.doi.org/10.1016/j.cviu.2019.07.005>.
- Li, J., Wang, W., Chen, J., Niu, L., Si, J., Qian, C., Zhang, L., 2021. Video semantic segmentation via sparse temporal transformer. In: *Proceedings of the 29th ACM International Conference on Multimedia. MM '21*, Association for Computing Machinery, pp. 59–68. <http://dx.doi.org/10.1145/3474085.3475409>.
- Li, G., Xie, Y., Wei, T., Wang, K., Lin, L., 2018b. Flow guided recurrent neural encoder for video salient object detection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3243–3252. <http://dx.doi.org/10.1109/CVPR.2018.00342>.
- Li, X., You, A., Zhu, Z., Zhao, H., Yang, M., Yang, K., Tan, S., Tong, Y., 2020. Semantic flow for fast and accurate scene parsing. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, pp. 775–793.
- Li, Y., Yuan, G., Wen, Y., Hu, J., Evangelidis, G., Tulyakov, S., Wang, Y., Ren, J., 2022a. EfficientFormer: Vision transformers at MobileNet speed. <http://dx.doi.org/10.48550/ARXIV.2206.01191>.
- Li, X., Yuan, H., Zhang, W., Cheng, G., Pang, J., Loy, C.C., 2023. Tube-link: A flexible cross tube baseline for universal video segmentation.
- Li, X., Zhang, W., Pang, J., Chen, K., Cheng, G., Tong, Y., Loy, C., 2022b. Video K-net: A simple, strong, and unified baseline for video segmentation. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 18825–18835. <http://dx.doi.org/10.1109/CVPR52688.2022.01828>, URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01828>.
- Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 936–944. <http://dx.doi.org/10.1109/CVPR.2017.106>.
- Lin, L., Yu, S., Zhou, L., Chen, W., Zhao, T., Wang, Z., 2020. PEA265: Perceptual assessment of video compression artifacts. *IEEE Trans. Circuits Syst. Video Technol.* 30 (11), 3898–3910. <http://dx.doi.org/10.1109/TCSVT.2020.2980571>.

- Liu, Y., Chen, K., Liu, C., Qin, Z., Luo, Z., Wang, J., 2019. Structured knowledge distillation for semantic segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 2599–2608. <http://dx.doi.org/10.1109/CVPR.2019.00271>.
- Liu, B., He, X., 2015. Multiclass semantic video segmentation with object-level active inference. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 4286–4294. <http://dx.doi.org/10.1109/CVPR.2015.7299057>.
- Liu, Z., Li, J., Ye, L., Sun, G., Shen, L., 2016. Saliency detection for unconstrained videos using superpixel-level graph and spatiotemporal propagation. IEEE Trans. Circuits Syst. Video Technol. PP, 1. <http://dx.doi.org/10.1109/TCSVT.2016.2595324>.
- Liu, Y., Shen, C., Yu, C., Wang, J., 2020. Efficient semantic video segmentation with per-frame inference. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), Computer Vision – ECCV 2020. Springer International Publishing, Cham, pp. 352–368.
- Liu, H., Simonyan, K., Yang, Y., 2018. DARTS: differentiable architecture search. CoRR abs/1806.09055.
- Liu, S., Wang, C., Qian, R., Yu, H., Bao, R., Sun, Y., 2017. Surveillance video parsing with single frame supervision. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA pp. 1013–1021. <http://dx.doi.org/10.1109/CVPR.2017.114>.
- Long, F., Qiu, Z., Pan, Y., Yao, T., Ngo, C.W., Mei, T., 2022. Dynamic temporal filtering in video models. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (Eds.), Computer Vision – ECCV 2022. Springer Nature Switzerland Cham, pp. 475–492.
- Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. pp. 3431–3440. <http://dx.doi.org/10.1109/CVPR.2015.7298965>.
- Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 60 (2), 91–110. <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>.
- Lu, X., Wang, W., Danelljan, M., Zhou, T., Shen, J., Van Gool, L., 2020. Video object segmentation with episodic graph memory networks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (Eds.), Computer Vision – ECCV 2020. Springer International Publishing, Cham, pp. 661–679.
- Ma, N., Zhang, X., Zheng, H.T., Sun, J., 2018. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), Computer Vision – ECCV 2018. Springer International Publishing, Cham, pp. 122–138.
- Mahadevan, S., Athar, A., Oşep, A., Hennen, S., Leal-Taixé, L., Leibe, B., 2020. Making a case for 3D convolutions for object segmentation in videos. In: BMVC.
- Mahasseni, B., Todorovic, S., Fern, A., 2017. Budget-aware deep semantic video segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR.
- Memmi, E., Perez, P., 1998. Dense estimation and object-based segmentation of the optical flow with robust techniques. IEEE Trans. Image Process. 7 (5), 703–719. <http://dx.doi.org/10.1109/83.668027>.
- Miao, J., Wei, Y., Wu, Y., Liang, C., Li, G., Yang, Y., 2021. VSPW: A large-scale dataset for video scene parsing in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR.
- Miksik, O., Munoz, D., Bagnell, J.A., Hebert, M., 2013. Efficient temporal consistency for streaming video scene analysis. In: 2013 IEEE International Conference on Robotics and Automation. pp. 133–139. <http://dx.doi.org/10.1109/ICRA.2013.6630567>.
- Miyama, M., 2021. FPGA implementation of 3-bit quantized CNN for semantic segmentation. J. Phys. Conf. Ser. 1729 (1), 012004. <http://dx.doi.org/10.1088/1742-6596/1729/1/012004>.
- Nathan Silberman, P.K., Fergus, R., 2012. Indoor segmentation and support inference from RGBD images. In: ECCV.
- Nekrasov, V., Chen, H., Shen, C., Reid, I., 2020. Architecture search of dynamic cells for semantic video segmentation. In: 2020 IEEE Winter Conference on Applications of Computer Vision. WACV, IEEE Computer Society, Los Alamitos, CA, USA, pp. 1959–1968. <http://dx.doi.org/10.1109/WACV45572.2020.9093531>.
- Neupane, B., Horanont, T., Aryal, J., 2021. Deep learning-based semantic segmentation of urban features in satellite images: A review and meta-analysis. Remote Sens. 13 (4), <http://dx.doi.org/10.3390/rs13040808>, URL: <https://www.mdpi.com/2072-4292/13/4/808>.
- Nilsson, D., Sminchisescu, C., 2018. Semantic video segmentation by gated recurrent flow propagation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 6819–6828. <http://dx.doi.org/10.1109/CVPR.2018.00713>.
- Nock, R., Nielsen, F., 2004. Statistical region merging. IEEE Trans. Pattern Anal. Mach. Intell. 26, 1452–1458. <http://dx.doi.org/10.1109/TPAMI.2004.110>.
- Ochs, P., Malik, J., Brox, T., 2014. Segmentation of moving objects by long term video analysis. IEEE Trans. Pattern Anal. Mach. Intell. 36 (6), 1187–1200. <http://dx.doi.org/10.1109/TPAMI.2013.242>.
- Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. 9 (1), 62–66. <http://dx.doi.org/10.1109/TSMC.1979.4310076>.
- Paupamah, K., James, S., Klein, R., 2020. Quantisation and pruning for neural network compression and regularisation. In: 2020 International SAUPEC/RobMech/PRASA Conference. pp. 1–6. <http://dx.doi.org/10.1109/SAUPEC/RobMech/PRASA48453.2020.9041096>.
- Peng, J., Liu, Y., Tang, S., Hao, Y., Chu, L., Chen, G., Wu, Z., Chen, Z., Yu, Z., Du, Y., Dang, Q., Lai, B., Liu, Q., Hu, X., Yu, D., Ma, Y., 2022. PP-LiteSeg: A superior real-time semantic segmentation model. <http://dx.doi.org/10.48550/ARXIV.2204.02681>.
- Phuong, M., Lampert, C., 2019. Towards understanding knowledge distillation. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 97, PMLR, pp. 5142–5151.
- Pinheiro, P.O., Lin, T.Y., Collobert, R., Dollár, P., 2016. Learning to refine object segments. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016. Springer International Publishing, Cham, pp. 75–91.
- Plath, N., Toussaint, M., Nakajima, S., 2009. Multi-class image segmentation using conditional random fields and global classification. In: Proceedings of the 26th Annual International Conference on Machine Learning. ICML '09, Association for Computing Machinery, New York, NY, USA, pp. 817–824. <http://dx.doi.org/10.1145/1553374.1553479>.
- Qin, Z., Lu, X., Nie, X., Liu, D., Yin, Y., Wang, W., 2023. Coarse-to-fine video instance segmentation with factorized conditional appearance flows. IEEE/CAA J. Autom. Sin. 10 (5), 1192–1208. <http://dx.doi.org/10.1109/JAS.2023.123456>.
- Qiu, Z., Yao, T., Mei, T., 2018. Learning deep spatio-temporal dependence for semantic video segmentation. IEEE Trans. Multimed. 20 (4), 939–949. <http://dx.doi.org/10.1109/TMM.2017.2759504>.
- Qu, L., Wu, C., Zou, L., 2020. 3D dense separated convolution module for volumetric medical image analysis. Appl. Sci. 10 (2), <http://dx.doi.org/10.3390/app10020485>.
- Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for activation functions. CoRR abs/1710.05941.
- Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A., 2016. XNOR-net: ImageNet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016. Springer International Publishing, Cham, pp. 525–542.
- Raza, S., Grundmann, M., Essa, I., 2013. Geometric context from videos. In: Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3081–3088. <http://dx.doi.org/10.1109/CVPR.2013.396>.
- Rhee, H., Min, D., Hwang, S., Andreis, B., Hwang, S.J., 2022. Distortion-aware network pruning and feature reuse for real-time video segmentation. <http://dx.doi.org/10.48550/ARXIV.2206.09604>.
- Richter, S.R., Vineet, V., Roth, S., Koltun, V., 2016. Playing for data: Ground truth from computer games. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016. Springer International Publishing, Cham, pp. 102–118.
- Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer International Publishing, Cham, pp. 234–241.
- Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M., 2016. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 3234–3243. <http://dx.doi.org/10.1109/CVPR.2016.352>.
- Saleh, F.S., Aliakbarian, M.S., Salzmann, M., Petersson, L., Alvarez, J.M., 2017. Bringing background into the foreground: Making all classes equal in weakly-supervised video semantic segmentation. In: 2017 IEEE International Conference on Computer Vision. ICCV, pp. 2125–2135. <http://dx.doi.org/10.1109/ICCV.2017.232>.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4510–4520. <http://dx.doi.org/10.1109/CVPR.2018.00474>.
- Schmidt, C., Athar, A., Mahadevan, S., Leibe, B., 2022. D2Conv3D: Dynamic dilated convolutions for object segmentation in videos. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision. WACV, IEEE Computer Society, Los Alamitos, CA, USA, pp. 1929–1938. <http://dx.doi.org/10.1109/WACV51458.2022.00199>.
- Schroff, F., Criminisi, A., Zisserman, A., 2006. Single-histogram class models for image segmentation. In: Kalra, P.K., Peleg, S. (Eds.), Computer Vision, Graphics and Image Processing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 82–93.
- Schroff, F., Criminisi, A., Zisserman, A., 2008. Object class segmentation using random forests. In: British Machine Vision Conference. URL: <https://api.semanticscholar.org/CorpusID:2136976>.
- Sevilla-Lara, L., Sun, D., Jampani, V., Black, M.J., 2016. Optical flow with semantic segmentation and localized layers. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 3889–3898. <http://dx.doi.org/10.1109/CVPR.2016.422>.
- Shelhamer, E., Rakelly, K., Hoffman, J., Darrell, T., 2016. Clockwork convnets for video semantic segmentation. In: Hua, G., Jégou, H. (Eds.), Computer Vision – ECCV 2016 Workshops. Springer International Publishing, Cham, pp. 852–868.

- Sherstinsky, A., 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D* 404, 132306. <http://dx.doi.org/10.1016/j.physd.2019.132306>.
- Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., Woo, W.c., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS '15, MIT Press, pp. 802–810.
- Shimoda, M., Sada, Y., Nakahara, H., 2019. Filter-wise pruning approach to FPGA implementation of fully convolutional network for semantic segmentation. In: Hochberger, C., Nelson, B., Koch, A., Woods, R., Diniz, P. (Eds.), *Applied Reconfigurable Computing*. Springer International Publishing, Cham, pp. 371–386.
- Shimoda, W., Yanai, K., 2020. Weakly supervised semantic segmentation using distinct class specific saliency maps. *Comput. Vis. Image Underst.* 191, 102712. <http://dx.doi.org/10.1016/j.cviu.2018.08.006>.
- Shotton, J., Winn, J., Rother, C., Criminisi, A., 2006. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (Eds.), *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–15.
- Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., Jagersand, M., Zhang, H., 2018. A comparative study of real-time semantic segmentation for autonomous driving. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Siam, M., Valipour, S., Jagersand, M., Ray, N., 2017. Convolutional gated recurrent networks for video segmentation. In: *2017 IEEE International Conference on Image Processing. ICIP*, pp. 3090–3094. <http://dx.doi.org/10.1109/ICIP.2017.8296851>.
- Sifre, L., Mallat, S., 2014. Rigid-motion scattering for texture classification. *CoRR abs/1403.1687*.
- Silberman, N., Fergus, R., 2011. Indoor scene segmentation using a structured light sensor. In: *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*.
- Song, H., Wang, W., Zhao, S., Shen, J., Lam, K.M., 2018. Pyramid dilated deeper convlstm for video salient object detection. In: *Proceedings of the European Conference on Computer Vision. ECCV*.
- Su, J., Yin, R., Zhang, S., Luo, J., 2023. Motion-state alignment for video semantic segmentation. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. CVPRW*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 3571–3580. <http://dx.doi.org/10.1109/CVPRW59228.2023.00365>, URL: <https://doi.ieeecomputersociety.org/10.1109/CVPRW59228.2023.00365>.
- Subramaniam, A., Vaidya, J., Ameen, M.A.M., Nambiar, A., Mittal, A., 2022. Co-segmentation inspired attention module for video-based computer vision tasks. *Comput. Vis. Image Underst.* 223, 103532. <http://dx.doi.org/10.1016/j.cviu.2022.103532>.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 1–9. <http://dx.doi.org/10.1109/CVPR.2015.7298594>.
- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V., 2019. MnasNet: Platform-aware neural architecture search for mobile. In: *IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2019*, Long Beach, CA, USA, June 16–20, 2019, Computer Vision Foundation / IEEE, pp. 2820–2828. <http://dx.doi.org/10.1109/CVPR.2019.00293>.
- Tao, A., Sapra, K., Catanzaro, B., 2020. Hierarchical multi-scale attention for semantic segmentation. *CoRR abs/2005.10821*.
- Thisanake, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R., Herath, D., 2023. Semantic segmentation using vision transformers: A survey. *Eng. Appl. Artif. Intell.* 126, 106669. <http://dx.doi.org/10.1016/j.engappai.2023.106669>, URL: <https://www.sciencedirect.com/science/article/pii/S0952197623008539>.
- Tokmakov, P., Alahari, K., Schmid, C., 2017. Learning video object segmentation with visual memory. In: *ICCV - IEEE International Conference on Computer Vision*. IEEE, Venice, Italy, pp. 4491–4500. <http://dx.doi.org/10.1109/ICCV.2017.480>.
- Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., 2015. Learning spatiotemporal features with 3D convolutional networks. In: *2015 IEEE International Conference on Computer Vision. ICCV*, pp. 4489–4497. <http://dx.doi.org/10.1109/ICCV.2015.510>.
- Tran, D., Wang, H., Feiszli, M., Torresani, L., 2019. Video classification with channel-separated convolutional networks. In: *2019 IEEE/CVF International Conference on Computer Vision. ICCV*, IEEE Computer Society, Los Alamitos, CA, USA, pp. 5551–5560. <http://dx.doi.org/10.1109/ICCV.2019.00565>.
- Tran, D., Wang, H., Torresani, L., Ray, J., Lecun, Y., Paluri, M., 2018. A closer look at spatiotemporal convolutions for action recognition. In: *Proceedings - 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR 2018*, In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 6450–6459. <http://dx.doi.org/10.1109/CVPR.2018.00675>.
- Tripathi, S., Belongie, S., Hwang, Y., Nguyen, T., 2015. Semantic video segmentation : Exploring inference efficiency. In: *ISOCC*.
- Tu, Z., Xie, W., Zhang, D., Poppe, R., Veltkamp, R.C., Li, B., Yuan, J., 2019. A survey of variational and CNN-based optical flow techniques. *Signal Process., Image Commun.* 72, 9–24. <http://dx.doi.org/10.1016/j.image.2018.12.002>.
- Unterwiesing, A., 2012. Compression artifacts in modern video coding and state-of-the-art means of compensation. *Multimed. Netw. Coding* 28–49. <http://dx.doi.org/10.4018/978-1-4666-2660-7.ch002>.
- Valada, A., Oliveira, G., Brox, T., Burgard, W., 2016. Deep multispectral semantic scene understanding of forested environments using multimodal fusion. In: *International Symposium on Experimental Robotics. ISER*.
- Varghese, S., Bayzidi, Y., Bar, A., Kapoor, N., Lahiri, S., Schneider, J.D., Schmidt, N.M., Schlicht, P., Huger, F., Fingscheidt, T., 2020. Unsupervised temporal consistency metric for video segmentation in highly-automated driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Varma, G., Subramanian, A., Nambodiri, A., Chandraker, M., Jawahar, C., 2019. IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments. In: *2019 IEEE Winter Conference on Applications of Computer Vision. WACV*, pp. 1743–1751. <http://dx.doi.org/10.1109/WACV.2019.00190>.
- Varol, G., Laptev, I., Schmid, C., 2018. Long-term temporal convolutions for action recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 40 (6), 1510–1517. <http://dx.doi.org/10.1109/TPAMI.2017.2712608>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), In: *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc.
- Vennerod, C.B., Kjær, A., Bugge, E.S., 2021. Long short-term memory RNN. *CoRR abs/2105.06756*.
- Vogel, S., Springer, J., Guntoro, A., Ascheid, G., 2019. Efficient acceleration of CNNs for semantic segmentation on FPGAs. In: *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. FPGA '19*, Association for Computing Machinery, p. 309. <http://dx.doi.org/10.1145/3289602.3294006>.
- Wang, W., Bao, H., Dong, L., Björck, J., Peng, Z., Liu, Q., Aggarwal, K., Mohammed, O.K., Singhal, S., Som, S., Wei, F., 2022a. Image as a foreign language: BEIT pretraining for all vision and vision-language tasks. <http://dx.doi.org/10.48550/ARXIV.2208.10442>.
- Wang, T., Cai, Y., Liang, L., Ye, D., 2020. A Multi-Level approach to waste object segmentation. *Sensors (Basel)* 20 (14).
- Wang, W., Dai, J., Chen, Z., Huang, Z., Li, Z., Zhu, X., Hu, X., Lu, T., Lu, L., Li, H., Wang, X., Qiao, Y., 2022b. InternImage: Exploring large-scale vision foundation models with deformable convolutions. <http://dx.doi.org/10.48550/ARXIV.2211.05778>.
- Wang, L., Ju, L., Zhang, D., Wang, X., He, W., Huang, Y., Yang, Z., Yao, X., Zhao, X., Ye, X., Ge, Z., 2021a. Medical matting: A new perspective on medical segmentation with uncertainty. In: de Bruijne, M., Cattin, P.C., Cotin, S., Paday, N., Speidel, S., Zheng, Y., Essert, C. (Eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*. Springer International Publishing, Cham, pp. 573–583.
- Wang, W., Lu, X., Shen, J., Crandall, D.J., Shao, L., 2019a. Zero-shot video object segmentation via attentive graph neural networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV*.
- Wang, Q., Piao, Y., 2023. Depth estimation of supervised monocular images based on semantic segmentation. *J. Vis. Commun. Image Represent.* 90, 103753. <http://dx.doi.org/10.1016/j.jvcir.2023.103753>, URL: <https://www.sciencedirect.com/science/article/pii/S1047320323000032>.
- Wang, W., Shen, J., Porikli, F., 2015a. Saliency-aware geodesic video object segmentation. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition. CVPR*, pp. 3395–3402. <http://dx.doi.org/10.1109/CVPR.2015.7298961>.
- Wang, W., Shen, J., Porikli, F., Yang, R., 2019b. Semi-supervised video object segmentation with super-trajectories. *IEEE Trans. Pattern Anal. Mach. Intell.* 41 (04), 985–998. <http://dx.doi.org/10.1109/TPAMI.2018.2819173>.
- Wang, W., Shen, J., Shao, L., 2015b. Consistent video saliency using local gradient flow optimization and global refinement. *IEEE Trans. Image Process.* 24 (11), 4185–4196. <http://dx.doi.org/10.1109/TIP.2015.2460013>.
- Wang, H., Wang, W., Liu, J., 2021b. Temporal memory attention for video semantic segmentation. In: *2021 IEEE International Conference on Image Processing. ICIP*, pp. 2254–2258. <http://dx.doi.org/10.1109/ICIP42928.2021.9506731>.
- Weber, M., Xie, J., Collins, M., Zhu, Y., Voigtlaender, P., Adam, H., Green, B., Geiger, A., Leibe, B., Cremers, D., Osep, A., Leal-Taixe, L., Chen, L.C., 2021. STEP: Segmenting and tracking every pixel. In: *Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*.
- Wiskott, L., Sejnowski, T.J., 2002. Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.* 14, 715–770.
- Wu, J., Liu, Q., Jiang, Y., Bai, S., Yuille, A.L., Bai, X., 2022. In defense of online models for video instance segmentation. In: *European Conference on Computer Vision*. URL: <https://api.semanticscholar.org/CorpusID:250918749>.

- Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K., 2018a. Shift: A zero FLOP, zero parameter alternative to spatial convolutions. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 9127–9135. <http://dx.doi.org/10.1109/CVPR.2018.00951>.
- Wu, H., Zheng, S., Zhang, J., Huang, K., 2018b. Fast end-to-end trainable guided filter. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1838–1847. <http://dx.doi.org/10.1109/CVPR.2018.00197>.
- Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR.
- Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K., 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, pp. 318–335.
- Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P., 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., pp. 12077–12090.
- Xu, C., Corso, J.J., 2012. Evaluation of super-voxel methods for early video processing. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1202–1209. <http://dx.doi.org/10.1109/CVPR.2012.6247802>.
- Xu, M., Ding, Y., 2021. Fully automatic image colorization based on semantic segmentation technology. *PLoS One* 16 (11), e0259953.
- Xu, Y., Fu, T., Yang, H., Lee, C., 2018. Dynamic video segmentation network. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 6556–6565. <http://dx.doi.org/10.1109/CVPR.2018.00686>.
- Xu, J., Xiong, Z., Bhattacharyya, S.P., 2022. PIDNet: A real-time semantic segmentation network inspired from PID controller. <http://dx.doi.org/10.48550/ARXIV.2206.02066>.
- Yang, T.J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., Sze, V., Adam, H., 2018. NetAdapt: Platform-aware neural network adaptation for mobile applications. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), *Computer Vision – ECCV 2018*. Springer International Publishing, Cham, pp. 289–304.
- Yang, Z., Wang, Q., Bertinetto, L., Bai, S., Hu, W., Torr, P., 2019. Anchor diffusion for unsupervised video object segmentation. In: 2019 IEEE/CVF International Conference on Computer Vision. ICCV, IEEE Computer Society, Los Alamitos, CA, USA, pp. 931–940. <http://dx.doi.org/10.1109/ICCV.2019.00102>.
- Yang, C., Zhou, H., An, Z., Jiang, X., Xu, Y., Zhang, Q., 2022. Cross-image relational knowledge distillation for semantic segmentation. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 12309–12318. <http://dx.doi.org/10.1109/CVPR52688.2022.01200>.
- Young, S.I., Girod, B., Taubman, D., 2020. Fast optical flow extraction from compressed video. *IEEE Trans. Image Process.* 29, 6409–6421. <http://dx.doi.org/10.1109/TIP.2020.2985866>.
- Yu, Z., Wong, H.S., Wen, G., 2011. A modified support vector machine and its application to image segmentation. *Image Vis. Comput.* 29 (1), 29–40. <http://dx.doi.org/10.1016/j.imavis.2010.08.003>, URL: <https://www.sciencedirect.com/science/article/pii/S0262885610001113>.
- Yu, Y., Yuan, J., Mittal, G., Fuxin, L., Chen, M., 2022. BATMAN: Bilateral attention transformer in motion-appearance neighboring space for video object segmentation. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (Eds.), *Computer Vision – ECCV 2022*. Springer Nature Switzerland, Cham, pp. 612–629.
- Zhai, M., Xiang, X., Lv, N., Kong, X., 2021. Optical flow and scene flow estimation: A survey. *Pattern Recognit.* 114, 107861. <http://dx.doi.org/10.1016/j.patcog.2021.107861>.
- Zhang, Y., Borse, S., Cai, H., Wang, Y., Bi, N., Jiang, X., Porikli, F., 2022. Perceptual consistency in video segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. WACV, pp. 2564–2573.
- Zhang, W., Pang, J., Chen, K., Loy, C.C., 2021. K-Net: Towards unified image segmentation. *arXiv abs/2106.14855*.
- Zhang, T., Tian, X., Wu, Y., Ji, S., Wang, X., Zhang, Y., Wan, P., 2023. DVIS: Decoupled video instance segmentation framework. *arXiv abs/2306.03413*.
- Zhang, H., Wang, J., Sun, Z., Zurada, J.M., Pal, N.R., 2020. Feature selection for neural networks using group lasso regularization. *IEEE Trans. Knowl. Data Eng.* 32 (4), 659–673. <http://dx.doi.org/10.1109/TKDE.2019.2893266>.
- Zhang, X., Zhou, X., Lin, M., Sun, J., 2017. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6848–6856.
- Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid scene parsing network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 6230–6239. <http://dx.doi.org/10.1109/CVPR.2017.660>.
- Zhao, Z., Zhao, S., Shen, J., 2021. Real-time and light-weighted unsupervised video object segmentation network. *Pattern Recognit.* 120, 108120. <http://dx.doi.org/10.1016/j.patcog.2021.108120>.
- Zhen, M., Li, S., Zhou, L., Shang, J., Feng, H., Fang, T., Quan, L., 2020. Learning discriminative feature with CRF for unsupervised video object segmentation. In: *European Conference on Computer Vision*.
- Zhou, T., Porikli, F., Crandall, D.J., Van Gool, L., Wang, W., 2023. A survey on deep learning technique for video segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 45 (6), 7099–7122. <http://dx.doi.org/10.1109/TPAMI.2022.3225573>.
- Zhu, Y., Sapra, K., Reda, F.A., Shih, K.J., Newsam, S., Tao, A., Catanzaro, B., 2019. Improving semantic segmentation via video propagation and label relaxation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR, pp. 8848–8857. <http://dx.doi.org/10.1109/CVPR.2019.00906>.
- Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J., 2021. Deformable DETR: deformable transformers for end-to-end object detection. In: 9th International Conference on Learning Representations. ICLR 2021, Virtual Event, Austria, May 3–7, 2021, OpenReview.net, URL: <https://openreview.net/forum?id=gZ9hCDW6ke>.
- Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y., 2017. Deep feature flow for video recognition. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, IEEE Computer Society, Los Alamitos, CA, USA, pp. 4141–4150. <http://dx.doi.org/10.1109/CVPR.2017.441>.