Learning from motion

Pavel Tokmakov

Communauté UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Pavel TOKMAKOV

Thèse dirigée par **Cordelia SCHMID**, directeur de recherche , INRIA

préparée au sein du **Laboratoire Jean Kuntzmann**

dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Apprentissage à partir du mouvement

Learning from Motion

Thèse soutenue publiquement le **4 juin 2018**,
devant le jury composé de :

Madame CORDELIA SCHMID
DIRECTRICE DE RECHERCHE, INRIA CENTRE DE GRENOBLE RHÔNE-ALPES, Directeur de thèse
Monsieur JEAN PONCE
PROFESSEUR, ECOLE NORMALE SUPERIEURE DE PARIS, Président
Monsieur THOMAS BROX
PROFESSEUR, UNIVERSITE DE FRIBOURG - ALLEMAGNE, Rapporteur
Monsieur IASONAS KOKKINOS
PROFESSEUR, UNIVERSITE COLLEGE DE LONDRES, Rapporteur
Monsieur KARTEEK ALAHARI
CHARGE DE RECHERCHE, INRIA CENTRE DE GRENOBLE RHÔNE-ALPES, Examinateur

Abstract

Weakly-supervised learning studies the problem of minimizing the amount of human effort required for training state-of-the-art models. This allows to leverage a large amount of data. However, in practice weakly-supervised methods perform significantly worse than their fully-supervised counterparts. This is also the case in deep learning, where the top-performing computer vision approaches remain fully-supervised, which limits their usage in real world applications. This thesis attempts to bridge the gap between weakly-supervised and fully-supervised methods by utilizing motion information. It also studies the problem of moving object segmentation itself, proposing one of the first learning-based methods for this task.

We focus on the problem of weakly-supervised semantic segmentation. This is especially challenging due to the need to precisely capture object boundaries and avoid local optima, as for example segmenting the most discriminative parts. In contrast to most of the state-of-the-art approaches, which rely on static images, we leverage video data with object motion as a strong cue. In particular, our method uses a state-of-the-art video segmentation approach to segment moving objects in videos. The approximate object masks produced by this method are then fused with the semantic segmentation model learned in an EM-like framework to infer pixel-level semantic labels for video frames. Thus, as learning progresses, the quality of the labels improves automatically. We then integrate this architecture with our learning-based approach for video segmentation to obtain a fully trainable framework for weakly-supervised learning from videos.

In the second part of the thesis we study unsupervised video segmentation, the task of segmenting all the objects in a video that move independently from the camera. This task presents challenges such as strong camera motion, inaccuracies in optical flow estimation and motion discontinuity. We address the camera motion problem by proposing a learning-based method for motion segmentation: a convolutional neural network that takes optical flow as input and is trained to segment objects that move independently from the camera. It is then extended with an appearance stream and a visual memory module to improve temporal continuity. The appearance stream capitalizes on the semantic information which is complementary to the motion information. The visual memory module is the key component of our approach: it combines the outputs of the motion and appearance streams and aggregates a spatio-temporal representation of the moving objects. The final segmentation is then produced based on this aggregated representation. The resulting approach obtains state-of-the-art performance on several benchmark datasets, outperforming the concurrent deep learning and heuristic-based methods.

Keywords : weakly-supervised learning, semantic segmentation, motion segmentation, video object segmentation, computer vision, machine learning

Résumé

L'apprentissage faiblement supervisé cherche à réduire au minimum l'effort humain requis pour entrainer les modèles de l'état de l'art. Cette technique permet de tirer parti d'une énorme quantité de données. Toutefois, dans la pratique, les méthodes faiblement supervisées sont nettement moins efficaces que celles qui sont totalement supervisées. Plus particulièrement, dans l'apprentissage profond, oú les approches de vision par ordinateur sont les plus performantes, elles restent entièrement supervisées, ce qui limite leurs utilisations dans les applications du monde réel. Cette thèse tente tout d'abord de combler le fossé entre les méthodes faiblement supervisées et entièrement supervisées en utilisant l'information de mouvement. Puis étudie le problème de la segmentation des objets en mouvement, en proposant l'une des premières méthodes basées sur l'apprentissage pour cette tâche.

Dans une première partie de la thèse, nous nous concentrons sur le problème de la segmentation sémantique faiblement supervisée. Le d'efi est de capturer de manières précises les bordures des objets et d'éviter les optimums locaux (ex : segmenter les parties les plus discriminantes). Contrairement à la plupart des approches de l'état de l'art, qui reposent sur des images statiques, nous utilisons les données vidéo avec le mouvement de l'objet comme informations importantes. Notre méthode utilise une approche de segmentation vidéo de l'état de l'art pour segmenter les objets en mouvement dans les vidéos. Les masques d'objets approximatifs produits par cette méthode sont ensuite fusionnés avec le modèle de segmentation sémantique appris dans un EM-like framework, afin d'inférer pour les trames vidéo, des labels sémantiques au niveau des pixels. Ainsi, au fur et à mesure que l'apprentissage progresse, la qualité des labels s'améliore automatiquement. Nous intégrons ensuite cette architecture á notre approche basée sur l'apprentissage pour la segmentation de la vidéo afin d'obtenir un framework d'apprentissage complet pour l'apprentissage faiblement supervisé à partir de vidéos.

Dans la deuxième partie de la thèse, nous étudions la segmentation vid'eo non supervisée, plus précisément comment segmenter tous les objets dans une vidéo qui se déplace indépendamment de la caméra. De nombreux défis tels qu'un grand mouvement de la caméra, des inexactitudes dans l'estimation du flux optique et la discontinuité du mouvement, complexifient la tâche de segmentation. Nous abordons le problème du mouvement de caméra en proposant une méthode basée sur l'apprentissage pour la segmentation du mouvement : un réseau de neurones convolutif qui prend le flux optique comme entrée et qui est entraîné pour segmenter les objets qui se déplacent indépendamment de la caméra. Il est ensuite étendu avec un flux d'apparence et un module de mémoire visuelle pour améliorer la continuité temporelle. Le flux d'apparence tire profit de l'information sémantique qui est complémentaire de l'information de mouvement. Le module de mémoire visuelle est un

paramètre clè de notre approche : il combine les sorties des flux de mouvement et d'apparence et agréger une représentation spatio-temporelle des objets en mouvement. La segmentation finale est ensuite produite à partir de cette représentation agrégée. L'approche résultante obtient des performances de l'état de l'art sur plusieurs jeux de données de référence, surpassant la méthode d'apprentissage en profondeur et heuristique simultanée.

Mots-clefs : Apprentissage de manière faiblement supervisée, segmentation sémantique, segmentation de mouvement, segmentation d'objets vidéo, vision par ordinateur, apprentissage automatique

Acknowledgements

The author would like to express his gratitude to the people who made this work possible: to Dr. Cordelia Schmid for teaching him how to conduct research and for imparting the highest quality standards; to Dr. Karteek Alahari for his advice on how to present the research results and for his patience; to Nathalie Gillot for her tireless assistance in fighting the French bureaucracy; to Xavier Martin and Ghislain Durif for showing him how to turn it off and on again; to his family for their constant support and encouragement; to Dima, Nikita, Lina, Tom, Valentin and Dan for being good friends; and to Bauhaus bar for providing a recreational environment that helped to keep his mental health though all this.

Contents

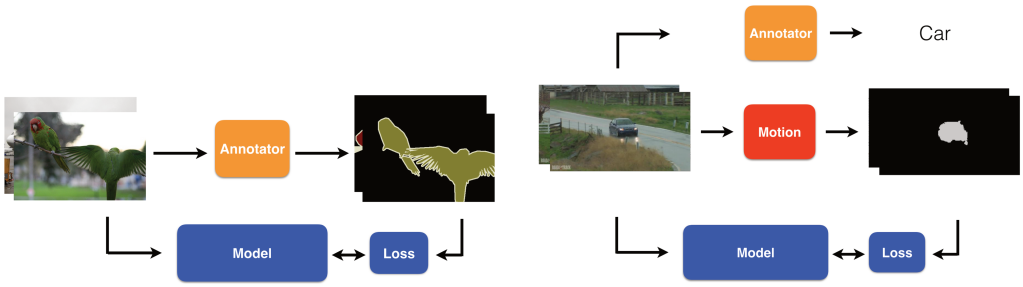
Chapter 1

Introduction

Contents

Machine learning and, in particular, deep learning-based approaches have taken over nearly all areas of computer vision and AI recently. Indeed, learning task-specific representations from data has achieved better than human performance on image classification with CNNs [54], produced almost human-quality results in some machine translation settings with LSTMs [151] and beaten the best Go players in the world using deep reinforcement learning [127], all of which seemed unthinkable even five years ago. In addition, deep networks have been shown to better model neural responses in higher visual cortical areas of the human brain than earlier, hand-designed approaches [50, 155]. A natural question arises: is this all we need? Is deep learning going to solve computer vision and AI, given enough data? Many researchers disagree with this statement. To see why, consider a typical computer vision task.

In particular, let us say we want to classify image pixels into different visual categories, a task commonly known as semantic segmentation, using deep learning. Semantic segmentation combines the problems of object classification and precise localization, which makes it critical for many applications, including robotics, where knowing precise object boundaries is necessary for grasping [84], or biomedical imaging [122], where accurately



Fully-supervised setting Weakly-supervised setting

Figure 1.1 – Overview of fully-supervised semantic segmentation and our weakly-supervised approach, that uses motion to infer pixel-level labels.

segmenting different tissues helps in diagnosing diseases. The first step towards learning a semantic segmentation model is selecting one of the top-performing architectures in the literature [24, 92, 159]. The set of categories that need to be processed is defined then, hundreds to thousands of images for each category are collected, and each image is annotated with pixel-level labels (see Figure 1.1, left). Given that annotating each object in an image takes about 79 seconds, as reported in [93], users that actively exploit machine learning in their business tend to hire specialized data annotation companies to label their data. After the annotations have been obtained, a model can be trained and used for the task of semantic segmentation. If the set of categories the model segments has to be extended later, several images for the new categories have to be collected and annotated again. The model then has to be retrained on this extended set. Such an approach becomes extremely expensive as the set of categories grows and is completely infeasible for the ultimate task of capturing the richness of the entire visual world. A different learning framework is needed, which is less dependent on expensive manual annotations.

The problem of reducing the amount of supervision in machine learning has received significant attention, resulting in the fields of weakly- [33, 99] and semi-supervised [66, 117] learning. Semi-supervised learning exploits a small set of labeled examples to extract useful information from a large pool of unlabeled ones, whereas weakly-supervised learning explores the setting where each example is provided with a limited form of annotation (e.g,, boxes instead of pixel-level labels for semantic segmentation). In this work we focus on the weakly-supervised setting and, in particular, study weakly-supervised semantic segmentation, which has been a core problem in computer vision recently. For instance, Khoreva et al. [69] have developed a method for learning state-of-the-art semantic segmentation models from

bounding box annotations, but these annotations remain relatively costly to obtain. Methods that use image level tags are more promising, but, despite the recent progress [75, 107], their performance remains considerably lower than their fully-supervised counterparts, with the top approaches relying on additional manual annotations [106, 109]. This is due to the fact that the problem of weakly-supervised semantic segmentation from image tags is under-constrained, and additional cues are needed to obtain precise segmentations. These cues can either be provided manually, or obtained automatically. In this dissertation we focus on the latter setting, which is both more rewarding and more challenging. To find a natural source for such additional cues we now turn to the literature on vision in humans.

In the human visual system, learning happens with a minimal amount of supervision [130]. Being born with only the most primitive forms of object perception [14, 36], infants acquire the notion of objectness and of semantic categories through interaction with the environment during their initial stage of life. In [130], Elizabeth Spelke has studied cues used by four-month old infants to group surfaces into objects. This research has shown that motion coherence is one of the primary signals that guide the learning of visual concepts in the early stages of development. In particular, infants observe the world as a combination of surfaces of different colors and group those that move together into objects, which first leads to learning the notion of shapes and later of semantic categories. This is in strong contrast to the majority of weakly-supervised semantic segmentation methods, which ignore motion information, relying exclusively on static images.

We attempt to close this gap by introducing the first video-based weakly-supervised semantic segmentation method (Figure 1.1, right) in Chapter 3. It integrates motion cues, extracted with a heuristic-based approach, with a semantic model learned in an expectation maximization-like framework, which allows to obtain high-quality labels automatically for training the model. Motion plays the role of a key learning bias in this approach, thus the quality of motion estimation is of a special importance. This motivates us to study the moving object segmentation problem in isolation and develop novel, learning-based approaches for motion and video object segmentation in Chapter 5. Finally, in Chapter 6, we integrate our video segmentation method in the weakly-supervised semantic segmentation framework to obtain a fully trainable approach for learning from videos, and demonstrate its superiority over the original method based on heuristic motion segmentation.

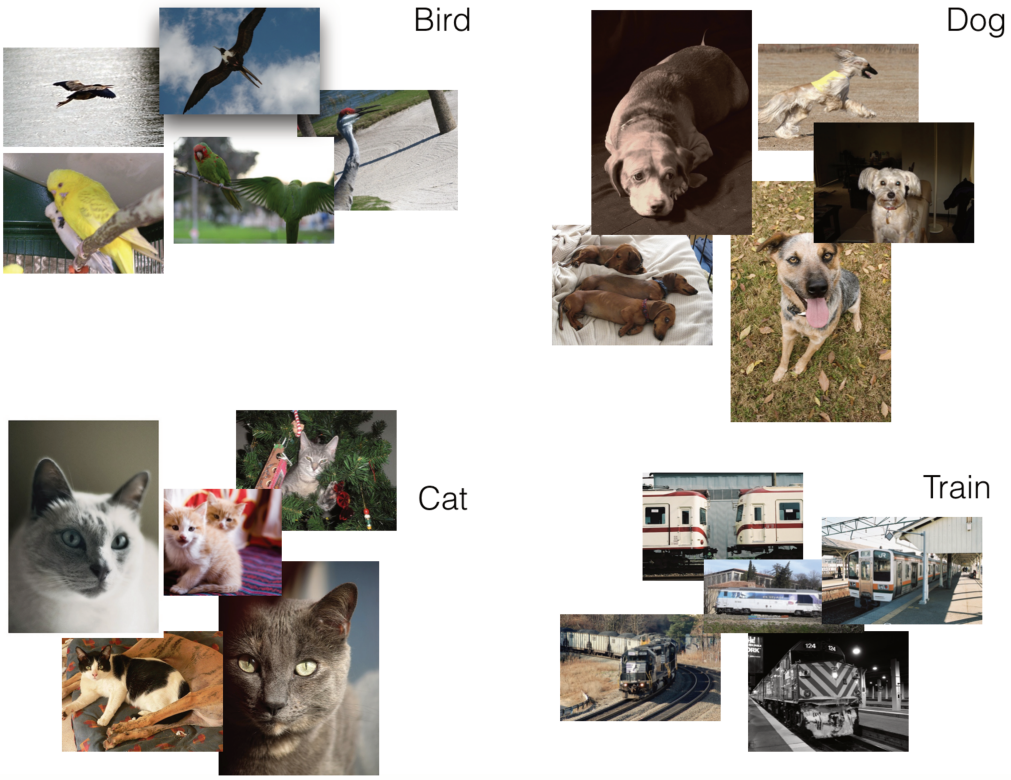


Figure 1.2 – Illustration of a weakly-supervised semantic segmentation setting, where only image tags are available as supervision. Some of the main challenges specific to this setting are: appearance variability in the train example, dominance of discriminative object parts in the cat and dog examples, and correlation of object and background in the train example.

1.1 Goals and challenges

This dissertation addresses two related but independent topics: weakly-supervised learning and video object segmentation. The first one is concerned with reducing the annotation effort in learning a semantic segmentation model. The second studies the problem of segmenting moving objects in videos. We now briefly present these two tasks, as well as the challenges involved in each of them.

1.1.1 Weakly-supervised learning

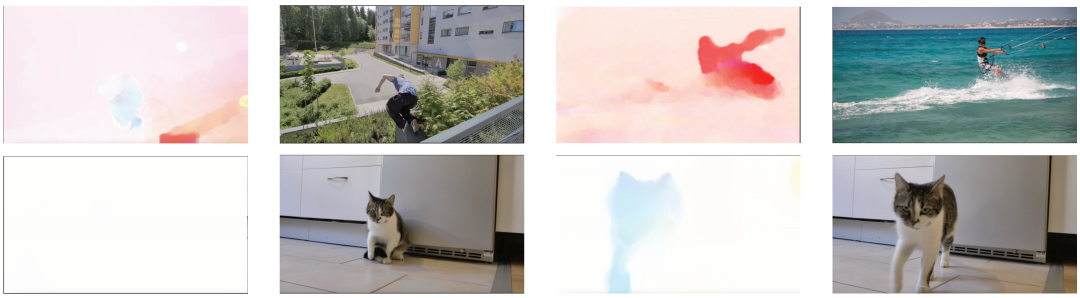
The goal of weakly-supervised learning is to reduce the amount of supervision required to train state-of-the-art models. The particular setting

depends on the task at hand and the desired granularity of annotations. For instance, for semantic segmentation, which requires pixel-level labels in a fully-supervised regime, the annotation effort can be reduced by shifting to bounding-boxes [69, 152, 163], image tags [75, 107, 114], or even to no manual annotations at all, capitalizing on web search engines to query images by keywords [65]. This dissertation is concerned with the tag setting, where only the presence of categories in an image or a video is indicated, but not the location or the number of corresponding objects (see Figure 1.2). Thus, the only information available to weakly-supervised methods in this setting is co-occurrence statistics (i.e., what is common among these images containing cats and how they differ from those which contain dogs, trains, and birds), and low-level cues, such as colour constancy in images. This under-constrained problem poses many challenges, including object size and appearance variation, convergence to the most discriminative parts, object-background correlations, and capturing the exact object boundaries. We now elaborate on the aforementioned challenges.

The **variability of the visual world** itself is one of the main difficulties for the tag-only setting. Objects come in a variety of shapes, poses and sizes and state-of-the-art approaches based on co-occurrence tend to ignore the ones that are most different from the modes of the distribution. For example, at test time the model might perform well on large objects, captured from canonical angles, like the flying birds in the top left corner of Figure 1.2, but will fail to segment the head of a crane.

A major problem is **convergence to the most discriminative parts**, which is is essentially a more severe version of the problem described above. Weakly-supervised approaches, faced with large appearance variation, learn to ignore object parts that show a large variability and segment only the parts that have a relatively consistent appearance in most of the images. Examples include snouts of animals or radiators of cars. Avoiding this issue requires imposing object-specific constraints on the extent of the segmentation, which can not be achieved with low-level image cues only.

Another challenge common to weakly-supervised semantic segmentation is **object-background correlation**: capturing correspondence between the object and the background during training. For example, since birds tend to appear with sky as a background (see examples in the top left corner of Figure 1.2), the model learns to segment sky as a bird as well. The same type of object-background correlation occurs for most of the categories (for instance, trains usually appear on railroad tracks and cars on roads), making this a significant issue. Notice that this problem can be seen as the opposite of convergence to parts and different design choices usually lead to the model being susceptible to either one or the other.



LDOF flow [19] Frame LDOF flow [19] Frame

Figure 1.3 – Illustration of common video segmentation challenges with videos from DAVIS [112] and FBMS [105] datasets. The 'parkour' and 'kite-surf' examples in the top row illustrate camera motion and 'stuff' motion, respectively. The 'cat' example in the bottom row shows a case of motion discontinuity in videos.

An issue that most of the weakly-supervised semantic segmentation approaches are facing is that of **capturing the exact object boundaries**. As mentioned above, some of the challenges of weakly-supervised semantic segmentation are conflicting with each other. Avoiding all of them at the same time without additional information about the extent of the object requires introducing generic constraints into the learning framework. This results in segmentation models that can approximately capture the location and extent of the object, but not its precise boundaries.

1.1.2 Video segmentation

The second problem studied in this dissertation is video object segmentation: segmenting objects which exhibit independent motion in at least one frame in the video [35, 108]. Apart from being useful for weakly-supervised learning, as we will show later, this problem is of importance for video editing [145] and is related to semantic video segmentation [41, 141]. A key component for video object segmentation is motion segmentation - estimation of independent object motion between pairs of frames [101, 139]. It is often used as a local, frame-level signal to bootstrap video segmentation pipelines. The main challenges for both these problems, are strong camera motion, 'stuff' motion, flow estimation inaccuracies and motion discontinuity (see Figure 1.3 for examples). We now define each of these challenges in more detail.

In the absence of **camera motion**, independently moving objects can be segmented at the frame level by simply thresholding optical flow mag-

nitude. If the camera is not static, however, the problem becomes much more challenging, as shown in the 'parkour' example in the top left corner of Figure 1.3. Strong camera motion in combination with complex background can produce patterns in the optical flow field that are non-trivial to discriminate from those formed by independent object motion (see the strong optical flow variation on the ramp in the bottom right corner of the 'parkour' example).

Another issue related to background motion is that of **'stuff' motion** (water, smoke, etc.). These materials can indeed move independently from the camera (see the example of moving water in the top right corner of Figure 1.3). They, however, are not the focus of video object segmentation approaches, as they do not constitute objects. Methods relying exclusively on optical flow, such as [101, 139], can not resolve this ambiguity, since the object and stuff motion patterns are similar. Thus, it is necessary to leverage appearance cues in addition to the motion cues for video object segmentation.

Optical **flow estimation inaccuracies** are a significant issue for any motion-related task. Consider, for instance, the flow for the parkour video in the top left corner of Figure 1.3. It is estimated with the classical LDOF algorithm [19], which produces artifacts on the ramp and on the bushes. This can in turn lead to a motion segmentation method incorrectly segmenting these regions as moving. Object appearance and temporal consistency can serve as additional sources of information to avoid this type of mistakes.

Another issue is that in general objects in videos do not move consistently throughout the whole sequence. They exhibit **motion discontinuity**, stopping for a part of the sequence and then resuming with a potentially different direction and velocity. Moreover, objects can be static in the beginning of the video and only manifest independent motion in the latter parts, as shown in the bottom row in Figure 1.3, where the cat is static at first, but starts to walk towards the camera later. Thus, segmenting objects in all the video frames requires remembering the previous frames to segment the current one, as well as processing the video as a whole, and not only in the forward direction.

1.2 Contributions

1.2.1 Weakly-supervised semantic segmentation

Our contributions in weakly-supervised learning are centered around demonstrating the importance of motion as a cue for learning with tag-only

annotations. Traditional approaches for weakly-supervised semantic segmentation relied exclusively on static images, and hence resorted to heuristic learning biases. For instance, the method of Duyuglu et al. [33] operated on the level of unsupervised segmentation proposals, thus incorporating a strong object prior into the learned model. It's performance was, however, limited by the quality of the generated proposals. Some of the more recent, FCNN-based approaches [110, 114] formulated weakly-supervised segmentation as a multiple instance learning problem. They assumed that there is only one instance of the object of interest in every image and it has to be correctly selected from a pool of irrelevant instances. In addition to the single-object-per-image assumption being too restrictive, these methods are especially susceptible to the convergence to parts problem described in Section 1.1.1.

Other methods [107, 109] have attempted to introduce generic constraints into the weakly-supervised learning framework. In particular, they proposed to manually constraint the area an object can occupy in an image while inferring pixel labels. These constraints allowed them to partially mitigate the convergence to parts and object-background correlation issues. They, however, were not object-specific and thus, did not necessarily hold for all the images. This resulted in models that could roughly localize the objects, but failed to capture their exact boundaries.

Some approaches proposed ways to obtain object-specific constraints for static images. For instance, Kolesnikov et al. [75] utilized the method of Zhou et al. [162] to extract class activation maps (CAMs) from a model trained for image classification. These activation maps are not accurate object segmentations per se, but can be used to estimate the object location in an image. They then capitalized on low-level image cues, such as colour constancy, to infer the boundaries of an object. In another work Oh et al. [106] augment CAM localization cues with a stronger shape prior - objectness. They employ a dataset with bounding-box level object annotation to learn a generic object segmentation model. Foreground/background segmentations produced by this model are then combined with class-specific localization cues of Zhou et al. [162] to infer pixel-level labels for training a semantic segmentation model. This approach achieves outstanding results, but requires bounding box annotations for learning objectness.

Our work is different in that we propose to utilize motion as an object-specific segmentation cue that can be obtained automatically. Our first contribution is an expectation-maximization-like framework for weakly-supervised learning from videos that encodes motion information as a prior to significantly improve the accuracy of the label inference. The second contribution resides in integrating a learning-based video segmentation approach into the

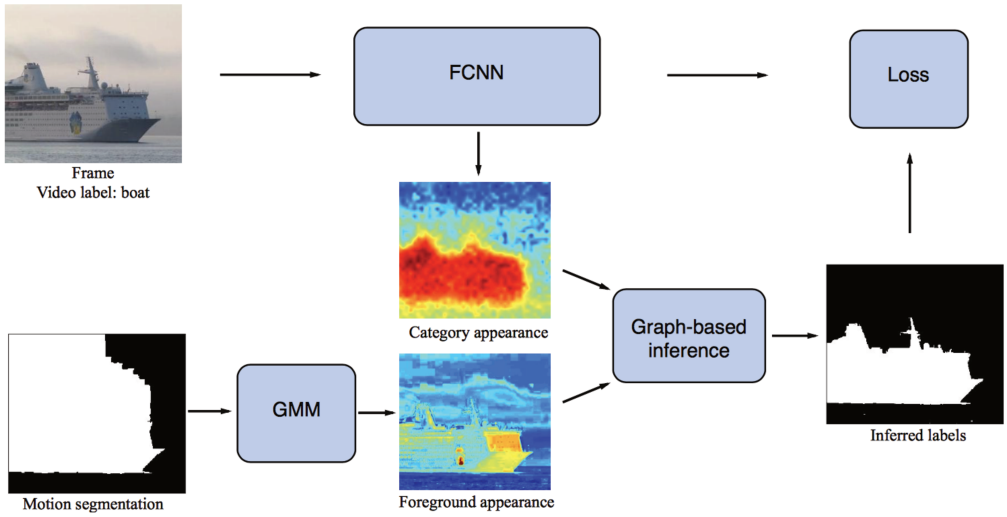


Figure 1.4 – Overview of our weakly-supervised semantic segmentation framework. The soft potentials (foreground appearance) computed from motion segmentation and the semantic predictions computed by FCNN jointly determine the latent segmentation (inferred labels) used to learn the model.

EM-framework, to demonstrate the benefits of a fully-trainable model.

Weakly-Supervised Semantic Segmentation using Motion Cues. In our ECCV'16 publication we propose motion-CNN (M-CNN) [136], a novel fully-convolutional neural network (FCNN) framework which incorporates motion cues, and is learned from tag-level video annotations. Our learning scheme to train the network, shown in Figure 1.4, uses motion segments as soft constraints, thereby handling noisy motion information. When trained on weakly annotated videos, our method outperforms the state-of-the-art approach on the PASCAL VOC 2012 image segmentation benchmark. We also demonstrate that the performance of M-CNN learned with 150 weak video annotations is on par with state-of-the-art weakly-supervised methods trained with thousands of images [107, 109, 110]. Finally, M-CNN substantially outperforms recent approaches in a related task of video co-localization on the YouTube-Objects dataset. The method is presented in detail in Chapter 3 and the implementation is available online (see Software chapter in the end of the thesis for details).

End-To-End Learning Framework for Weakly-Supervised Semantic Segmentation. After developing a learning-based approach for video

object segmentation in Chapter 5 we extend it to the problem of weakly-supervised semantic segmentation from videos. In particular, our two-stream architecture with a convolutional memory module, shown in Figure 1.6, is used to infer pixel-level labels of the moving objects, which are in turn used to train the appearance stream for the task of semantic segmentation. This results in an EM-like framework, similar to M-CNN, which is however, fully-trainable. It is learned from a small set of videos with pixel-level, moving-object annotations and a large set of videos with weak semantic labels. We demonstrate the benefits of the fully-trainable framework by significantly improving our previous results both on the PASCAL VOC 2012 image segmentation and on the DAVIS 2016 video object segmentation benchmarks. We preset this method in Chapter 6

1.2.2 Motion and video segmentation

The problem of video object segmentation is tightly related to the task of motion segmentation - binary classification of pixels on static and independently moving. Thus, the research on these two topics is often coupled. Early works on motion segmentation were based on geometry. For instance, Philip Torr [139] proposed to identify independently moving objects by recovering information about 3D object motion from 2D flow. He then fit several motion models to the video to explain the individual motions of all the pixels, and assigned one of the models to the camera. The pixels whose motion was not explained by the camera model were then assigned a foreground label. Estimating 3D motion models from a single-camera video is ambiguous, however. Thus, for instance, in [101] the authors proposed a geometry-based model operating in the 2D space.

Other motion segmentation methods are based on heuristics instead. In [108] Papazoglou et al. first extract motion boundaries by measuring changes in optical flow field, and then use them to estimate moving regions. They identify a particular pattern in the optical flow boundary filed corresponding to moving objects, and devise an efficient algorithm for capturing such patterns. A different heuristic proposed in [134] is based on object occlusion. The authors observe that a segmentation of a scene into layers corresponding to objects and background can be revealed when either the object or the viewer move, causing parts of the scene to become hidden and others dissoccluded. Such regions can be estimated as a byproduct of optical flow algorithms. Notice however, that in the presence of a camera motion, occlusion-based methods can also segment static objects, which is desirable in some scenarios but not in others.

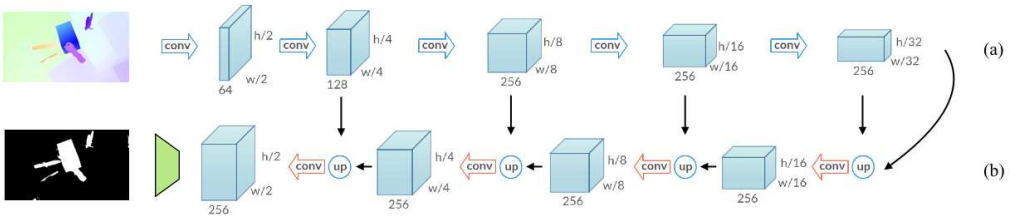


Figure 1.5 – Our motion pattern network: MP-Net. The blue arrows in the encoder part (a) denote convolutional layers, together with ReLU and max-pooling layers. The red arrows in the decoder part (b) are convolutional layers with ReLU, 'up' denotes 2×2 upsampling of the output of the previous unit. The unit shown in green represents bilinear interpolation of the output of the last decoder unit.

Unsupervised video segmentations approaches often use motion estimation as a frame-level cue to bootstrap their pipeline. For instance, Papazoglou et al. [108] propagate frame-level estimates of the moving regions throughout the video with optical flow and learn an appearance model of the moving object from these estimates. Both motion and appearance cues are then encoded as unary terms in an objective function optimized over a spatio-temporal graph of a video with GraphCut [15]. This allows them to both improve object segmentation in the frames where object moves and propagate segmentation to the frames where the object becomes static. A similar framework is proposed in [134]. Both these approaches exploit temporal relations between consecutive frames only. To overcome this limitation, Faktor and Irani [35] propose to include non-local connections into the video graph. They cluster superpixels in an appearance space and connect the most similar ones with edges that potentially span multiple frames.

Another family of video segmentation approaches is based on long-term analysis of pixel trajectories [18, 68]. In these methods each pixel is tracked individually for multiple frames with optical flow. The motion pattern of each pixel is then encoded in a compact descriptor and these descriptors are clustered to group the ones that are spatio-temporally close and exhibit similar motion into objects.

In contrast to the previous work, our approaches to motion and video segmentation are based on deep learning. Inspired by the progress in deep learning for semantic segmentation we train an FCNN on a synthetic dataset for the task of motion segmentation. We then extend this network with an appearance stream and a visual memory module to obtain a video object segmentation model.

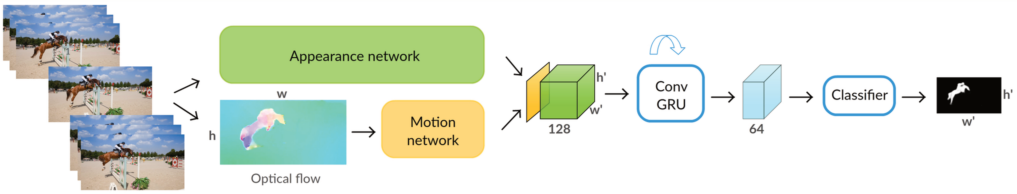


Figure 1.6 – Overview of our video object segmentation approach. Each video frame is processed by the appearance (green) and the motion (yellow) networks to produce an intermediate two-stream representation. The visual memory module combines this with the aggregated moving object representation to compute the final segmentation result.

Learning motion patterns in videos. In our CVPR'17 publication we have proposed the first learning-based approach for motion segmentation [137]. At its core is a fully convolutional network, which is learned entirely from synthetic video sequences, and their ground-truth optical flow and motion segmentation (see Figure 1.5). This encoder-decoder style architecture first learns a coarse representation of optical flow field features, and then refines it iteratively to produce motion labels at the original high-resolution. The output label of each pixel denotes whether it has undergone independent motion, i.e., irrespective of camera motion. In Section 5.4.3 we demonstrate the benefits of this learning framework on the moving object segmentation task, where the goal is to segment all objects in motion. It shows high performance on the synthetic data, where most of the issues described in Section 1.1.2 are not present, but on real videos the segmentation quality is limited. Finetuning the network on real videos helps to bridge this gap, but does not address all the model's limitations. To this end we propose to extend our motion segmentation model to a video object segmentation framework, which is described in the next paragraph.

Learning video object segmentation with visual memory. In our ICCV'17 publication we have introduced a novel two-stream neural network with an explicit memory module for video object segmentation [138] in Chapter 5. The two streams of the network, shown in Figure 1.6, encode spatial and temporal features in a video sequence respectively, while the memory module captures the evolution of objects over time. The module to build a "visual memory" in video, i.e., a joint representation of all the video frames, is realized with a convolutional recurrent unit learned from a small number of training video sequences. Given a video frame as input, our approach assigns each pixel an object or background label based on

the learned spatio-temporal features as well as the "visual memory" specific to the video, acquired automatically without any manually-annotated frames. The visual memory is implemented with convolutional gated recurrent units, which allows to propagate spatial information over time. We evaluate our method extensively on three benchmarks, DAVIS, SegTrack-v2 and Freiburg-Berkeley motion segmentation datasets. The code for both motion and video object segmentation approaches is available online (see Software chapter in the end of the thesis for details).

Part I

Weakly-Supervised Semantic Segmentation with Motion Cues

The need for weakly-supervised learning for semantic segmentation has been highlighted recently [52, 114, 143]. It is particularly important, as acquiring a training set by labeling images manually at the pixel level is significantly more expensive than assigning class labels at the image level. Recent segmentation approaches have used weak annotations in several forms: bounding boxes around objects [99, 150], image labels denoting the presence of a category [114, 143] or a combination of the two [107]. All these previous approaches only use annotation in images, i.e., bounding boxes, image tags, as a weak form of supervision. Naturally, additional cues would come in handy to address this challenging problem. As noted in [18], motion is one such cue for semantic segmentation, which helps us identify the extent of objects and their boundaries in the scene more accurately. To our knowledge, motion has not yet been leveraged for weakly-supervised semantic segmentation. In this part, we aim to fill this gap by learning an accurate segmentation model with the help of motion cues extracted from weakly-annotated videos.

The rest of this part of the thesis is organized as follows: in Chapter 2 we review the related work on weakly-supervised semantic segmentation, including appraoches with both box-level and image-level supervision. In Chapter 3 we introduce our method for weakly-supervised semantic segmentation with motion cues and provide and extensive experimental evaluation on the PASCAL VOC 2012 dataset.

Chapter 2

Related work

Contents

In this chapter we review the related work on weakly-supervised semantic segmentation. We start by defining the problem in Section 2.1. The first step towards reducing the amount of supervision for semantic segmentation is shifting from pixel-level to box level annotations. We review the box-level methods in the Section 2.2. This thesis is, however, concerned with the setting where only image- or video-level tags are available. The work related to this setting is reviewed in the Section 2.3. Finally, since the main contribution described in this part is utilizing motion as an additional cue in weakly-supervised learning, we review other methods based on other external cues in the Section 2.4.

2.1 Weakly-supervised semantic segmentation

Semantic segmentation is the task of assigning a category label to each pixel in an image (see top right image in Figure 2.1). It combines the tasks of object classification and precise localization. This makes semantic segmentation of importance for many applications, including robotics, where

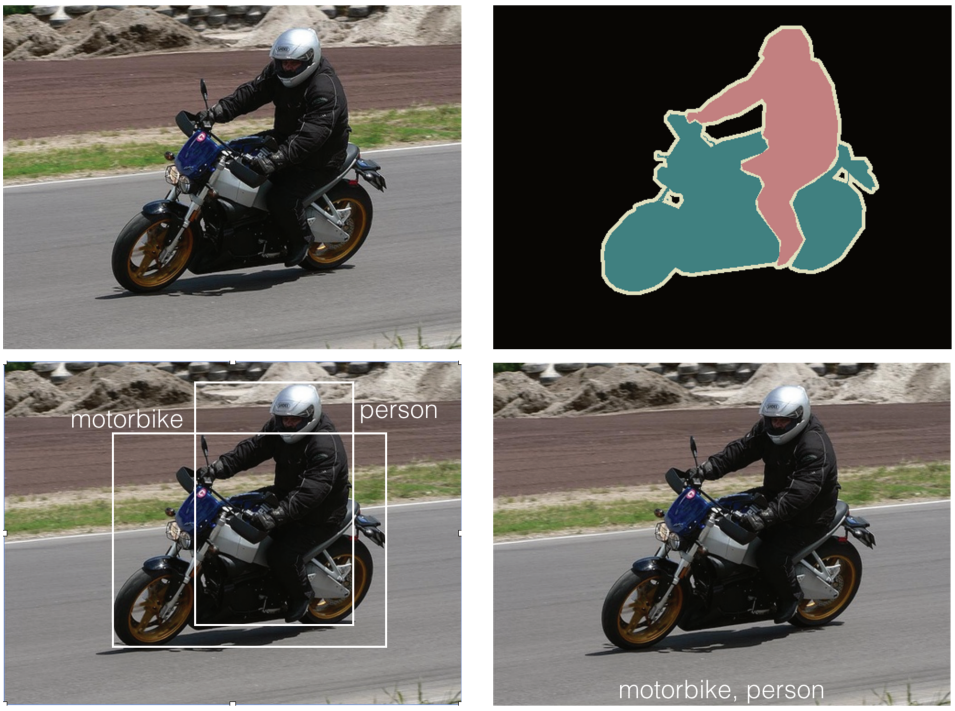


Figure 2.1 – Different degrees of supervision on an example from the PASCAL VOC'12 dataset. In the fully-supervised annotation in the top right corner pink represents category 'person', blue represents 'motorbike' and black 'background'.

knowing the precise object boundaries is necessary for grasping [84], or biomedical imaging [122], where segmenting different tissues helps in disease diagnosing. The traditional approaches for this task [21, 22] used a variety of hand-crafted visual features, namely, SIFT histograms, color, texture, in combination with a graphical or a parametric structured model. Such early attempts have been recently outperformed by FCNN methods, shown in Figure 2.2. FCNN architecture [23, 37, 95] adapts standard CNNs [77, 82] to handle input images of any arbitrary size by treating the fully connected layers as convolutions with kernels of appropriate size. This allows them to output scores for every pixel in the image. Most of these methods rely on strong pixel-level annotation to train the network. Let's denote FCNN parameters as θ, image values as $\mathbf{x}$ and pixel labels as $\mathbf{y}$. The training objective can then be formulated as maximizing the probability of the data:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}, \theta), \tag{2.1}$$

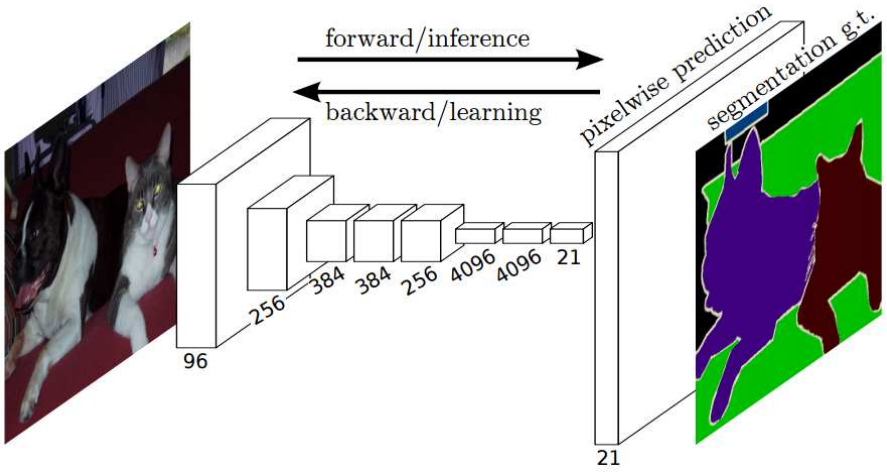


Figure 2.2 – Fully-convolutional network, that takes an image as input and directly outputs scores for every pixel. Given training images with pixel-level labels, this model can be optimized with stochastic gradient descent. (Image courtesy of Long et al. [95])

where θ^* represents the optimal model parameters. Since both $\mathbf{x}$ and $\mathbf{y}$ are observed, this objective can be directly optimized with stochastic gradient descent.

In addition to fully-supervised segmentation approaches, several weakly-supervised methods have been proposed over the years: some of them use bounding boxes [99, 163], while others rely on image labels [33, 154] (see Figure 2.1). Nearly all of these approaches can be viewed in a probabilistic formulation, where missing pixel labels are treated as latent variables. In particular, let's denote the annotations (either bounding boxes or image tags) as $\mathbf{z}$. The unobserved pixel labels $\mathbf{y}$ are latent variables then and the joint distribution can be written as:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{z}, \theta) = P(\mathbf{z}|\mathbf{y}, \mathbf{x}, \theta)P(\mathbf{y}|\mathbf{x}, \theta)P(\theta|\mathbf{x})P(\mathbf{x}), \tag{2.2}$$

where θ represents the parameters of the probability distribution of pixel labels given the image, which is usually modeled by an FCNN. Observing that the weak labels $\mathbf{z}$ are independent of $\mathbf{x}$ and θ, given the pixels labels $\mathbf{y}$, and that θ is independent of $\mathbf{x}$, we can simplify the previous equation:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{z}, \theta) = P(\mathbf{z}|\mathbf{y})P(\mathbf{y}|\mathbf{x}, \theta)P(\theta)P(\mathbf{x}). \tag{2.3}$$

If we, in addition, assume that the priors for $\mathbf{x}$ and θ are uniform, a further simplification can be achieved:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{z}, \theta) \approx P(\mathbf{z}|\mathbf{y})P(\mathbf{y}|\mathbf{x}, \theta). \tag{2.4}$$

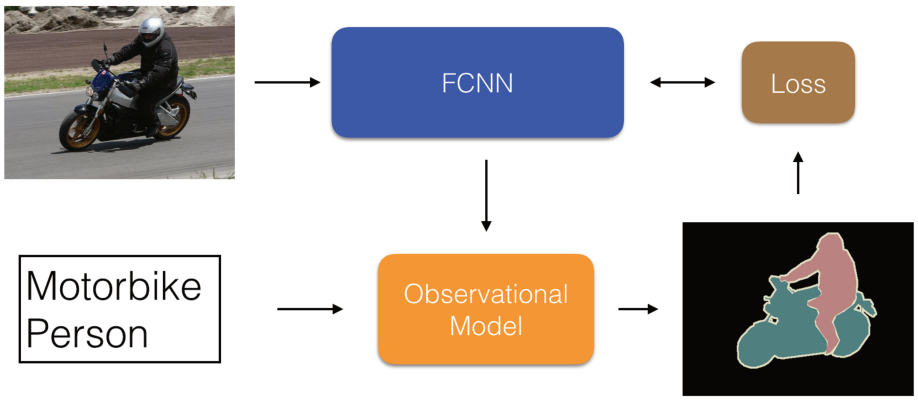


Figure 2.3 – Weakly supervised semantic segmentation in the image-label setting. The blue box represents $P(\mathbf{y}|\mathbf{x}, \theta)$, modeled by an FCNN, and the orange box represents the observation model $P(\mathbf{z}|\mathbf{y})$.

Learning a semantic segmentation model can then be formulated as maximizing the joint probability with respect to θ over a training set

$$\begin{aligned}
\theta^* &= \underset{\theta}{\operatorname{argmax}}\, P(\mathbf{x}, \mathbf{y}, \mathbf{z}, \theta) \\
&= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} P(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i, \theta) \\
&\approx \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} P(\mathbf{z}_i|\mathbf{y}_i) P(\mathbf{y}_i|\mathbf{x}_i, \theta) \\
&\approx \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log P(\mathbf{z}_i|\mathbf{y}_i) + \log P(\mathbf{y}_i|\mathbf{x}_i, \theta),
\end{aligned} \tag{2.5}$$

where N is the number of training images. The presence of the latent variables $\mathbf{y}$ in this equation makes a direct, gradient-based optimization impossible, however. Instead, statistics literature suggests using Expectation Maximization (EM) [30]: an alternating optimization technique consisting of first computing the expected values of latent variables $\hat{\mathbf{y}}$ given the previous estimate of the model parameters θ':

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log P(\mathbf{z}|\mathbf{y}) + \log P(\mathbf{y}|\mathbf{x}, \theta'), \tag{2.6}$$

and then using these expected values to maximize the join probability with respect to θ, essentially treating $\hat{\mathbf{y}}$ as the ground truth segmentations.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log P(\mathbf{z}|\hat{\mathbf{y}}) + \log P(\hat{\mathbf{y}}|\mathbf{x}, \theta) = \underset{\theta}{\operatorname{argmax}} \log P(\hat{\mathbf{y}}|\mathbf{x}, \theta), \tag{2.7}$$

where the second equality holds, since $P(\mathbf{z}|\hat{\mathbf{y}})$ is independent of θ. This process is illustrated in Figure 2.3. In the most cases the maximization step is approximated with a single iteration of stochastic gradient descent. For the expectation step, however, there are two main approaches: single-pixel one, inspired by multi-instance learning and multi-pixel one. We now describe each of them in more detail.

2.1.1 Single-pixel inference

The methods that fall into this category treat pixel label inference as an instance of the multi-instance learning (MIL) problem. In MIL images are modeled as bags of instances, some of which are positive, belonging to the categories indicated by the image tags (background is treated as an additional category present in every image) and others are negative and are thus irrelevant for the task. Since in the tag setting the number of instances of each category is usually unknown a simplifying assumption is made that only one object per category is present in an image. Computing the expected pixel labels then boils down to selecting the highest scoring pixel for every category given in the annotation and ignoring all the other pixels by not including them in the loss computation:

$$\hat{\mathbf{y}}_{ij} = \begin{cases} c, & \text{if } c \in \mathbf{z_i} \text{ and } j = \underset{j}{\text{argmax}} P(\mathbf{y}_{ij} = c|\mathbf{x}_i, \theta), \\ ignore, & \text{otherwise,} \end{cases} \tag{2.8}$$

where j is the pixel index in an image $\mathbf{x}_i$.

Single-pixel approach is appealing due to its simplicity, relatively small number of assumptions and the lack of ambiguity in label assignment. In practice, however, methods following the MIL framework are extremely susceptible to the "convergence to the most discriminative parts" problem, described in the Section 1.1.1. In particular, a model trained to segment the highest scoring regions converges to segmenting the most discriminative parts of objects, like animal faces, or car radiators and completely ignores the rest of the object. Multi-pixel approaches are addressing this issue by inferring a label assignment to all the pixels in an image.

2.1.2 Multi-pixel inference

As mentioned in the previous section, using a single pixel per category for supervising the training of a semantic segmentation model leads to convergence to segmenting object parts, not object as a whole. Thus, the

multi-pixel approaches modify the pixel label inference equation (2.8) to compute the labels for all the image pixels:

$$\hat{\mathbf{y}}_i = \underset{\mathbf{y}_i}{\operatorname{argmax}}\, f(\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i, \theta), \tag{2.9}$$

where f represents the label preference function. In its simplest form,

$$\begin{aligned} f(\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i, \theta) &= \log P(\mathbf{y}_i|\mathbf{x}_i, \theta) + \log P(\mathbf{z}_i|\mathbf{y}_i) \\ &= \sum_j \log P(y_{ij}|\mathbf{x}_i, \theta) + \log P(\mathbf{z}_i|y_{ij}), \end{aligned} \tag{2.10}$$

where

$$\log P(\mathbf{z}_i|y_{ij}) = \begin{cases} -\infty, & \text{if } y_{ij} \notin \mathbf{z}_i, \\ 0, & \text{otherwise}, \end{cases} \tag{2.11}$$

The preference function is then equivalent to the data log likelihood according to the semantic segmentation model being learned, with a constraint on the possible labels coming from the image annotation. It is easy to see though, that being put in the EM framework, such a label inference approach allows for a trivial solution: assigning all the pixels to the background category. To avoid converging to this local optimum, additional constraints on the label assignment have to be encoded in the observation model $P(\mathbf{z}|\mathbf{y})$. The way these constraints are obtained is exactly what differs most of the weakly-supervised semantic segmentation approaches from each other. We now describe the most popular formulations for the box-level and image-level settings in the Sections 2.2 and 2.3 respectively.

2.2 Box-level approaches

In this setting, a rough localization of objects in images is provided in the form of bounding boxes (bottom left example in Figure 2.1). This significantly simplifies the label inference problem, by providing constraints on the extent and location of the object segmentations. The most straightforward approach is directly treating the bounding boxes as segmentations (as in the Bbox-Rect method in [107]). In this case label inference becomes a static assignment:

$$\hat{y}_{ij} = \begin{cases} c, & \text{if } j \in B_{ci}, \\ bkg, & \text{otherwise}, \end{cases} \tag{2.12}$$

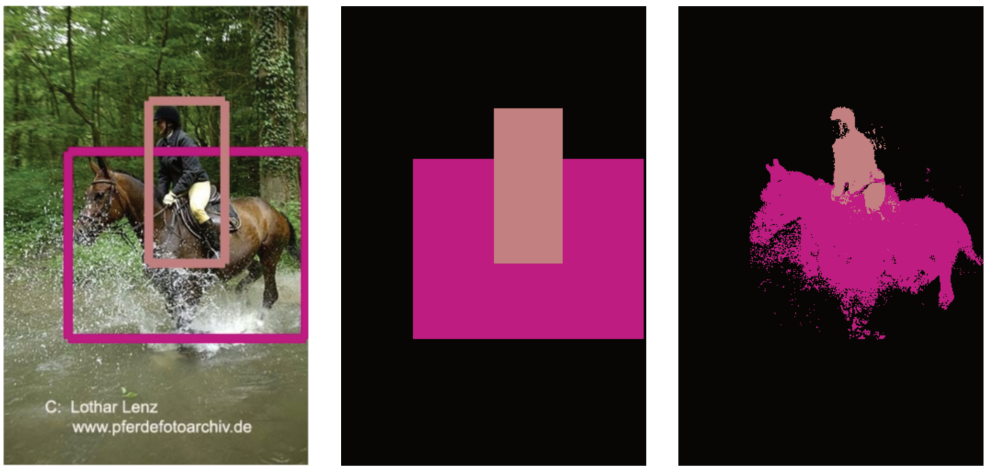


Figure 2.4 – Exploiting bounding boxes for supervision. Boxes can either be directly converted to pixel labels, as shown in the image in the middle, or first refined with a foreground/background segmentation technique. The last image is the result of the DenseCRF [76] refinement on the initial box segmentation obtained by the method of [107]. (Image courtesy of Papandreou et al. [107])

where B_{ci} is the union of all the bounding boxes of category c in the image i. To find the θ^* one can then directly optimize the data likelihood in the equation (2.7) with stochastic gradient descent. However, as shown in the middle image in Figure 2.4, many background pixels get a foreground label assignment with this approach, further exaggerating the foreground-background correlation issue of weakly-supervised semantic segmentation (see Section 1.1.1).

Several works suggest to mitigate this issue by refining the initial crude bounding box segmentations, by combining them with unsupervised region proposal methods (see [163] and [28]) or by utilizing foreground/background, colour-based segmentation techniques (see Figure 2.4). In particular, [152] and [69] are using GraphCut and [107] are relying on the DenseCRF [76]. Both are based on optimizing an energy function over the pixel label assignment:

$$E(\mathbf{y}) = \sum_i u(y_i) \; + \sum_{(i,j)\in\mathcal{E}} \psi(y_i, y_j), \tag{2.13}$$

where $y_i \in \{0, 1\}\ \forall i$, u is a unary potential encoding the initial belief about the pixel's label, ψ is a colour-based label compatibility function and $\mathcal{E}$ denotes all pairs of neighboring pixels in the image. The difference between the two approaches lies in the definition of u and $\mathcal{E}$. In GraphCut, to define

Figure 2.5 – Sample result from the iterative box refinement approach of [69]. (Image courtesy of Khoreva et al. [69])

the unary term a foreground/background colour-based GMM is learned from the pixels inside the box treated as foreground and all the other pixels treated as background. It is then used to compute the unary scores for the pixels inside of the box. In contrast, [107] simply assign the α fraction of the pixels in the center of the box the score 1 for the foreground label and 0 for the background and the pixels outside the box get the opposite assignment. The neighborhood graph $\mathcal{E}$ is defined as a pots model for GraphCut and as a fully-connected model for the DenseCRF. In practice, both approaches produce a more accurate pixel label assignment and, as a result, improve the performance of the learned semantic segmentation model.

One disadvantage of the box refinement approach is that it ignores the semantic information being learned, since the label preference function is independent of θ: $f(\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i, \theta) = f(\mathbf{y}_i, \mathbf{x}_i, \mathbf{z}_i)$. Both methods described above address this issue. In [69] the authors modify the EM inference by performing full model optimization in the M-step, with refined boxes as labels. The predictions of the learned model are then used to obtain more accurate unary potentials in the GraphCut energy function (3.4) and thus compute an improved estimate of the pixel labels $\hat{\mathbf{y}}$. As shown in Figure 2.5, this process is iterated several times, allowing their method to achieve fully-supervised performance with bounding box annotations.

A more direct approach to incorporating model predictions in the label inference framework is proposed in [107]. The authors formulate the observation model as:

$$\log P(\mathbf{z}_i|\mathbf{y}_i) = \sum_j \phi(y_{ij}, \mathbf{z}_i) + const, \tag{2.14}$$

where *const* is a normalization constant and the function ϕ is encouraging the label assignment to be consistent with the box annotations:

$$\phi(y_{ij} = c, \mathbf{z_i}) = \begin{cases} b, \text{if } j \in B_{ci}, \\ 0, \text{otherwise}, \end{cases} \tag{2.15}$$

and b is a predefined constant. In other words, for the pixels that fall into one of the bounding boxes the probability of the corresponding category is increased and the probabilities of the remaining categories are left unchanged. This method does performs slightly better than naively treating bounding boxes as segmentation but is outperformed even by the simple box refinement approach. This is due to the fact that low level image information exploited by the box refinement provides a strong cue, which is completely ignored by this method. Nevertheless, the formulation in which external knowledge is incorporated into the observation model and used in the E-step of the expectation maximization algorithm is sufficiently general and is used by many approaches we are about to review.

2.3 Image-level approaches

Early image-level methods for weakly-supervised semantic segmentation operated on the level of segments [33] or superpixels [154], not pixels, which allowed them to obtain non-trivial solutions with a simple constraint that every image label must be matched with at least one segment in the E-step. Their performance, however, was limited by the performance of the segmentation method used to generate the input.

Some early, FCNN-based approaches, like [110] extend the MIL framework used for object detection [27, 125] to segmentation by treating the pixel with the highest prediction score for a category as its positive sample when computing the loss. However, as we explained in the Section 2.1.1, this approach is susceptible to standard issues suffered by MIL, like converging to the most discriminative parts of objects. An alternative MIL strategy is used in [114], by introducing a soft aggregation function that translates pixel-level FCNN predictions into an image label distribution. In other words, the latent variables $\mathbf{y}$ are removed from the model and the distribution over the image-level labels $\mathbf{z}$ depends directly on $\mathbf{x}$ and θ:

$$P(\mathbf{z}_i|\mathbf{y}_i, \mathbf{x}_i, \theta) = P(\mathbf{z}_i|\mathbf{x}_i, \theta) = aggreg(S_i), \tag{2.16}$$

where $S_i := \{s_{ij}, \forall j\}$ is the set of pixel category scores compute by an FCNN, $S_i = fcnn(\mathbf{x}_i, \theta)$ and $aggreg$ is a differentiable score aggregation procedure designed by the authors. Getting rid of the latent variables allows for a direct, gradient-based optimization to find θ^*. This strategy works better in practice than [110], but requires training images that contain only a single object, as well as explicit background images. In addition, it produces inaccurate segmentations, which are particularly prone to false

positives. To achieve competitive results, the authors use a complex post-processing step, computing image-level priors with multi-scale segmentation algorithm in testing.

Weakly-supervised FCNNs in [107] and [109] are the most relevant for our work. They operate in the EM-framework and, like the adaptive, box-level approach in [107], incorporate constraints on the predicted pixel labels in the observation model $P(\mathbf{z}|\mathbf{y})$. Since no information about the cardinality, location or extent or the objects in an image is provided in this case, the constraints encoded in the function ϕ (equation (2.15)) are more general. [107] present two formulation, one similar to (2.15), where the condition is applied to all the pixels in an image, and the value of b is fixed

$$\phi(y_{ij} = c, \mathbf{z_i}) = \begin{cases} b, \text{if } c \in z_i, \\ 0, \text{otherwise}, \end{cases} \tag{2.17}$$

and another, where b is adaptively computed for every image in every iteration to ensure a certain proportion of foreground and background labels in $\hat{\mathbf{y}}_i$. In particular, they constraint at least 20% of the pixels in an image to be assigned to each of the image-label categories, at least 40% to the background and enforce that no pixel is assigned to class c, if $c \notin \mathbf{z}_i$. An heuristic-based algorithm is used to compute the bias values that would result in a pixel label assignment $\hat{\mathbf{y}}_i$ satisfying the constraints above. This approach performs significantly better than the baseline with a fixed b, solving the convergence to parts issue. It was extended in [109] to include generic linear constraints on the label space, by formulating label prediction as a convex optimization problem. In practice, however, this extension performs similarly to the simpler method of [107].

Both these methods showed state-of-the-art results on the VOC'12 dataset. Figure 2.6 demonstrates the predictions of [107]. It is able to correctly localize the objects, but the segmentations are blobby, failing to capture the exact object boundaries. This is due to the fixed cardinality constraints, used to compute the bias term b in equation (2.17). Indeed, the assumption that any object in any image occupies at least 20% of the area does not hold in many realistic images. Thus, the same background correlation issue that was observed for the approach using bounding boxes as segmentation in the Section 2.2 manifests itself here as well. In fact, any general constraint would not hold for all the images, due to the large variability of the visual world. More specific category-level and, ideally, instance-level constraints are necessary for learning accurate segmentation. The methods described in the next section explore sources of such constraints and efficient ways of integrating them in the EM-framework.

Figure 2.6 – Sample results of the weakly-supervised method of [107] on the images from the VOC'12 dataset.

2.4 Exploiting external cues

One example of external cues that are helpful in learning accurate segmentations are constraints on the object size and location provided by the user. Such constraints can be represented in the form of pixel- or box-level annotations, giving rise to the fully-supervised and box-supervised scenarios described above. In [109] and [10] the authors suggest two complementary approaches to reducing the effort in collecting the object location and extent annotations. The former work addresses the issue of generic cardinality constraints of [107]. Instead of assuming that any object occupies at least 20% of the image area they propose to make this constraint category specific. For example, for large objects, like trains or trucks the number can be larger, whereas for small objects like birds or cups it can be smaller. The exact fraction for every category is determined by the user, slightly increasing the annotation effort, but leading to a significant improvement in the overall performance.

In [10], the authors consider the complementary issue of object location ambiguity. Their method extends the MIL-based approach of [110], but instead of fully relying on the model predictions $P(\mathbf{y}_i|\mathbf{x}_i, \theta)$ to select

a positive sample for each category, they suggest to use the minimal form of user annotation: single pixel supervision. In particular, the annotator, in addition to identifying the presence of the categories, labels a single pixel inside of each object instance in an image. This external information indeed allows to improve the model's precision and increase the overall performance at a very little additional annotation cost. However, the convergence to parts issue, common to the MIL-based approaches, still limits the method's performance. To address it, the authors resort to another external cue, commonly used by weakly-supervised semantic segmentation methods: objectness.

Objectness is defined as a probability of a pixel belonging to an object, irrespective of the object category. It can thus be used as a natural source of instance-specific constraints on the object extent. A lot of methods for estimating objectness have been proposed, from early, heuristic-based that operated on the box level [8], or pixel-level [25], to the more recent neural network-based approaches, like [89] and [87]. The later obtain superior performance, but themselves require pixel-level annotations to train. Thus, [10] are using the early, heuristic-based method of [8] and compute pixel level scores by aggregating the scores of the bounding boxes containing the pixel. Since their approach does not explicitly compute the pixel label assignment $\hat{\mathbf{y}}_i$, they incorporate the objectness directly into the loss function, used to learn the semantic segmentation model parameters:

$$\mathcal{L}(S, G, \mathbf{z}, O) = \mathcal{L}_{img}(S, \mathbf{z}) - \mathcal{L}_{point}(S, G) - \mathcal{L}_{obj}(S, O), \tag{2.18}$$

where S represents the pixel-level category scores computed by the FCNN, G is the sparse set of manually annotate pixels and O is the set of the objectness scores. The individual components of the loss $\mathcal{L}_{img}$ and $\mathcal{L}_{point}$ encode the image-level and pixel-level constraints (only categories in the image tag can be predicted and for the manually annotated pixels the prediction has to match the annotation), whereas $\mathcal{L}_{obj}$ encodes the soft objectness constraints:

$$\mathcal{L}_{obj}(S, O) = \sum_i O_i \log\left(\sum_{c \in obj} S_{ic}\right) + (1 - O_i) \log\left(1 - \sum_{c \in obj} S_{ic}\right), \tag{2.19}$$

where *obj* is the set of non-background categories. This formulation encourages assigning high scores to object categories for pixels with high objectness and high background score to pixels with low objectness. In combination with the single pixel supervision, this approach obtains both relatively high precision and recall in testing, outperforming the generic constraint-based

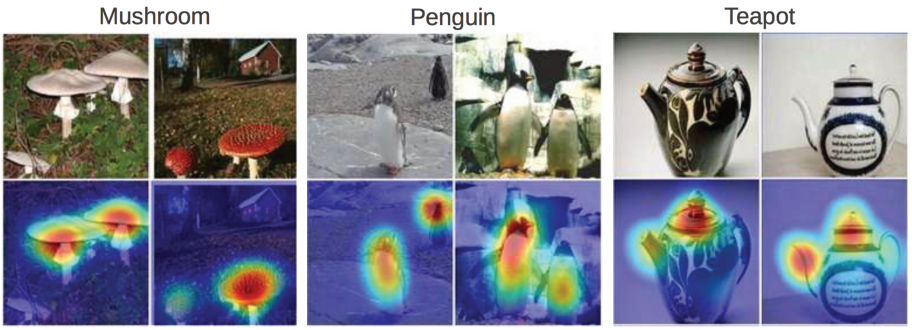


Figure 2.7 – Examples of the class activation maps produced by the method of [162] on the Caltech256 dataset. (Image courtesy of Zhou et al. [162])

method of [107]. Its reliance on additional human supervision remains a limitation though.

An alternative way of automatically obtaining object localization cues has been proposed in [75]. Their work builds upon a weakly-supervised object localization method of [162], where the authors design a mechanism for extracting approximate object localization maps, which they call class activation maps (CAMs), from a network trained for image classification (see Figure 2.7). These maps are category specific and, in addition to an estimate of the object extent, provide a pixel-level belief in the correctness of the prediction. Thus, although CAMs are inaccurate overall (see, for instance, the teapot example in Figure 2.7, where only parts of the objects are segmented), a small subset of confident pixels can be extracted from every image and used as annotations. These annotations are similar to the point-supervision in [10], but can be obtained automatically. The resulting label assignment can be written as:

$$\hat{y}_{ij}^{cam} = \begin{cases} c, & \text{if } j \in G_{ic}^{cam}, \\ ignore, & \text{otherwise}, \end{cases} \tag{2.20}$$

where G_{ic}^{cam} is the set of the confident CAM pixels labeled with the category c.

To increase the recall of the learned model, instead of computing pixel-level objectness [75] utilize low-level image cues with the DenseCRF model (see Section 2.2). In particular, the unary term u in the energy function equation (3.4) is initialized with the model predictions $P(\mathbf{y}_i|\mathbf{x}_i, \theta)$ and they are refined using the appearance constancy assumption, to obtain pixel labels $\hat{\mathbf{y}}_i^{crf}$. This assumption implies that if two pixels in an image have a similar appearance they are likely to belong to the same category. This

refinement step allows to both suppress incorrect predictions on the background regions and prevent the convergence to parts behavior, by expanding the segmentation to the full extent of an object in many cases (the method of [75] also has a separate component responsible for expanding the predictions, but their experimental evaluation demonstrates that it's effect on the overall performance is marginal, thus we choose not to describe it here). Notice that this approach produces two independent label estimates for each image: $\hat{\mathbf{y}}_i^{cam}$ and $\hat{\mathbf{y}}_i^{crf}$. The data log likelihood equation (2.7) thus changes to:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \log P(\hat{\mathbf{y}}^{cam}|\mathbf{x}, \theta) + \log P(\hat{\mathbf{y}}^{crf}|\mathbf{x}, \theta), \tag{2.21}$$

The optimization has to balance the hard localization constraints and the segmentation refinement objective imposed by the DenseCRF, resulting in the model both avoiding convergence to parts behavior and learning to precisely capture the object boundaries.

Another way of exploiting class activation maps as an object localization cue is explored in the recent work of [106]. The authors combine CAMs with objectness maps, computed with a neural network, which is itself trained in a weakly-supervised way on thousands on bounding-box annotated images. To estimate the labels $\hat{\mathbf{y}}$ they design an heuristic-based algorithm, that combines class activations and generic object segmentations on the level of connected components. The algorithm strives to obtain both high precision and recall in the inferred labels, ignoring connected components for which a confident prediction can not be made. This approach achieves top results on the PASCAL VOC'12 semantic segmentation dataset, outperforming the method of [75], but requires expensive bounding box annotations to train the objectness model and relies on many heuristics that are not likely to generalize to different settings.

Our work introduces an object localization cue that is complementary to the ones described above: independent object motion. Indeed, objects moving independently from the camera in a video can be precisely segmented, providing both localization and an accurate segmentation of an object for free. Motion cues have been explored in the weakly-supervised object detection literature before, e.g., in [116] and [91], but we are the first to propose a weakly-supervised semantic segmentation method that capitalizes on the motion information.

Chapter 3

Our Approach

Contents

Our proposed framework is based on fully convolutional neural networks (FCNNs) [23, 37, 95, 160], which extend deep CNNs, and are able to classify every pixel in an input image in a single forward pass. While

Image DeepLab (full) EM-Adapt (weak) M-CNN (weak)

Figure 3.1 – Comparison of state-of-the-art fully [23] and weakly [107] supervised methods with our weakly-supervised M-CNN model.

FCNNs show state-of-the-art results on segmentation benchmark datasets, they require thousands of pixel-level annotated images to train on—a requirement that limits their utility. Recently, there have been some attempts [107, 109, 110, 114] to train FCNNs with weakly-annotated images, but they remain inferior in performance to their fully-supervised equivalents (see Figure 3.1). In this chapter, we develop a new CNN variant named M-CNN, which leverages motion cues in weakly-labeled videos, in the form of unsupervised motion segmentation, e.g., [108]. It builds on the architecture of FCNN by adding a motion segmentation based label inference step, as shown in Figure 3.2. In other words, predictions from the FCNN layers and motion segmentation jointly determine the loss used to learn the network (see §3.1.2).

Our approach uses unsupervised motion segmentation from real-world videos, such as the YouTube-Objects [116] and the ImageNet-VID [1] datasets, to train the network. In this context, we are confronted with two main challenges. The first one is that even the best-performing algorithms cannot produce good motion segmentations consistently, and the second one is the ambiguity of video-level annotations, which cannot guarantee the presence of object in all the frames. We develop a novel scheme to address these challenges automatically without any manual annotations, apart from the labels assigned at the video level, denoting the presence of objects somewhere in the video. To this end, we use motion segmentations as soft constraints in the learning process, and also fine-tune our network with a small number of video shots to refine it.

We evaluated the proposed method on two related problems: semantic segmentation and video co-localization. When trained on weakly-annotated videos, M-CNN outperforms state-of-the-art EM-Adapt [107] on the PASCAL VOC 2012 image segmentation benchmark [34]. Furthermore, our trained model, despite using only 150 video labels, achieves performance similar to EM-Adapt trained on more than 10,000 VOC image labels. Augmenting our training set with 1,000 VOC images results in a further gain, achieving the best performance on VOC 2012 test set in the weakly-supervised setting (see §3.2.4). On the video co-localization task, where the goal is to localize common objects in a set of videos, M-CNN substantially outperforms a recent method [79] by over 16% on the YouTube-Objects dataset.

The contributions of this work are twofold: (i) We present a novel CNN framework for segmentation that integrates motion cues in video as soft constraints. (ii) Experimental results show that our segmentation model learned from weakly-annotated videos can indeed be applied to evaluate on challenging benchmarks and achieves top performance on semantic segmentation as well as video co-localization tasks.

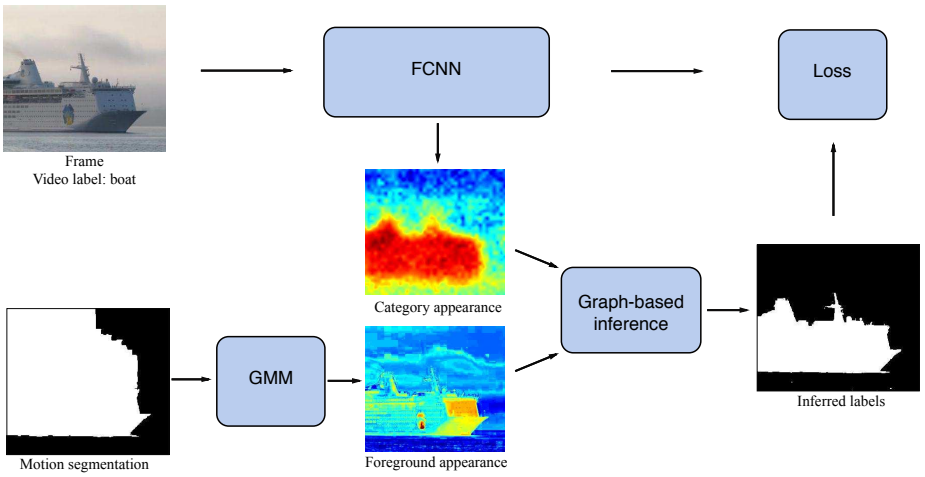


Figure 3.2 – Overview of our M-CNN framework, where we show only one frame from a video example for clarity. The soft potentials (foreground appearance) computed from motion segmentation and the FCNN predictions (category appearance) jointly determine the latent segmentation (inferred labels) to compute the loss, and thus the network update.

3.1 Learning semantic segmentation from video

We train our network by exploiting motion cues from video sequences. Specifically, we extract unsupervised motion segments from video, with algorithms such as [108], and use them in combination with the weak labels at the video level to learn the network. We sample frames from all the video sequences uniformly, and assign them the class label of the video. This collection forms our training dataset, along with their corresponding motion segments.

The parameters of M-CNN are updated with a standard mini-batch SGD, similar to other CNN approaches [107], with the gradient of a loss function. Here, the loss measures the discrepancy between the ground truth segmentation label and the label predicted at each pixel. Thus, in order to learn the network for the semantic segmentation task, we need pixel-level ground truth for all the training data. These pixel-level labels are naturally latent variables in the context of weakly-supervised learning. Now, the task is to estimate them for our weakly-labeled videos. An ideal scenario in this setting would be near-perfect motion segmentations, which can be directly used as object ground truth labels. However, in practice, not only are the segmentations far from perfect (see Figure 3.3), but also fail to

capture moving objects in many of the shots. This makes a direct usage of motion segmentation results suboptimal. To address this, we propose a novel scheme, where motion segments are only used as soft constraints to estimate the latent variables together with object appearance cues.

The other challenges when dealing with real-world video datasets, such as YouTube-Objects and ImageNet-VID, are related to the nature of video data itself. On one hand, not all parts of a video contain the object of interest. For instance, a video from a show reviewing boats may contain shots with the host talking about the boat, and showing it from the inside for a significant part—content that is unsuitable for learning a segmentation model for the VOC 'boat' category. On the other hand, a long video can contain many nearly identical object examples which leads to an imbalance in the training set. We address both problems by fine-tuning our M-CNN with an automatically selected, small subset of the training data.

3.1.1 Network architecture

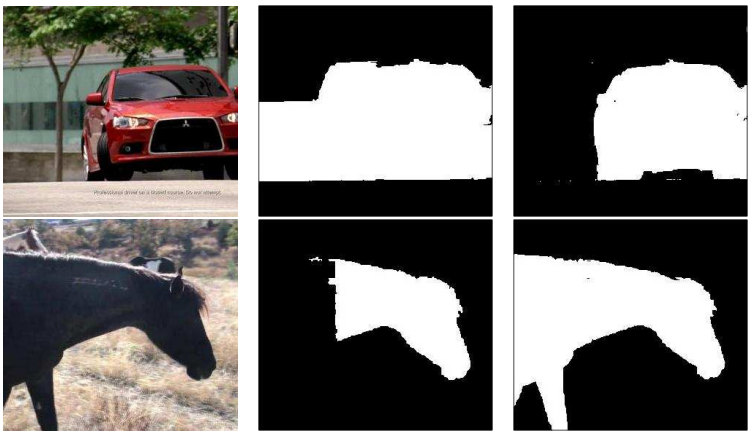
Our network is built on the DeepLab model for semantic image segmentation [23]. It is an FCNN, obtained by converting the fully-connected layers of the VGG-16 network [129] into convolutional layers. A few other changes are implemented to get a dense network output for an image at its full resolution efficiently. Our work builds on this network. We develop a more principled and effective label prediction scheme involving motion cues to estimate the latent variables, in contrast to the heuristic size constraints used in [107], which is based on DeepLab.

3.1.2 Estimating latent variables with label prediction

Given an image of N pixels, let $\mathbf{s}$ denote the output of the softmax layer of the convolutional network. Then, $s_i^l \in [0, 1]$ is the prediction score of the network at pixel i for label l. The parameters of the network are updated with the gradient of the loss function, given by:

$$\mathcal{L}(\mathbf{y}, \mathbf{s}) = \sum_{i=1}^{N} \sum_{c=0}^{C} \delta(y_i - c) \log(s_i^l), \tag{3.1}$$

where $\mathbf{y}$ denotes ground truth segmentation labels in the fully-supervised case, $\mathbf{s}$ is the current network prediction, and $\delta(y_i - c)$ is the Dirac delta function, i.e., $\delta(y_i - c) = 1$, if $y_i = c$, and 0 otherwise. The segmentation label y_i of pixel i takes values from the label set $\mathbf{C} = \{0, 1, \ldots, C\}$, containing



Frame Motion seg. Our label prediction

Figure 3.3 – Examples highlighting the importance of label prediction for handling imprecise motion segmentations (second column). The soft potentials computed from motion segments along with network predictions produce better labels (third column) to learn the network.

the background class (0) and C object categories. Naturally, in the weakly-supervised case, ground truth segmentation labels are unavailable, and $\mathbf{y}$ represents latent segmentation variables, which need to be estimated. We perform this estimation with soft motion segmentation cues in this work.

Given the motion segmentation $\mathbf{m} = \{m_i | i = 1, \ldots, N\}$, where $m_i \in \{0, 1\}$ denotes whether a pixel i belongs to foreground (1) or background (0).[1] The regions assigned to foreground can represent multiple objects when the video is tagged with more than one object label. A simple way of transforming motion segmentation labels m_i into estimated latent semantic segmentation labels $\hat{y}_i$ is with a hard assignment:

$$\hat{y}_i = \begin{cases} c, \text{if } m_i = 1, \\ bkg, \text{otherwise}, \end{cases} \tag{3.2}$$

where c represents the video label. This hard assignment is limited to videos containing a single object label, and also makes the assumption that motion segments are accurate and can be used as they are. We will see in our experiments that this performs poorly when using real-world video datasets (cf. 'M-CNN* hard' in Table 3.1). We address this by using motion cues as soft constraints for estimating the label assignment $\hat{\mathbf{y}}$ in the following.

1. We do not include an index denoting the frame number in the video for brevity.

Inference of the segmentation $\hat{\mathbf{y}}$. Our label inference procedure is motivated by the general EM scheme (see Section 2.1). We modify it by incorporating the motion cues $\mathbf{m}$ in the joint probability distribution. In particular, an additional probability term linking motion labels with category labels is added to the distribution: $P(\mathbf{m}|\mathbf{y})$. The label inference equation (2.6) then takes the following form:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log P(\mathbf{m}|\mathbf{y}) + \log P(\mathbf{z}|\mathbf{y}) + \log P(\mathbf{y}|\mathbf{x}, \theta^{'}), \tag{3.3}$$

In practice, we formulate the label preference function used to compute the pixel-level segmentation $\hat{\mathbf{y}}$ as an energy function $E(\mathbf{y})$ defined by:

$$E(\mathbf{y}) = \sum_{i \in \mathcal{V}} \left(\psi_i^m(r_i) + \alpha \psi_i^{fc}(s_i^{x_i}) \right) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(y_i, y_j), \tag{3.4}$$

where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of all the pixels, r_i denotes the RGB color at pixel i and the set $\mathcal{E}$ denotes all pairs of neighboring pixels in the image. Unary terms ψ_i^m and ψ_i^{fc} are computed from motion cues and current predictions of the network respectively, with α being a scalar parameter balancing their impact. The pairwise term ψ_{ij} imposes a smoothness over the label space.

The first unary term ψ_i^m captures the appearance of all foreground objects obtained from motion segments. To this end, we learn two Gaussian mixture models (GMMs), one each for foreground and background, with RGB values of pixel colors, similar to standard segmentation methods [108, 123]. The foreground GMM is learned with RGB values of all the pixels assigned to foreground in the motion segmentation. The background GMM is learned in a similar fashion with the corresponding background pixels. Given the RGB values of a pixel i, $\psi_i^m(r_i)$ is the negative log-likelihood of the corresponding GMM (background one for $l = 0$ and foreground otherwise). Using motion cues to generate this soft potential ψ_i^m helps us alleviate the issue of imperfect motion segmentation. The second unary term ψ_i^{fc} represents the learned object appearance model determined by the current network prediction $s_i^{y_i}$ for pixel i, i.e., $\psi_i^{fc}(s_i^{y_i}) = -\log(s_i^{y_i})$.

The pairwise term is based on a contrast-sensitive Potts model [15, 123] as:

$$\psi_{ij}(y_i, y_j) = \lambda(1 - \Delta(i,j))(1 - \delta(y_i - y_j)) \frac{\exp(-\gamma ||r_i - r_j||^2)}{\operatorname{dist}(i,j)}, \tag{3.5}$$

where r_i and r_j are colors of pixels i and j, λ is a scalar parameter to balance the order of magnitude of the pairwise term with respect to the

unary term, and γ is a scalar parameter set to 0.5 as in [108]. The function $\text{dist}(i, j)$ is the Euclidean distance between pixels. The Dirac delta function $\delta(y_i - y_j)$ ensures that the pairwise cost is only applicable when two neighboring pixels take different labels. In addition to this, we introduce the term $(1 - \Delta(i, j))$, where $\Delta(i, j) = 1$ if pixels i and j both fall in the boundary region around the motion segment, and 0 otherwise. This accounts for the fact that motion segments may not always respect color boundaries, and allows the minimization algorithm to assign different labels to neighboring pixels around motion edges.

We minimize the energy function ((3.4)) with an iterative GrabCut-like [123] approach, wherein we first apply the alpha expansion algorithm [16] to get a multi-label solution, use it to re-estimate the (background and foreground) GMMs, and then repeat the two steps a few times. We highlight the importance of our label prediction technique with soft motion-cue constraints in Fig. 3.3. Here, the original, binary motion predictions are imprecise (bottom) or incorrect (top), whereas using them as soft constraints in combination with the network prediction results in a more accurate estimation of the latent segmentation variables.

3.1.3 Fine-tuning M-CNN

We learn an initial M-CNN model from all the videos in the dataset which have sufficient motion information (see §3.2.2 for implementation details). To refine this model we add a fine-tuning step, which updates the parameters of the network with a small set of unique and reliable video examples. This set is built automatically by selecting one shot from each video sequence, whose motion segment has the highest overlap (intersection over union) score with the current M-CNN prediction. The intuition behind this selection criterion is that our MCNN has already learned to discriminate categories of interest from the background, and thus, its predictions will have the highest overlap with precise motion segmentations. This model refinement leverages the most reliable exemplars and avoids near duplicates, often occurring within one video. In Section 3.2.3 we demonstrate the importance of this step for dealing with real-world non-curated video data.

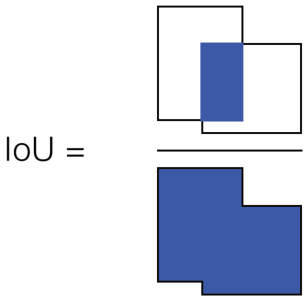


Figure 3.4 – Illustration of the intersection over union measure between two rectangular segments.

3.2 Experiments

3.2.1 Datasets and evaluation

In this section, we review the metrics and the datasets used for evaluating weakly supervised semantic segmentation.

To evaluate the performance of weakly supervised semantic segmentation, we use the standard measure for segmentation problems: intersection over union (IoU). It captures both precision and recall of the predicted segmentation against the groundtruth according to the following equation:

$$IoU = \frac{Intersection}{Union}, \tag{3.6}$$

which is also visualized in Figure 3.4. We compute IoU for each class as well as the average over all the classes, including background, following standard protocols of [34].

We also evaluate our segmentation results on the video datasets in the co-localization setting. It these experiments we are measuring how well our algorithm is able to extract the moving objects from videos given video-level semantic labels. Since the video datasets we are working with provide bounding box-level annotations only, we measure the performance on this task with the CorLoc measure ([108], [116]), which is defined as the percentage of images with IoU score, between ground truth and predicted bounding boxes, more than 0.5. To convert the segmentations predicted by our model into bounding boxes we follow the protocol of [108] and extract a tight box around the largest connected component in the segmentation.

We trained our video-based approach in two settings. The first one is on purely video data, and the second on a combination of image and video data.

We performed experiments primarily with the weakly-annotated videos in the YouTube-Objects v2.2 dataset [2]. Additionally, to demonstrate that our approach adapts to other datasets automatically, we used the ImageNet video (ImageNet-VID) dataset [1]. The weakly-annotated images to train our network jointly on image and video data were taken from the training part of the PASCAL VOC 2012 segmentation dataset [34] with their image tags only. We then evaluated variants of our method on the VOC 2012 segmentation validation and test sets.

The **VOC 2012** dataset has 20 foreground object classes and a background category. It is split into 1464 training, 1449 validation and 1456 test images. For experiments dealing with the subset of 10 classes in common with YouTube-Objects, we treat the remaining 10 from VOC as irrelevant classes. In other words, we exclude all the training/validation images which contain only the irrelevant categories. This results in 914 training and 909 validation images. In images that contain an irrelevant class together with any of the 10 classes in YouTube-Objects, we treat their corresponding pixels as background for evaluation. Some of the state-of-art methods [107, 109] use an augmented version of the VOC 2012 dataset, with over 10,000 additional training images [51]. Naturally the variants trained on this large dataset perform significantly better than those using the original VOC dataset. We do not use this augmented dataset in our work, but report state-of-the-art results due to our motion cues.

The **YouTube-Objects** dataset consists of 10 classes, with 155 videos in total. Each video is annotated with one class label and is split automatically into shots, resulting in 2511 shots overall. For evaluation, one frame per shot is annotated with a bounding box in some of the shots. We use this exclusively for evaluating our video co-localization performance in Section 3.2.5.

For experiments with **ImageNet-VID**, we use 795 training videos corresponding to the 10 classes in common with YouTube-Objects. ImageNet-VID has bounding box annotations produced semi-automatically for every frame in a video shot (2120 shots in total). We accumulate the labels over a shot and assign them as class labels for the entire shot. As in the case of YouTube-Objects, we only use class labels at the video level, and none of the available additional annotations.

3.2.2 Implementation details

Motion segmentation. In all our experiments we used [108], a state-of-the-art method for motion segmentation. We perform two pruning steps before training the network. First, we discard all shots with less than 20

frames (2× the batch size of our SGD training). Second, we remove shots without relevant motion information: (i) when there are nearly no motion segments, or (ii) a significant part of the frame is assigned to foreground. We prune them out by a simple criterion based on the size of the foreground segments. We keep only the shots where the estimated foreground occupies between 2.5% and 50% of the frame area in each frame, for at least 20 contiguous frames in the shot. In cases where motion segmentation fails in the middle of a shot, but recovers later, producing several valid sequences, we keep the longest one. These two steps combined remove about a third of the shots, with 1675 and 1691 shots remaining in YouTube-Objects and ImageNet-VID respectively. We sample 10 frames uniformly from each of these remaining shots to train the network.

Training. We use a mini-batch of size 10 for SGD, where each mini-batch consists of the 10 frame samples of one shot. Our CNN learning parameters follow the setting in [107]. The initial learning rate is set to 0.001 and multiplied by 0.1 after a fixed number of iterations. We use a momentum of 0.9 and a weight decay of 0.0005. Also, the loss term $\delta(y_i - c)\log(s_i^c)$ computed for each object class l with num_l training samples, in ((3.1)), is weighted by $\min_{j=1\ldots L} \text{num}_j/\text{num}_l$. This accounts for imbalanced number of training samples for each class in the dataset.

In the energy function ((3.4)), the parameter α, which controls the relative importance of the current network prediction and the soft motion cues, is set to 1 when training on the entire dataset. It is increased to 2 for fine-tuning, where the predictions are more reliable due to an improved network. We perform 4 iterations of the graph cut based inference algorithm, updating the GMMs at each step. The inference algorithm is either alpha expansion (for videos with multiple objects) or graph cut (when there is only one object label for the video). Following [108], we learn GMMs for a frame t with the motion segments from all the 10 frames in a batch, weighting each of them inversely according to their distance from t. The fine-tuning step is performed very selectively with the best shot for each video, where the average overlap is no less than 0.2. A systematic evaluation on the VOC 2012 validation set confirmed that the performance is not sensitive to the number of iterations and the α parameter. More details on this and our implementation in the Caffe framework [64] are available online [3].

Table 3.1 – Performance of M-CNN and EM-Adapt variants, trained with YouTube-Objects, on the VOC 2012 validation set. '*' denotes the M-CNN models without fine-tuning. 'M-CNN* hard' is the variant without the label prediction step. 'M-CNN' is our complete method: with fine-tuning and label prediction.

Method	FOV	bkg	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Average
EM-Adapt	small	65.7	25.1	20.5	9.3	21.6	23.7	12.4	17.7	14.9	19.5	25.4	23.2 ± 3.0
EM-Adapt	large	69.1	12.9	14.7	9.0	12.9	15.4	5.6	9.9	7.8	15.9	23.0	17.9 ± 4.4
M-CNN*	small	83.4	30.3	35.2	13.5	11.6	36.5	22.1	19.8	22.2	5.2	13.7	26.7 ± 1.0
M-CNN*	large	84.6	35.3	44.8	24.7	21.7	44.4	26.3	26.5	27.9	10.0	22.9	33.6 ± 0.2
M-CNN* hard	large	83.6	35.3	38.6	24.0	21.2	39.6	20.2	21.3	19.2	7.9	17.9	29.9 ± 0.7
M-CNN	large	86.3	46.5	43.5	27.6	34.0	47.5	28.7	31.0	30.8	32.4	43.4	41.2 ± 1.3

3.2.3 Evaluation of M-CNN

We start by evaluating the different components of our M-CNN approach and compare to the state-of-the-art EM-Adapt method, see Table 3.1. We train EM-Adapt and M-CNN with the pruned shots from our YouTube-Objects training set in two network settings: large and small field of view (FOV). The large FOV is 224×224, while the small FOV is 128×128. We learn 5 models which vary in the order of the training samples and their variations (cropping, mirroring), and report the mean score and standard deviation.

The small FOV M-CNN without the fine-tuning step achieves an IoU of 26.7%, whereas large FOV gives 33.6% on the PASCAL VOC 2012 validation set. In contrast, EM-Adapt [107] trained [2] on the same dataset performs poorly with large FOV. Furthermore, both the variants of EM-Adapt are lower in performance than our M-CNN. This is because EM-Adapt uses a heuristic (where background and foreground are constrained to a fraction of the image area) to estimate the latent segmentation labels, and fails to leverage the weak supervision in our training dataset effectively. Our observation on this failure of EM-Adapt is further supported by the analysis in [107], which notes that a large FOV network performs poorer than its small FOV counterpart when only a "small amount of supervision is leveraged". The label prediction step (§3.1.2) proposed in our method leverages training data better than EM-Adapt, by optimizing an energy function involving soft motion constraints and network responses. We also evaluated the significance of using motion cues as soft constraints (M-CNN*) instead of introducing them as hard labels (M-CNN* hard), i.e., directly using motion segmentation result as latent labels $\mathbf{y}$. 'M-CNN* hard' achieves 29.9 compared to 33.6 with soft constraints. We then take our best variant (M-CNN with large FOV) and fine-tune it, improving the performance further to 41.2%. In all the remaining experiments, we use the best variants of EM-Adapt and M-CNN.

2. We used the original implementation provided by the authors to train EM-Adapt.

Table 3.2 – Performance of our M-CNN variants on the VOC 2012 validation set is shown as IoU scores. We also compare with the best variants of EM-Adapt [107] trained on YouTube-Objects (YTube), ImageNet-VID (ImNet), VOC, and augmented VOC (VOC aug.) datasets. † denotes the average result of 5 trained models.

Method	Dataset	bkg	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Average
EM-Adapt	YTube	65.7	25.1	20.5	9.3	21.6	23.7	12.4	17.7	14.9	19.5	25.4	23.2†
EM-Adapt	ImNet	66.1	22.8	18.7	16.9	26.7	35.7	22.4	23.6	21.4	28.4	24.3	27.9
EM-Adapt	VOC	75.5	30.5	27.4	24.1	41.8	36.8	25.5	33.3	29.3	40.0	29.7	35.8
EM-Adapt	VOC aug.	77.4	32.1	30.8	26.4	42.6	40.7	32.8	37.8	35.1	45.2	41.1	40.2
M-CNN	YTube	86.3	46.5	43.5	27.6	34.0	47.5	28.7	31.0	30.8	32.4	43.4	41.2†
M-CNN	VOC+YTube	85.4	54.5	40.8	35.5	41.2	47.5	38.3	42.0	41.5	45.0	47.8	47.2†
M-CNN	VOC aug.+YTube	82.5	47.8	35.3	29.6	45.6	54.6	40.3	46.6	44.8	52.2	56.6	48.7
M-CNN	ImNet	85.6	41.4	45.3	23.2	38.6	42.3	36.0	35.1	21.1	15.3	44.8	39.0
M-CNN	VOC+ImNet	85.1	53.3	46.8	32.5	33.9	37.3	40.7	32.3	34.2	40.0	45.0	43.7
M-CNN	VOC aug.+ImNet	83.1	47.6	40.3	26.4	44.1	51.1	41.7	51.0	34.9	44.6	52.7	47.0

3.2.4 Training on weakly-annotated videos & images

We also trained our M-CNN with weakly-annotated videos and images. To this end, we used images from the VOC 2012 training set. We added the 914 images from the VOC 2012 training set containing the 10 classes, and used only their weak annotations, i.e., image-level labels. In this setting, we first trained the network with the pruned video shots from YouTube-Objects, fine-tuned it with a subset of shots (as described in §3.1.3), and then performed a second fine-tuning step with these selected video shots and VOC images. To estimate the latent segmentation labels we use our optimization framework (§3.1.2) when the training sample is from the video dataset and the EM-Adapt label prediction step when it is from the VOC set. We can alternatively use our framework with only the network prediction component for images, but this is not viable when training on classes without video data, i.e., the remaining 10 classes in VOC. As shown in Table 3.2, using image data, with additional object instances, improves the IoU score from 41.2 to 47.2. In comparison, EM-Adapt re-trained for 10 classes on the original VOC 2012 achieves only 35.8. Augmenting the dataset with several additional training images [51], improves it to 40.2, but this remains considerably lower than our result. M-CNN trained with ImageNet-VID achieves 39.0 (ImNet in the table), which is comparable to our result with YouTube-Objects. The performance is significantly lower for the motorbike class (15.3 vs 32.4) owing to the small number of video shots available for training. In this case, we only have 67 shots compared to 272 from YouTube-Objects. Augmenting this dataset with VOC images boosts the performance to 43.7 (VOC+ImNet). Augmenting the training set with additional images (VOC aug.) further increases the performance.

Qualitative results. Fig. 3.5 shows qualitative results of M-CNN (trained on VOC and YouTube-Objects) on a few sample images. These have much more accurate object boundaries than the best variant of EM-Adapt [107], which tends to localize the object well, but produces a 'blob-like' segmentation, see the last three rows in the figure in particular. The first two rows show example images containing multiple object categories. M-CNN recognizes object classes more accurately, e.g., cow in row 4, than EM-Adapt, which confuses cow (shown in green) with horse (shown in magenta). Furthermore, our segmentation results compare favorably with the fully-supervised DeepLab [23] approach (see rows 3-4), highlighting the impact of motion to learn segmentation. There is scope for further improvement, e.g., overcoming the confusion between similar classes in close proximity to each other, as in the challenging case in row 2 for cat vs dog.

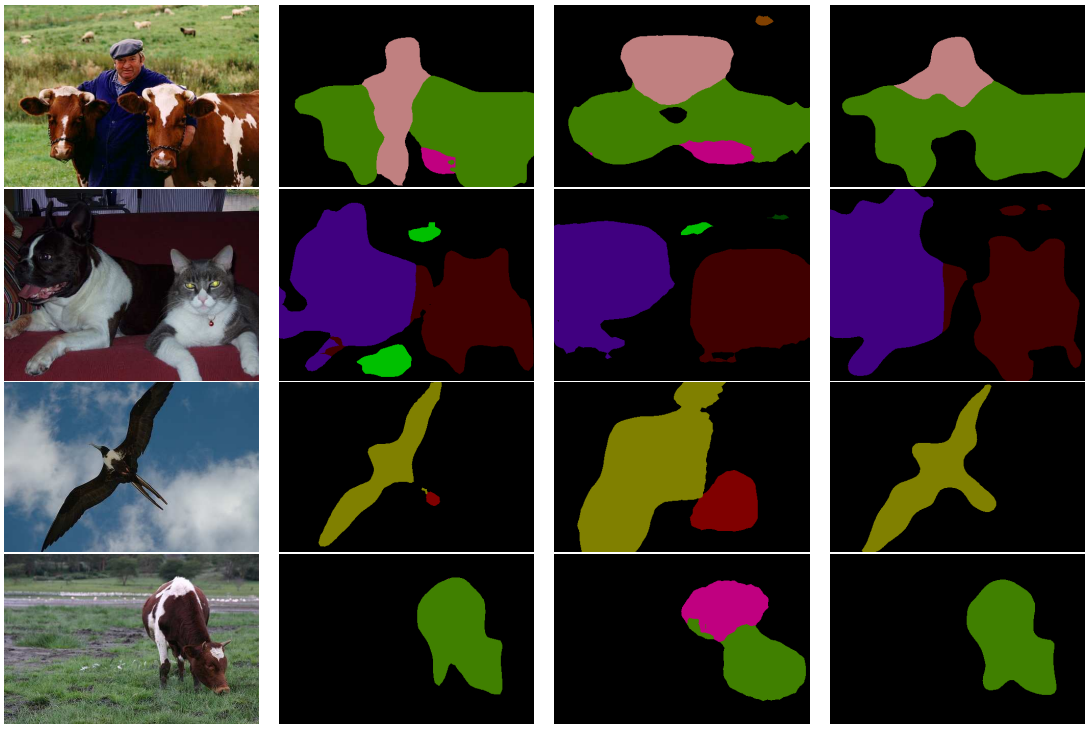


Figure 3.5 – Sample results on the VOC 2012 validation set. Results of fully-supervised DeepLab[23], weakly-supervised EM-Adapt [107] trained on augmented VOC, and our weakly-supervised M-CNN trained on VOC+YouTube-Objects are shown in 2nd, 3rd and 4th columns respectively.

Comparison to the state of the art. Table 3.3 shows evaluation on the VOC 2012 test set, with our M-CNN trained on 20 classes using image and video data for 10 classes and image data only for the other 10. We performed this by uploading our segmentation results to the evaluation server, as ground truth is not publicly available for the test set. We compare with several state-of-the-art methods with scores taken directly from the publications, except [107] without the post-processing CRF step. This result, shown as '[107]' in the table, is with a model we trained on the VOC augmented dataset. We train M-CNN on all the 20 VOC classes with the model trained (and fine-tuned) on YouTube-Objects and perform a second fine-tuning step together with videos from YouTube-Objects and images from VOC. This achieves 39.8 mean IoU over all the 20 classes, and 49.6 on the 10 classes with video data. This result is significantly better than methods using only weak labels, which achieve 25.7 [110], 35.6 [109] and

Table 3.3 – Evaluation on the VOC 2012 test set shown as IoU scores.

Method	Training data	# train. samples	Average	Avg. 10-cls
Strong/Full supervision				
[114] + bb	VOC+ImNet	∼762,500	37.0	43.8
[114] + seg	VOC+ImNet	∼761,500	40.6	48.0
[95] (full)	VOC aug.	9,610	62.2	71.3
[107] + seg	VOC aug.	12,031	69.0	78.2
[100] (full)	VOC aug.	10,582	69.6	79.3
[160] (full)	VOC aug.	11,685	72.0	80.8
[160] (full)	VOC aug.+COCO	77,784	74.7	82.9
[23] (full)	VOC aug.	12,031	71.6	79.1
Weak supervision with additional info.				
[114] + sp	ImNet	∼760,000	35.8	42.3
[109] + sz	VOC aug.	10,582	43.3	48.9
[109] + sz + CRF	VOC aug.	10,582	45.1	51.2
[107] + CRF	VOC aug.	12,031	39.6	45.2
Weak supervision				
[110]	VOC aug.	12,031	25.7	-
[109]	VOC aug.	10,582	35.6	39.5
[107]	VOC aug.	12,031	35.2	40.3
Ours	VOC+YTube	3,139	**39.8**	**49.6**
Ours	VOC+ImNet	3,155	36.9	48.0

35.2 [107]. The improvement shown by our M-CNN is more prominent when we consider the average over 10 classes where we use soft motion segmentation cues, and the background, with nearly 10% and 9% boost over [109] and [107] respectively. We also show the evaluation of the model trained on ImageNet-VID in the table.

A few methods have used additional information in the training process, such as the size of objects (+ sz in the table), superpixel segmentation (+ sp), or post-processing steps, e.g., introducing a CRF with pairwise terms learned from fully-annotated data (+ CRF), or even strong or full supervision, such as bounding box (+ bb) or pixel-level segmentation (+ seg) annotations. Even though our pure weakly-supervised method is not directly comparable to these approaches, we have included these results in the table for completeness. Nevertheless, M-CNN outperforms some of these methods [107, 114], due to our effective learning scheme. Also from

Table 3.4 – Co-localization performance of M-CNN on the YouTube-Objects dataset. We report per class and average CorLoc scores, and compare with state-of-the-art unsupervised and weakly-supervised methods.

Method	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Average
Unsupervised											
[18]	53.9	19.6	38.2	37.8	32.2	21.8	27.0	34.7	45.4	37.5	34.8
[108]	65.4	67.3	38.9	65.2	46.3	40.2	65.3	48.4	39.0	25.0	50.1
[79]	55.2	58.7	53.6	72.3	33.1	58.3	52.5	50.8	45.0	19.8	49.9
Weakly-supervised											
[116]	51.7	17.5	34.4	34.7	22.3	17.9	13.5	26.7	41.2	25.0	28.5
[67]	25.1	31.2	27.8	38.5	41.2	28.4	33.9	35.6	23.1	25.0	31.0
[79]	56.5	66.4	58.0	76.8	39.9	69.3	50.4	56.3	53.0	31.0	55.7
M-CNN	76.1	57.7	77.7	68.8	71.6	75.6	87.9	71.9	80.0	52.6	**7**2.0

Table 3.3, the number of training samples used for M-CNN (number of videos shots + number of VOC training images) is significantly lower than those for all the other methods.

3.2.5 Co-localization

We perform co-localization in the standard setting, where videos contain a common object. Here, we use our M-CNN trained on the YouTube-Objects dataset with 10 categories. We evaluate it on all the frames in YouTube-Objects to obtain prediction scores $\mathbf{p}_i$ for each pixel i. With these scores, we compute a foreground GMM by considering pixels with high predictions for the object category as foreground. A background GMM is also computed in a similar fashion. These form the unary term ψ_i^m in the energy function ((3.4)). We then minimize this function with graph cut based inference to compute the binary (object vs background) segmentation labels. Since we estimate segmentations for all the video frames, we do this at the superpixel level [5] to reduce computation cost. We then extract the bounding box enclosing the largest connected component in each frame, and evaluate them following [116]. Quantitative results with this are summarized as CorLoc scores in Table 3.4. We observe that our result outperforms previous state of the art [79] by over 16%. Performing this experiment with ImageNet-VID data we obtain 42.1 on average, in comparison to 37.9 of [108]. ImageNet-VID being a more challenging dataset than YouTube-Objects results in a lower performance for both these methods.

We qualitatively demonstrate the performance of our method on the YouTube-Objects dataset in Figure 3.6. Our method produces stable results on a variety of categories (third column in the figure). The performance of the motion segmentation method [108] is also shown for comparison. It is

limited by the quality of optical flow and the heuristics used to distinguish foreground from background motion. As a result, it often fails, see second column in the figure.

3.3 Summary

This chapter introduces a novel weakly-supervised learning approach for semantic segmentation, which uses only class labels assigned to videos. It integrates motion cues computed from video as soft constraints into a fully convolutional neural network. Experimental results show that our soft motion constraints can handle noisy motion information and improve significantly over the heuristic size constraints used by state-of-the-art approaches for weakly-supervised semantic segmentation, i.e., EM-Adapt [107]. We show that our approach outperforms previous state of the art [107, 109] on the PASCAL VOC 2012 image segmentation dataset, thereby overcoming domain-shift issues typically seen when training on video and testing on images. Furthermore, our weakly-supervised method shows excellent results for video co-localization and improves over several methods [67, 79, 108]. Hong et al. [57], have recently extended our method by introducing a problem-specific network architecture and incorporating class activation maps of Zhou et al. [162] in the label inference objective. They then trained this model on a large dataset of videos collected from YouTube, achieving in a significant performance improvement. We discuss their approach in more detail in Chapter 6.

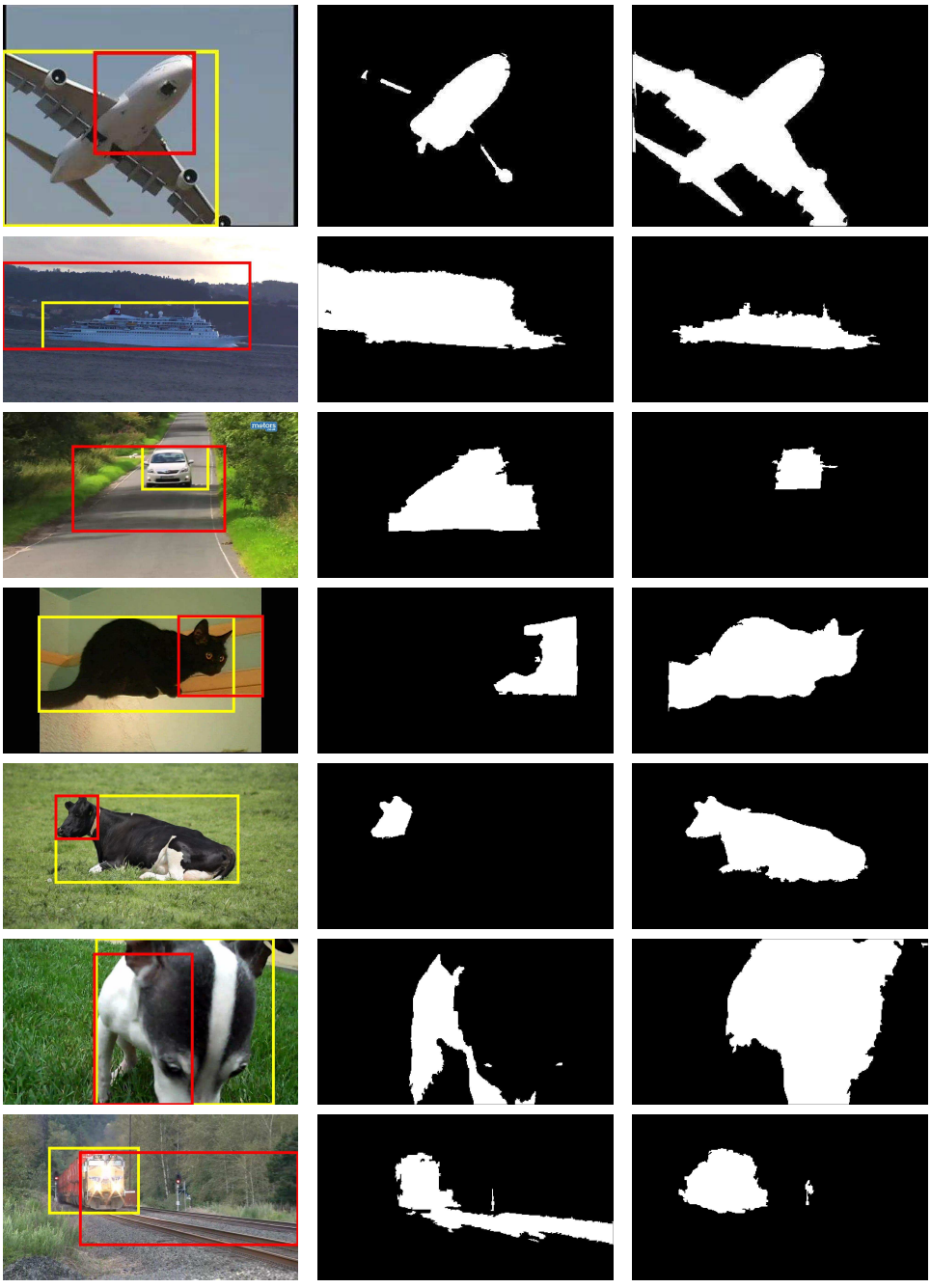


Figure 3.6 – Sample co-localization results on the YouTube-Objects dataset. In the first column the estimated bounding boxes are shown, where yellow corresponds to our result, and red to that of [108]. Segmentations corresponding to [108] and our method are shown in columns 2 and 3 respectively.

Part II

Video Object Segmentation

Video object segmentation is the task of extracting spatio-temporal regions that correspond to object(s) moving in at least one frame in the video sequence. The top-performing methods for this problem [35, 108] continue to rely on hand-crafted features and do not leverage a learned video representation, despite the impressive results achieved by convolutional neural networks (CNNs) for other vision tasks, e.g., image segmentation [115], object detection [118]. Very recently, there have been attempts to build CNNs for video object segmentation [20, 63, 72]. They are some of the the first approaches to use deep learning for video segmentation, but suffer from various drawbacks. For example, [20, 72] rely on a manually-segmented subset of frames (typically the first frame of the video sequence) to guide the segmentation pipeline. The approach of [63] does not require manual annotations, but remains frame-based, failing to exploit temporal consistency in videos. Furthermore, none of these methods has a mechanism to *memorize* relevant features of objects in a scene.

In this part we propose a two-stream network with an explicit memory module for video object segmentation. The memory module is a convolutional gated recurrent unit (GRU) that encodes the spatio-temporal evolution of object(s) in the input video sequence. This spatio-temporal representation used in the memory module is extracted from two streams—the appearance stream which describes static features of objects in the video, and the temporal stream which captures the independent object motion.

The rest of this part of the thesis is organized as follows: in Chapter 4 we review the related work on motion and video object segmentation. In Chapter 5 we introduce our learning-based methods for motion and video object segmentation and present an experimental evaluation on several benchmark dataset. In addition, we provide an extensive ablation study to analyze different design choices of the method.

Chapter 4

Related Work

Contents

The contributions described in the second part of the thesis are related to three topics: motion estimation, video object segmentation, and recurrent neural networks. In this section we review the work relevant to these topics. The key subproblem in video object segmentation is segmenting independent motion between a pair of frames, thus we start by describing motion segmentation methods in the Section 4.1. We then continue by introducing video object segmentation and reviewing the methods that addressed this problem in the past in the Section 4.2. A convolutional recurrent unit is an important component of our approach for video segmentation, thus we conclude with a review of related work on recurrent neural networks in Section 4.3.

4.1 Motion estimation

Consider a pair of frames from the FlyingThings3D dataset of [97] shown in Figure 4.1. It depicts a scene generated synthetically, involving a moving camera (can be easily observed by comparing the top left corners of the images (a) and (b)), with three large objects in the centre of the frame and

(a) (b) (c) (d)

Figure 4.1 – (a,b) Two example frames from a sequence in the FlyingThings3D dataset [97]. The camera is in motion in this scene, along with four independently moving objects. (c) Ground-truth optical flow of (a), which illustrates motion of both foreground objects and background with respect to the next frame (b). (d) Ground-truth segmentation of moving objects in this scene.

several objects in the background. The problem of motion segmentation can then be defined as segmenting the objects that move independently from the camera (see the ground truth segmentation (d)). Observing the corresponding optical flow field (image (c) in the figure), one can see that this problem is not trivial, since flow is not constant on the static objects in the background. The reason for this is that, in presence of a camera motion, 2D flow direction and magnitude depend on the distance from the camera to the object [59]. In particular, if we denote the translational component of the camera motion in the 3D world as $\mathbf{T} = (U, V, W)$ and the camera's angular velocity as $\omega = (A, B, C)$, then the optical flow (u, v) at a static point (x, y) can be computed as follows:

$$\begin{aligned} u &= (-\frac{U}{Z} - B + Cy) - x(-\frac{W}{Z} - Ay + Bx), \\ v &= (-\frac{V}{Z} - Cx + A) - y(-\frac{W}{Z} - Ay + Bx). \end{aligned} \tag{4.1}$$

Notice that both components of the flow vector depend on Z - the distance from the camera to the point (x, y) in the scene, which is unknown in the general case. To resolve this ambiguity, some approaches rely on geometric properties of the optical flow, whereas others propose various heuristics to identify independent motion. Next, we separately describe the methods which fall into both categories.

4.1.1 Geometry-based methods

Early attempts for estimating independent motion were primarily geometry-based, such as [139], where the author suggests to estimate the fundamental matrices $\mathbf{F}_j$, containing all the information about the motion of a pixel in

3D, for all the motion models in a scene (that is, for the camera motion and for each independent object motion). Consider a pixel $\mathbf{x_i} = (x_i, y_i, z_i)$ and a pixel $\mathbf{x}_i^{'} = (x_i^{'}, y_i^{'}, z_i^{'})$, which represents $\mathbf{x_i}$ after undergoing a motion j, the two can be related with the following equation:

$$\mathbf{x}_i^{'}\mathbf{F}_j\mathbf{x}_i = 0. \tag{4.2}$$

The matrix F_j can thus be estimated by solving for this equation given a set of point correspondences between two frames. Motion segmentation can then be obtained by selecting the model corresponding to the camera and labeling all the pixels that do not fit to the camera model as moving. Potential set of motions is identified with RANSAC [39], a method consisting of randomly sampling pixels in an image, estimating the motion model for these pixels and evaluating the consistency of the sampled data points with the estimated model. After several iterations the model with the highest consistency score is selected. Pixels having a high score according to the selected model are then removed from the set and the process is repeated until all the pixels are assigned to a model. Notice that this algorithm is akin to k-means clustering, when the number of clusters is not known in advance. Despite the appeal of operating directly in the 3D space, estimating a 3D motion model from a single-camera video is often ambiguous. Thus, the most successful approaches rely on other principles.

Some of these approaches still utilize geometry, but operate in the 2D space instead. For instance, in [101] the authors consider the case of translational camera motion only and remark that in this scenario the direction of optical flow vectors (unlike their magnitude) does not depend on depth. Let's denote by t_θ the direction of the flow at a point (x, y). It can be shown that, in the absence of camera rotation, the flow direction can be computed as follows

$$t_\theta = \arctan(Wy - Vf, Wx - Uf), \tag{4.3}$$

where f represents the focal length of the camera. Thus, t_θ does not depend on Z and is constant for static pixels. The method proceeds similarly to [139] by estimating orientation-based motion models for background and object motions in a probabilistic framework. Per-pixel labels are then assigned based on consistency with one of the estimated models. This approach however, is prone to errors in case of camera rotation, and consistent object and camera motion orientation. An extension of the method is introduced in [13], where the estimated camera rotation component $\hat{O}_R$ is subtracted from the optical flow field O in every frame. The flow orientation-based motion models are then computed for the estimated

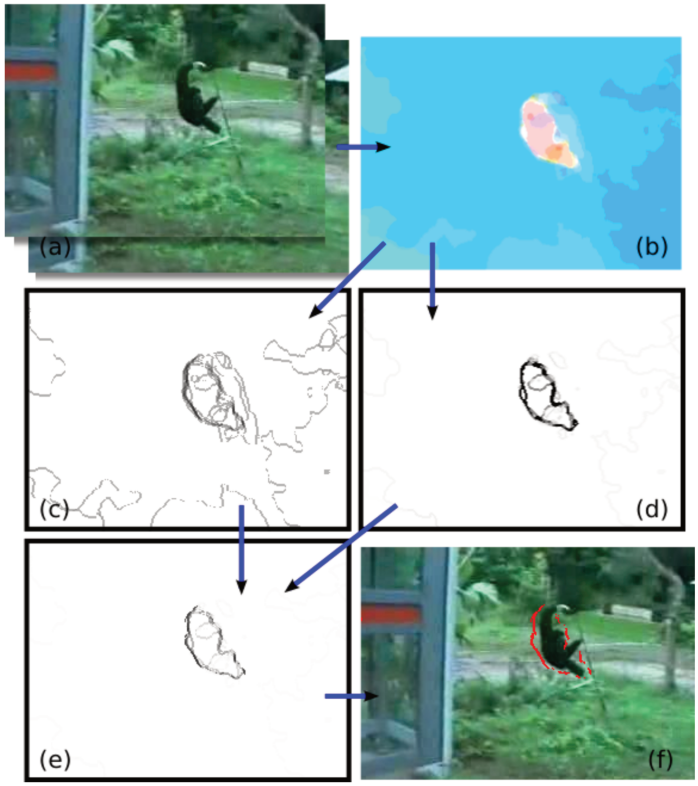


Figure 4.2 – Visualization of the approach of [108] for a frame pair (a). Optical flow (b) is first extracted and the motion boundaries based on flow gradients (c) and orientation differences (d) are computed and combined in (e). (f) Final motion boundaries thresholded and overlaid on the image. (Image courtesy of Papazoglou et al. [108])

translation component $\hat{O}_T$ only:

$$\hat{O}_T = O - \hat{O}_R, \tag{4.4}$$

thus the equation (4.3) holds. In addition, the initial estimates of foreground and background motion models are updated over time, with optical flow orientations of the new frames. This method, however, still relies on RANSAC for initialization, and lacks a robust learning framework. It then fails in challenging scenarios, as was demonstrate in [137].

4.1.2 Heuristic-based methods

Recent methods have relied on other cues to estimate moving object regions. For example, [108] first extract motion boundaries by measuring

changes in optical flow field, and then use them to estimate moving regions (see Figure 4.2). To increase the robustness of the approach, they combine two complementary motion boundary estimation techniques. The first one (image (c) in Figure 4.2) is based on flow gradient magnitude $\| \nabla \vec{f}_p \|$ for a pixel p:

$$b_p^m = 1 - \exp(-\lambda^m \| \nabla \vec{f}_p \|), \tag{4.5}$$

where $b_p^m \in [0, 1]$ measures the strength of the motion boundary and λ^m is a parameter controlling steepness of the function. This measure has high precision for rapidly moving pixels, where b_p^m is close to 1, but is unreliable for pixels with b_p^m values around 0.5. It is thus augmented by the second measure, based on the difference in direction between the motion of a pixel p and its neighbors $\mathcal{N}$ (image (d) Figure 4.2):

$$b_p^\theta = 1 - \exp(-\lambda^\theta \max_{q \in \mathcal{N}}(\sigma\theta_{p,q}^2)), \tag{4.6}$$

where $\sigma\theta_{p,q}^2$ denotes the angle between $\vec{f}_p$ and $\vec{f}_q$. This measure is complementary to b_p^m in that it produces confident predictions in regions with modest flow velocity, where the other measure is uncertain. For static pixels, however, b_p^θ scores are not well defined, but b_p^m is able to suppress possible false positives. The result of combining the two edge maps is shown in the image (e) in Figure 4.2.

To obtain an estimate of independently moving regions from the combined edge map the authors observe that objects in motion are often represented by motion boundaries that form closed contours and can be efficiently identified. They design an algorithm which finds regions lying inside closed motion contours based on the point-in-polygon problem in computational geometry [60]. It states that a ray starting from a point inside a polygon intersects its boundary an odd number of time, whereas a ray starting outside a polygon has an even number of intersection. Since motion boundaries are often incomplete, they suggest to shoot several rays from a point inside a region, evenly covering the image, to estimate its motion likelihood.

A different heuristic for identifying independent object motion based on surface occlusion was proposed in [12] and [134]. Occlusion occurs when either objects or the viewer move, causing parts of the scene to become hidden and others revealed. To identify such regions, optical flow algorithms are used. In particular, under the standard assumption of the most of the flow algorithms (i.e., constant illumination) a point x in the frame I_t is related to the frame I_{t+1} through the following equation:

$$I_t(x) = I_{t+1}(w_t^{t+1}(x)) + n_t(x), \tag{4.7}$$

Figure 4.3 – Sample of the segmentation results produced by the method of [134] on FBMS. Grey, blue, yellow and red regions correspond to different scene layers inferred by the method. (Image courtesy of Taylor et al. [134])

where w_t^{t+1} is the deformation filed that warps the frame I_t into I_{t+1}, which is typically represented by the optical flow field O_t^{t+1}, with $w_t^{t+1}(x) = x + O_t^{t+1}(x)$. The remaining component $n_t(x)$ models the phenomena that violate the assumptions. Occluded regions can thus be easily identified as the ones that yield a large residual n_t after optical flow estimation.

Occlusion relations provides information about the local order between surfaces in the scene, since the occluder surface is always closer to the viewer than the occluded one. This local information can be aggregated throughout the video to obtain a complete layered segmentation of every frame. Connected components within each layer then represent distinct objects in the scene. Notice that the objects segmented in this way include, but are not limited to the independently moving ones, since motion of the viewer can reveal depth ordering relation between two static objects as well, as demonstrated in the first and third images in Figure 4.3. The last image in this figure also shows that objects can be over-segmented when they exhibit non-rigid motion. In practice, the results of methods capitalizing on occlusion are often competitive with [108], but both are limited by their heuristic initialization.

While we also set out with the goal of finding objects in motion, our solution to this problem is a novel learning-based method. It is related to the approach of [108] in that it seeks to capture patterns in the flow filed that correspond to independent motion, but instead of defining these patterns manually, we propose to learn them from a large dataset of synthetic videos. In concurrent work [63] presented a deep network to segment independent motion in the flow field as well. While their approach is related to ours, they use frame pairs from real videos, in contrast to synthetic data in our case. Consequently, their work relies on estimated optical flow in training. Since obtaining accurate ground truth moving object segmentation labels is prohibitively expensive for a large dataset, they rely on an automatic, heuristic-based label estimation approach, which results in noisy annotations. We explore the pros and cons of using this realistic but noisy dataset

for training our motion segmentation model in Section 5.4.3.

4.2 Video object segmentation

The task of segmenting objects in video is to associate pixels belonging to a class spatio-temporally, in other words, extract segments that respect object boundaries, as well as associate object pixels temporally whenever they appear in the video. This can be accomplished either by propagating manual segment labels in one or more frames to the rest of the video sequence [119], or by capitalizing on the low-level video cues, such as independent object motion. The former setting is usually referred to as semi-supervised video object segmentation, and the later as unsupervised. We review state-of-the-art methods in both setting in Sections 4.2.1 and 4.2.2 respectively.

4.2.1 Semi-supervised approaches

Many approaches in this group rely heavily on the temporal consistency property of the video: the fact that video content usually does not change rapidly between two consecutive frames. Each pixel can then be tracked with optical flow, resulting in a spatio-temporal graph that defines constraints on the pixel label assignment. The actual assignment can be obtained by propagating manual segmentation labels through the graph. To increase the robustness, as well as efficiency of this process, the graph is often built over superpixels or generic object parts, like in the method of [148]. An interesting extension is proposed in [140], where labeling is interleaved with optical flow estimation (see Figure 4.4). Flow is used to obtain temporal edges between pixels, shown in red, whereas spatial edges are shown in black. This information is aggregated on the level of superpixels shown in turquoise, and used to compute the object mask $\mathcal{M}_t$, by label propagation. The mask is then used to re-estimate optical flow in object and background regions separately, and flow is, in turn, used to recompute temporal edges in the graph. The intuition behind this approach is that modeling the two tasks jointly can improve performance of the both of them.

Methods built around local label propagation are limited by the short range of the interaction they consider, however. To address this issue, several methods (see [113] and [157]) propose global label propagation strategies. They include long-range interaction into the neighborhood graph, connecting frames from different parts of a video, which allows to better

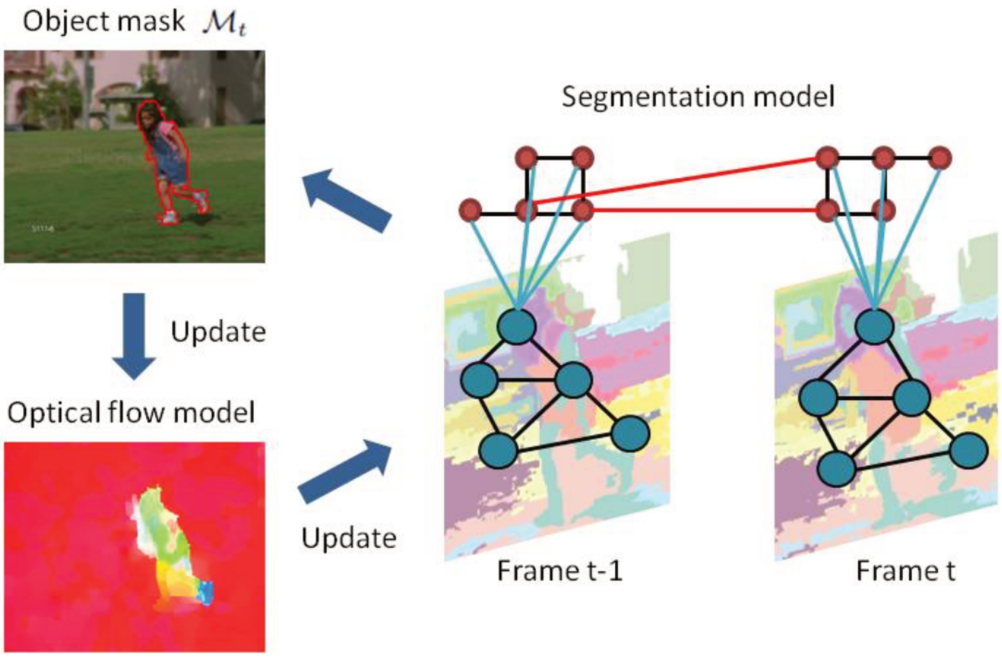


Figure 4.4 – Illustration of the approach of [140]. Red circles denote pixels, which belong to the superpixels marked by the turquoise circels. The black and red lines denote spatial and temporal relationship, respectively. The relationship between the pixels and the superpixel are denoted by the turquoise lines. (Image courtesy of Tsai et al. [140])

handle object appearance variation and occlusion. Inference in a global video graph built on the level of pixels or fine superpixels is prohibitively expensive, thus the methods in this family often utilize object proposal as basic units. This allows to decrease the cost of the graph inference step, but limits the performance of these methods by the performance of the proposal generation approach employed.

Very recently, CNN-based methods for video object segmentation were introduced by [20] and [72]. Starting with CNNs pre-trained for image segmentation, these methods find objects in video by fine-tuning a network on a manually annotated frame in the sequence and applying it to all the other video frames. This simple strategy leads to surprisingly good results, outperforming all the label propagation approaches described above. An obvious limitation of these CNN-based methods is that they, being trained on a single view of an object, fail to segment it in case of a large appearance variation. An extension proposed by [144] suggests to segment a video frame by frame, starting from the annotated one, finetuning the network

on the predicted segmentation in every step. Due to the temporal consistency property one can assume that object appearance will not change significantly between two consecutive frames, thus the prediction of the network trained on the previous frame will still be reliable, allowing to avoid the concept drift problem. This robustness comes at a high computational cost, however.

4.2.2 Unsupervised approaches

This work focuses on the problem of unsupervised video object segmentation: segmenting spatio-temporally consistent regions in videos without any manually-marked regions. Several methods in this paradigm (see [17, 49, 71, 85, 153]) are built on top of spatio-temporal graphs, similarly to the classical semi-supervised approaches described in the previous section. In the absence of manually annotated regions to propagate the labels from, however, they resort to unsupervised graph partitioning. For instance, in [42] and [71] the authors suggest to partition a set of vertices $\mathcal{V}$, corresponding to superpixels in a video, into N disjoint subsets $\{S_1, S_2, ..., S_N\}$ by optimizing for the normalized cut objective:

$$NCut(S_1, S_2, ..., S_N) = \sum_{k=1}^{N} \frac{cut(S_k, \mathcal{V} \setminus S_k)}{vol(S_k)}, \tag{4.8}$$

where $cut(S_k, \mathcal{V} \setminus S_k)$ denotes the cost of separating the nodes in S_k from the rest of the graph and $vol(S_k)$ measures the consistency of S_k. This objectives encourages splitting the video into a set of spatio-temporal segments that are consistent within each other and clearly separated from each other. The particular result is fully determined by the definition of the weights of the graph edges. In practice, these approaches utilize various heuristic combinations of appearance, location and motion features to compute the superpixel similarities. Several examples of segmentations obtained by the method of [71] are shown in Figure 4.5. They demonstrate that approaches relying purely on unsupervised graph partitioning result in over-segmentation of videos. For instance, in the first example both cars in the foreground are separated into several independent segments and in all cases background region is oversegmented. While such an oversegmentation can be a useful intermediate step for some recognition tasks in video, it has no notion of objects. Indeed, most of the extracted segments in this case do not directly correspond to objects, making it non-trivial to obtain video object segmentation from this intermediate result.

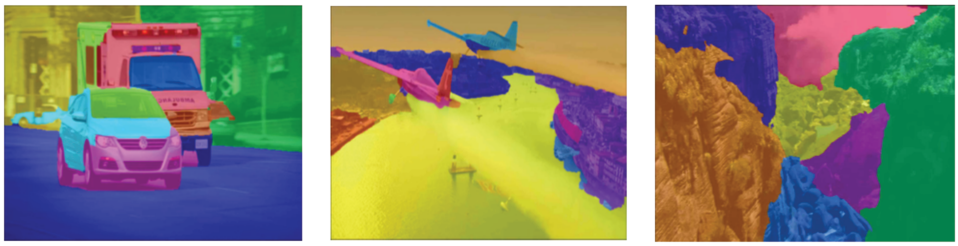


Figure 4.5 – Example segmentations obtained by the method of [71] on the videos from VSB100 [43]. Each color corresponds to a spatio-temporal region extracted by the algorithm. (Image courtesy of Khoreva et al. [71])

Capturing the notion of an object in a video in a purely unsupervised way is one of the key problems in computer vision. The seminal work of [18] addresses it in a fundamental way, by exploiting the well known principle of "common fate" from Gestalt psychology [73]: surfaces that move together belong together. Their method relies on long term pixel trajectories of [133]. In particular, they track each pixel individually throughout the video with optical flow. They then define a distance between two trajectories A and B as a maximal difference in motion between them over time:

$$d(A, B) = \max_t d_t(A, B), \tag{4.9}$$

where d_t is the distance between two trajectories at a time t:

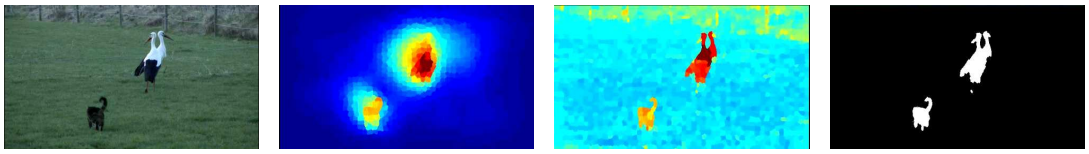
$$d_t(A, B) = d_{sp}(A, B)\frac{(u_t^A - u_t^B)^2 + (v_t^A - v_t^B)^2}{5\sigma_t^2}. \tag{4.10}$$

This formulation is based on the difference in the optical flow, as well as the spatial distance d_{sp} between the trajectories in the given frame, and σ is a normalization term. The intuition behind it is that only trajectories that move similarly throughout the whole video are assigned a small distance. Final segmentation is obtained with graph partitioning, where trajectories that overlap spatio-temporally are connected with edges, similarly to the methods described above. In contrast, since the clustering is performed on the level of long term pixel trajectories, not superpixels, it produces more coherent regions, as shown in Figure 4.6. This method, however, assumes homogeneity of motion over the entire object, which is invalid for non-rigid objects, like the person in the example. His left hand moves distinctly from the rest of the body and is thus assigned a different label. This demonstrates that performing graph partitioning on the trajectory level ameliorates but does not completely solve the oversegmentation problem.

Figure 4.6 – A sample result produced by the method of [18] on the FBMS dataset. Pixel of the same colour represent trajectories that were grouped together by the method. (Image courtesy of Brox and Malik [18])

Another issue with the approach of [18] is that it does not produce dense segmentations. Pixels can not be reliably tracked on homogeneous regions (see the person's vest or the wall in the background in Figure 4.6) and thus the method resorts to ignoring them. [103] addressed this limitation by proposing to propagate trajectory labels to the unlabeled pixels on the frame-level by utilizing appearance. Their formulation ensures that trajectory labels are preserved and consistently propagated to homogeneous regions. Other extensions include incorporating higher-order potentials in the trajectory similarity graph in [104] and employing a better graph partitioning approach in [68].

The methods that casts video object segmentation as a foreground-background classification task take a radical approach to the oversegmentation issue: instead of assigning a distinct label to each independently moving region, they propose to limit the number of labels to two. All the pixels that move independently from the camera are labeled as foreground and the rest of the pixels as background (see Figure 4.7). In particular, these methods usually start with one of the motion segmentation strategies described in the Section 4.1 to get an initial estimate of the moving and static regions in every frame (shown in the second image in Figure 4.7). They then learn appearance models for foreground and background regions, usually realized as Gaussian Mixture Models over RGB colour values, and use them to score all the regions in a video, providing a complementary cue to the initial motion estimates (third image in Figure 4.7). These motion and appearance cues are often integrated with other cues as well, e.g., saliency maps [146], pairwise constraints [108, 156], long-range interactions between



Frame Motion estimate Appearance model Final segmentation

Figure 4.7 – Visualization of the approach of [108]. Shown from left to right: an example video frame from the FBMS dataset; corresponding pixel-level estimate of independent motion; combined prediction of the foreground and background appearance models learned from the motion estimates; final segmentation obtained by combining appearance and motion cues in a spatio-temporal graph-based framework.

distinct parts of the video [35], or object shape estimates [83], to compute the final object segmentation. For instance, [108] propose to optimize an energy function over the possible binary label assignments $\mathcal{L}$:

$$E(\mathcal{L}) = \sum_{i,t} A^t(l_i^t) + \sum_{t,i} L(l_i^t) + \sum_{(i,j,t)\in\mathcal{E}_s} V(l_i^t, l_j^t) + \sum_{(i,j,t)\in\mathcal{E}_t} W(l_i^t, l_j^{t+1}) \quad (4.11)$$

with the unary terms A^t and L corresponding to the appearance and motion cues respectively and pairwise terms V and W encouraging spatial and temporal smoothness of the segmentation.

While our proposed method is similar in spirit to this class of approaches, in terms of formulating segmentation as a classification problem, we differ from previous work significantly. We propose an integrated approach to learn appearance and motion features, and update them with a memory module, in contrast to estimating an initial region heuristically and then propagating it over time. Our robust model outperforms all the top ones from this class [35, 83, 108, 134, 146], as shown in Section 5.4.5. Like us [63] augment their motion segmentation network with an appearance model and learn the parameters of a layer to combine the predictions of the two. Their model, however, does not feature a memory module, and also remains frame-based. Thus, it can not exploit the temporal consistency in video.

Our memory module is realized as a convolutional gated recurrent unit (ConvGRU) - the state-of-the-art recurrent neural network (RNN) architecture for modeling spatio-temoporal data. In the next section we briefly introduce RNNs and review the corresponding literature.

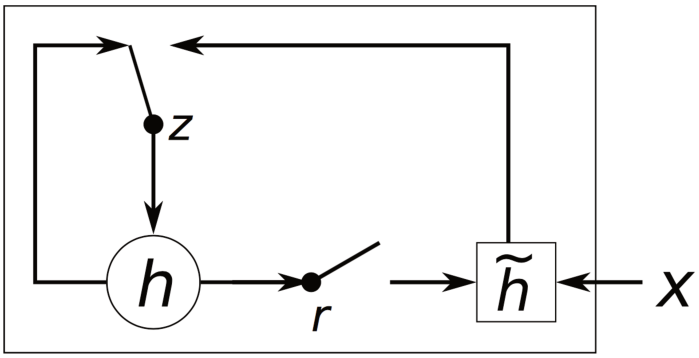


Figure 4.8 – Outline of the GRU architecture of [26]. (Image courtesy of Cho et al. [26])

4.3 Recurrent neural networks (RNNs)

Traditional neural networks, i.e., multi-layer perceptrons [40], or CNNs [81] were designed for recognizing patterns in static inputs, thus they treat each input x_t as independent of the previous ones $\{x_1, x_2, ..., x_{t-1}\}$. In many domains, however, like speech recognition or natural language processing, this assumption does not hold. Data in these domains is represented by sequences of inputs and, to recognize corresponding patterns, all the inputs together, as well as their order, have to be taken into account. Motivated by this setting, RNNs were introduced by [58] as a natural extension of multi-layer perceptrons. The main component of an RNN is an internal state h_t, which allows to accumulate information over time. The state in classical RNNs is updated with a weighted combination of the input x_t and the previous state h_{t-1}, where the weights w_h and w_x are learned from training data for the task at hand:

$$h_t = \tanh(w_h h_{t-1} + w_x x_t). \tag{4.12}$$

Prediction at a time t is then obtained from the state h_t:

$$o_t = softmax(w_o h_t), \tag{4.13}$$

that is from the accumulated representation of all the previous inputs in the sequence. A major problem with learning these models, however, is that of the gradient vanishing exponentially quickly with the length of the training sequence [55]. This issue was later addressed in the LSTM and GRU architectures.

Long short-term memory (LSTM) [56] and gated recurrent unit (GRU) [26] architectures are improved variants of RNN. They introduce gates with learnable parameters, to update the internal state selectively, and can propagate gradients further through time. In particular, GRU augments RNN with two gates: the reset gate r and the update gate z, that control the extent to which previous state h_{t-1} influences the new state h_t (see Figure 5.10). State update equation (4.12) is then split into two steps. First, the candidate state $\tilde{h}_t$ is computed using the reset gate to selectively zero out h_{t-1}:

$$\tilde{h}_t = \tanh(w_{\tilde{h}} r_t \odot h_{t-1} + w_x x_t). \tag{4.14}$$

The new state at time t is then computed as a weighted sum of the previous state and the candidate state, where the weight is defined by the update gate:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t. \tag{4.15}$$

The gates themselves are computed from the previous state and the the current input:

$$\begin{aligned} r_t &= \sigma(w_{hr} h_{t-1} + w_{xr} x_t), \\ z_t &= \sigma(w_{hz} h_{t-1} + w_{xz} x_t), \end{aligned} \tag{4.16}$$

where w_{hr}, w_{xr}, w_{hz} and w_{xz} are parameters that are learned as well. The main goal of this architecture is to break the symmetry in the computation of RNN activations. Instead of multiplying the input and the previous state by fixed matrices w_x, w_h from equation (4.12) at every time step, adaptive multipliers in the from of the gates are introduced. This partially mitigates the vanishing/exploding gradient issue and allows the model to learn to dynamically adapt its state to the changing input sequence.

Often exploiting information from the future inputs $\{x_{t+1}, x_{t+2}, ..., x_T\}$ can be helpful for making a prediction for the current input x_t. For instance, [48] introduce an LSTM-based model for speech recognition that takes the whole whole sequence of phonemes into account to classify each of them. They demonstrate that future information can help to resolve ambiguity in cases when the speaker's pronunciation is unclear. Recurrent models that process an input sequence in backward as well as in forward direction are called bidirectional. In particular, let's denote the function representing the GRU and depicted in Figure 5.10 as $\vec{g}$:

$$h_t = \vec{g}(h_{t-1}, x_t). \tag{4.17}$$

Bidirectional GRU then introduces another copy of a GRU $\overleftarrow{g}$ applied in the backward direction, with its own state h':

$$h'_t = \overleftarrow{g}(h'_{t+1}, x_t). \tag{4.18}$$

Prediction at a time t is then obtained from a combined state of the two models h_t and h'_t by either summing or concatenating them.

Recurrent models, originally used for text and speech recognition, e.g., [47, 98], are becoming increasingly popular for visual data. Initial work on vision tasks, such as image captioning [31], future frame prediction [131] and action recognition [102], has represented the internal state of the recurrent models as a 1D vector - without encoding any spatial information. LSTM and GRU architectures have been extended to address this issue with the introduction of ConvLSTM [38, 111, 126] and ConvGRU [9] respectively. In these convolutional recurrent models the state and the gates are 3D tensors and the weight vectors are replaced by 2D convolutions. These models have only recently been applied to vision tasks, such as video frame prediction [38, 111, 126], action recognition and video captioning [9].

In our work, we employ a visual memory module based on a convolutional GRU (ConvGRU) into our approach, and show that it is an effective way to encode the spatio-temporal evolution of objects in video for segmentation. Further, to fully benefit from all the frames in a video sequence, we apply the recurrent model bidirectionally. This makes our memory module a *bidirectional convolutional recurrent model*.

Chapter 5

Learning to Segment Moving Objects

Contents

Our model for video object segmentation is a two-stream network with an explicit memory module (see Figure 5.1). The two streams are the appearance stream (shown in green) that describes static features of objects in the video, and the temporal stream (show in yellow) that captures the independent object motion. The memory module is a convolutional gated

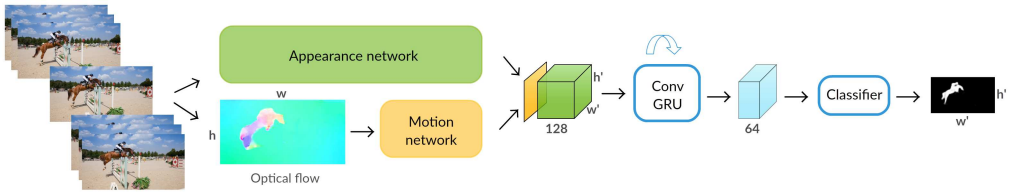


Figure 5.1 – Overview of our segmentation approach. Each video frame is processed by the appearance (green) and the motion (yellow) networks to produce an intermediate two-stream representation. The ConvGRU module combines this with the learned visual memory to compute the final segmentation result. The width (w') and height (h') of the feature map and the output are w/8 and h/8 respectively.

recurrent unit (ConvGRU) trained to aggregate the spatio-temporal representation of the moving object(s) in the video.

The temporal stream separates independent object and camera motion with our motion pattern network (MP-Net), a trainable model, which takes optical flow as input and outputs a per-pixel score for moving objects. Inspired by fully convolutional networks (FCNs) [32, 95, 122], we propose a related encoder-decoder style architecture to accomplish this two-label classification task. The network is trained from scratch with synthetic data [97]. An example of a such a synthetic video sequence is shown in Figure 4.1. Pixel-level ground-truth labels for training are generated automatically (see Figure 4.1(d)), and denote whether each pixel has moved in the scene. The input to the network is flow fields, such as the one shown in Figure 4.1(c). More details of the network, and the training procedure are provided in Section 5.2.2. With this training, our model learns to distinguish motion patterns of objects and background.

The appearance stream is the DeepLab network [23, 24], pretrained on the PASCAL VOC segmentation dataset, and it operates on individual video frames. With the spatial and temporal CNN features, we train the convolutional GRU component of the framework to learn a *visual memory* representation of object(s) in the scene. Given a frame t from the video sequence as input, the network extracts its spatio-temporal features and: (i) computes the segmentation using the memory representation aggregated from all frames previously seen in the video, (ii) updates the memory unit with features from t. The segmentation is improved further by processing the video in the forward and the backward directions in the memory unit, with our *bidirectional convolutional GRU*.

The contributions of the work are three-fold. First we demonstrate

that independent motion between a pair of frames can be learned, and emphasize the utility of synthetic data for this task. Second, we present an approach for moving object segmentation in unconstrained videos that does not require any manually-annotated frames in the input video (see §5.1). Our network architecture incorporates a memory unit to capture the evolution of object(s) in the scene (see §5.3). To our knowledge, this is the first recurrent network based approach for the video segmentation task. It helps address challenging scenarios where the motion patterns of the object change over time; for example, when an object in motion stops to move, abruptly, and then moves again, with potentially a different motion pattern. Finally, we present state-of-the-art results on the DAVIS [112] and Freiburg-Berkeley motion segmentation (FBMS) [105] benchmark datasets, and competitive results on SegTrack-v2 [86] (see §5.4.5). We also provide an extensive experimental analysis, with ablation studies to investigate the influence of all the components of our framework in Section 5.4.4.

5.1 Learning to segment moving objects in videos

We start by describing the overall architecture of our video object segmentation framework. It takes video frames together with their estimated optical flow as input, and outputs binary segmentations of moving objects, as shown in Figure 5.1. We target the most general form of this task, wherein objects are to be segmented in the entire video if they move in at least one frame. The proposed model is comprised of three key components: appearance and motion networks, and a visual memory module described below.

Appearance network. The purpose of the appearance stream is to produce a high-level encoding of a frame that will later aid the visual memory module in forming a representation of the moving object. It takes a $w \times h$ RGB frame as input and produces a $128 \times w/8 \times h/8$ feature representation (shown in green in Figure 5.1), which encodes the semantic content of the scene. As a baseline for this stream we use the largeFOV, VGG16-based version of the DeepLab network [23]. This network's architecture is based on dilated convolutions [23], which preserve a relatively high spatial resolution of features, and also incorporate context information in each pixel's representation. It is pretrained on a semantic segmentation dataset [34], resulting in features that can distinguish objects from background as well

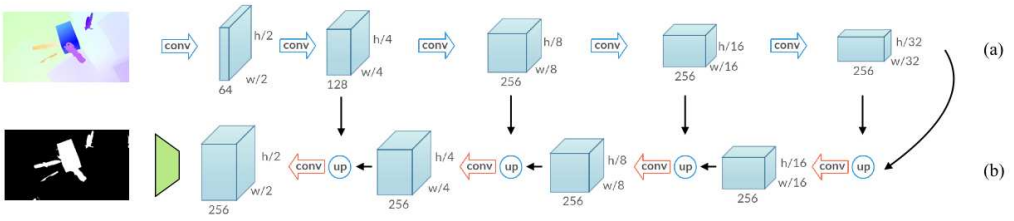


Figure 5.2 – Our motion pattern network: MP-Net. The blue arrows in the encoder part (a) denote convolutional layers, together with ReLU and max-pooling layers. The red arrows in the decoder part (b) are convolutional layers with ReLU, 'up' denotes 2×2 upsampling of the output of the previous unit. The unit shown in green represents bilinear interpolation of the output of the last decoder unit.

as from each other—a crucial aspect for the video object segmentation task. We also experiment (in §5.4.4) with upgrading the appearance stream to DeepLab-v2 [24], a more recent version of the model, where VGG16 architecture is replaced with ResNet101, and the network is additionally pretrained on the COCO semantic segmentation dataset [93].

Motion network. For the temporal stream we employ a CNN pretrained for the motion segmentation task. It is trained to estimate independently moving objects (i.e., irrespective of camera motion) based on optical flow computed from a pair of frames as input; see Section 5.2 for details. This stream (shown in yellow in Figure 5.1) produces a $w/4 \times h/4$ motion prediction output, where each value represents the likelihood of the corresponding pixel being in motion. Its output is further downsampled by a factor 2 (in w and h) to match the dimension of the appearance stream output.

The intuition behind using two streams is to benefit from their complementarity for building a strong representation of objects that evolves over time. For example, both appearance and motion networks are equally effective when an object is moving in the scene, but as soon as it becomes stationary, the motion network can not estimate the object, unlike the appearance network. We leverage this complementary nature, as done by two-stream networks for other vision tasks [128]. Note that our approach is not specific to the particular networks described above, but is in fact a general framework for video object segmentation. As shown is the Section 5.4.4, its components can easily replaced with other networks, providing scope for future improvement.

Memory module. The third component, i.e., visual memory module, takes the concatenation of appearance and motion stream outputs as its input. It refines the initial estimates from these two networks, and also memorizes the appearance and location of objects in motion to segment them in frames where: (i) they are static, or (ii) motion prediction fails. The output of this ConvGRU memory module is a $64 \times w/8 \times h/8$ feature map obtained by combining the two-stream input with the internal state of the memory module, as described in detail in Section 5.3. We further improve the model by processing the video bidirectionally; see Section 5.3.1. The output from the ConvGRU module is processed by a 1×1 convolutional layer and softmax nonlinearity to produce the final pairwise segmentation result.

5.2 Motion pattern network

Our MP-Net takes the optical flow field corresponding to two consecutive frames of a video sequence as input, and produces per-pixel motion labels. We treat each video as a sequence of frame pairs, and compute the labels independently for each pair. As shown in Figure 5.2, the network comprises several "encoding" (convolutional and max-pooling) and "decoding" (upsampling and convolutional) layers. The motion labels are produced by the last layer of the network, which are then rescaled to the original image resolution (see §5.2.1). We train the network entirely on synthetic data—a scenario where ground-truth motion labels can be acquired easily (see §5.2.2).

5.2.1 Network architecture

Our encoder-decoder style network is motivated by the goal of segmenting diverse motion patterns in flow fields, which requires a large receptive field as well as an output at the original image resolution. A large receptive field is critical to incorporate context into the model. For example, when the spatial region of support (for performing convolution) provided by a small receptive field falls entirely within an object with non-zero flow values, it is impossible to determine whether it is due to object or camera motion. On the other hand, a larger receptive field will include regions corresponding to the object as well as background, providing sufficient context to determine what is moving in the scene. The second requirement of output generated at the original image resolution is to capture fine details of objects, e.g., when only a part of the object is moving. Our network satis-

fies these two requirements with: (i) the encoder part learning features with receptive fields of increasing sizes, and (ii) the decoder part upsampling the intermediate layer outputs to finally predict labels at the full resolution.

Figure 5.2 illustrates our network architecture. Optical flow field input is processed by the encoding part of the network (denoted by (a) in the figure) to generate a coarse representation that is a 32×32 downsampled version of the input. Each 3D block here represents a feature map produced by a set of layers. In the encoding part, each feature map is a result of applying convolutions, followed by a ReLU non-linearity layer, and then a 2×2 max-pooling layer. The coarse representation learned by the final set of operations in this part, i.e., the 32×32 downsampled version, is gradually upsampled by the decoder part ((b) in the figure). In each decoder step, we first upsample the output of the previous step by 2×2, and concatenate it with the corresponding intermediate encoded representation, before max-pooling (illustrated with black arrows pointing down in the figure). This upscaled feature map is then processed with two convolutional layers, followed by non-linearities, to produce input for the next (higher-resolution) decoding step. The final decoder step produces a motion label map at half the original resolution. We perform a bilinear interpolation on this result to estimate labels at the original resolution.

5.2.2 Training with synthetic data

We need a large number of fully-labelled examples to train a convolutional network such as the one we propose. In our case, this data corresponds to videos of several types of objects, captured under different conditions (e.g., moving or still camera), with their respective moving object annotations. No large dataset of real-world scenes satisfying these requirements is currently available, predominantly due to the cost of generating ground-truth annotations and flow for every frame. We adopt the popular approach of using synthetic datasets, followed in other work [32, 97]. Specifically, we use the FlyingThings3D dataset [97] containing 2250 video sequences of several objects in motion, with ground-truth optical flow. We augment this dataset with ground-truth moving object labels, which are accurately estimated using the disparity values and camera parameters available in the dataset, as outlined in Section 5.4.1. See Figure **??**(d) for an illustration.

We train the network with mini-batch SGD under several settings. The one trained with ground-truth optical flow as input shows the best performance. This is analyzed in detail in Section 5.4.3. Note that, while we use ground-truth flow for training and evaluating the network on synthetic

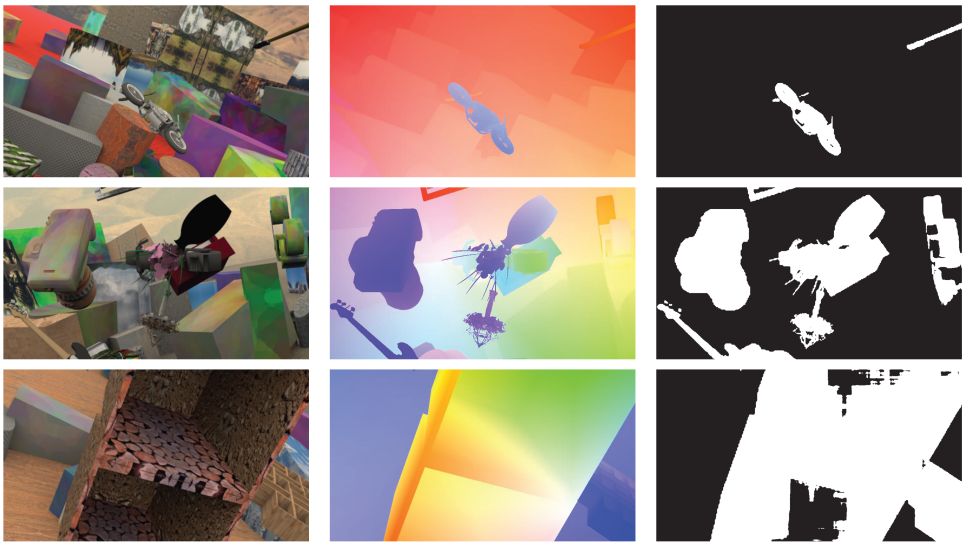
datasets, all our results on real-world test data use only the estimated optical flow. After convergence of the training procedure, we obtain a learned model for motion patterns.

Our approach capitalizes on the recent success of CNNs for pixel-level labeling tasks, such as semantic image segmentation, which learn feature representations at multiple scales in the RGB space. The key to their top performance is the ability to capture local patterns in images. Various types of object and camera motions also produce consistent local patterns in the flow field, which our model is able to learn to recognize. This gives us a clear advantage over other pixel-level motion estimation techniques [13, 101] that can not detect local patterns. Motion boundary based heuristics used in [108] can be seen as one particular type of pattern, representing independent object motion. Our model is able to learn many such patterns, which greatly improves the quality and robustness of motion estimation.

5.2.3 Detecting motion patterns

We apply our trained model on synthetic (FlyingThings3D) as well as real-world (DAVIS, FBMS, SegTrack-v2) test data. Figure 5.3 shows sample predictions of our model on the FlyingThings3D test set with ground-truth optical flow as input. Examples in the first two rows show that our model accurately identifies fine details in objects: thin structures even when they move subtlely, such as the neck of the guitar in the top-right corner in the first row (see the subtle motion in the optical flow field (b)), fine structures like leaves in the vase, and the guitar's headstock in the second row. Furthermore, our method successfully handles objects exhibiting highly varying motions in the second example. The third row shows a limiting case, where the receptive field of our network falls entirely within the interior of a large object, as the moving object dominates. Traditional approaches, such as RANSAC, do not work in this case either.

In order to detect motion patterns in real-world videos, we first compute optical flow with popular methods [19, 61, 120, 133]. With this flow as input to the network, we estimate a motion label map, as shown in the examples in Figure 5.4(c). Although the prediction of our frame-pair feedforward model is accurate in several regions in the frame ((c) in the figure), we are faced with two challenges, which were not observed in the synthetic training set. The first one is motion of *stuff* [6] in a scene, e.g., patterns on the water due to the kiteboarder's motion (first row in the figure), which is irrelevant for moving object segmentation. The second one is significant errors in optical flow, e.g., in front of the pram ((b) in the bottom row in the figure). Furthermore, this motion segmentation approach is purely



(a) (b) (c)

Figure 5.3 – Each row shows: (a) example frame from a sequence in FlyingThings3D, (b) ground-truth optical flow of (a), which illustrates motion of both foreground objects and background, with respect to the next frame, and (c) our estimate of moving objects in this scene with ground-truth optical flow as input.

frame-based, thus unable to exploit temporal consistency in a video, and does not segment object in frames where they stop moving. In our previous work [137] we introduced post-processing steps to handle some of these problems. In particular, we incorporated an objectness map computed with object proposals [115] to suppress motion corresponding to stuff, as well as false positives due to errors in flow estimation. This post-processing allowed the method to achieve competitive results, but it remained frame-level. The video object segmentation framework presented in this chapter addresses all these issues, as shown experimentally in Section 5.4.5.

5.3 ConvGRU visual memory module

The key component of the ConvGRU module is the state matrix h, which encodes the visual memory. For frame t in the video sequence, ConvGRU uses the two-stream representation x_t and the previous state h_{t-1} to compute the new state h_t. The dynamics of this computation are guided by an update gate z_t, a forget gate r_t. The states and the gates are 3D tensors, and can characterize spatio-temporal patterns in the video, effec-

(a) (b) (c)

Figure 5.4 – Sample results on the DAVIS dataset showing all the components of our approach. Each row shows: (a) video frame, (b) optical flow estimated with LDOF [19], (c) output of our MP-Net with LDOF flow as input.

tively memorizing which objects move, and where they move to. These components are computed with convolutional operators and nonlinearities as follows.

$$z_t = \sigma(x_t * w_{xz} + h_{t-1} * w_{hz} + b_z), \tag{5.1}$$
$$r_t = \sigma(x_t * w_{xr} + h_{t-1} * w_{hr} + b_r), \tag{5.2}$$
$$\tilde{h}_t = \tanh(x_t * w_{x\tilde{h}} + r_t \odot h_{t-1} * w_{h\tilde{h}} + b_{\tilde{h}}), \tag{5.3}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{5.4}$$

where $\odot$ denotes element-wise multiplication, $*$ represents a convolutional operation, σ is the sigmoid function, w's are learned transformations, and b's are bias terms.

The new state h_t in ((5.4)) is a weighted combination of the previous state h_{t-1} and the candidate memory $\tilde{h}_t$. The update gate z_t determines how much of this memory is incorporated into the new state. If z_t is close to zero, the memory represented by $\tilde{h}_t$ is ignored. The reset gate r_t controls the influence of the previous state h_{t-1} on the candidate memory $\tilde{h}_t$ in ((5.3)), i.e., how much of the previous state is let through into the candidate memory. If r_t is close to zero, the unit forgets its previously computed state h_{t-1}.

The gates and the candidate memory are computed with convolutional operations over x_t and h_{t-1} shown in equations ((5.1)-(5.3)). We illustrate the computation of the candidate memory state $\tilde{h}_t$ in Figure 5.5. The state at $t-1$, h_{t-1}, is first multiplied (element-wise) with the reset gate r_t. This

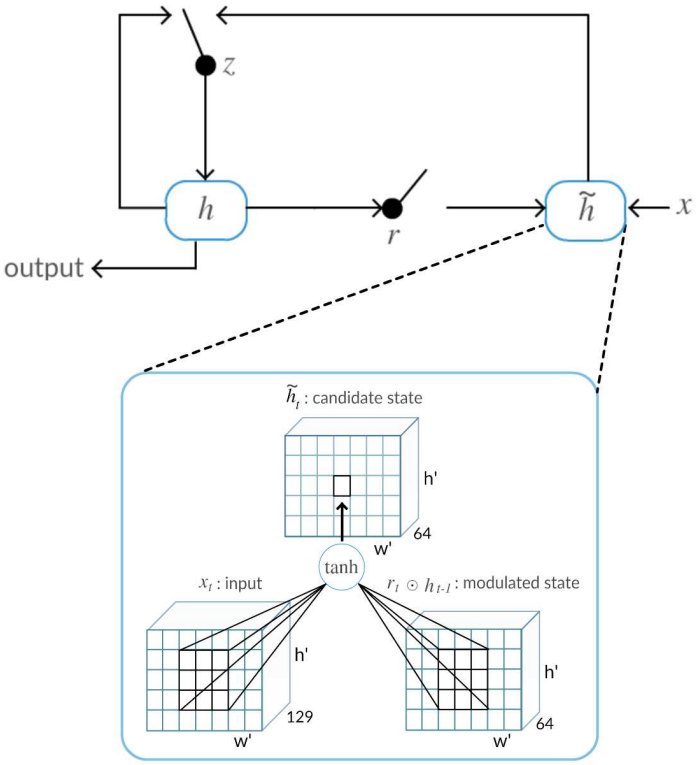


Figure 5.5 – Illustration of ConvGRU with details for the candidate hidden state module, where $\tilde{h}_t$ is computed with two convolutional operations and a tanh nonlinearity.

modulated state representation and the input x_t are then convolved with learned transformations, $w_{h\tilde{h}}$ and $w_{x\tilde{h}}$ respectively, summed together with a bias term $b_{\tilde{h}}$, and passed though a tanh nonlinearity. In other words, the visual memory representation of a pixel is determined not only by the input and the previous state at that pixel, but also its local neighborhood. Increasing the size of the convolutional kernels allows the model to handle spatio-temporal patterns with larger motion.

The update and reset gates, z_t and r_t, are computed in an analogous fashion using a sigmoid function instead of tanh. Our ConvGRU applies a total of six convolutional operations at each time step. All the operations detailed here are fully differentiable, and thus the parameters of the convolutions (w's and b's) can be learned in an end-to-end fashion with back propagation through time [149]. In summary, the model learns to combine appearance features of the current frame with the memorized video representation to refine motion predictions, or even fully restore them from the

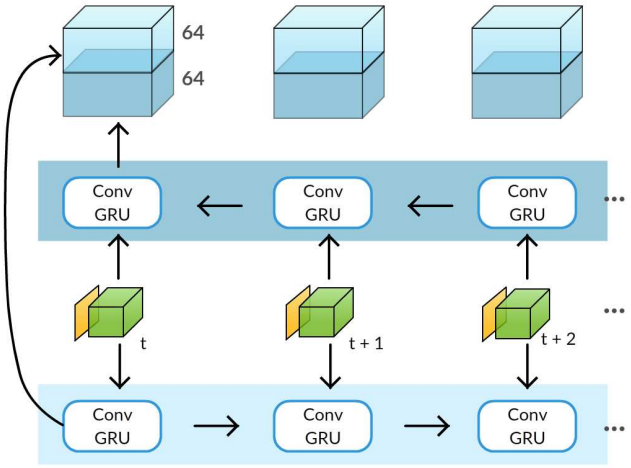


Figure 5.6 – Illustration of the bidirectional processing with our ConvGRU module.

previous observations in case a moving object becomes stationary.

5.3.1 Bidirectional processing

Consider an example where an object is stationary at the beginning of a video sequence, and starts to move in the latter frames. Our approach described so far, which processes video frames sequentially (in the forward direction), starting with the first frame can not segment the object in the initial frames. This is due to the lack of prior memory representation of the object in the first frame. We improve our framework with a bidirectional processing step, inspired by the application of recurrent models bidirectionally in the speech domain [46, 48].

The bidirectional variant of our ConvGRU is illustrated in Figure 5.6. It is composed of two ConvGRU instances with identical learned weights, which are run in parallel. The first one processes frames in the forward direction, starting with the first frame (shown at the bottom in the figure). The second instance process frames in the backward direction, starting with the last video frame (shown at the top in the figure). The activations from these two directions are concatenated at each time step, as shown in the figure, to produce a $128 \times w/8 \times h/8$ output. It is then passed through a 3×3 convolutional layer to finally produce a $64 \times w/8 \times h/8$ output for each frame. Pixel-wise segmentation from this activation is the result of the last 1×1 convolutional layer and softmax nonlinearity, as in the unidirectional case.

Bidirectional ConvGRU is used both in training and in testing, allowing the model to learn to aggregate information over the entire video. In addition to handling cases where objects move in the latter frames, it improves the ability of the model to correct motion prediction errors. As discussed in the experimental evaluation, bidirectional ConvGRU improves segmentation performance by nearly 3% on the DAVIS dataset (see Table 5.4). The influence of bidirectional processing is more prominent on the FBMS dataset, where objects can be static in the beginning of a video, with 5% improvement over the unidirectional variant.

5.3.2 Training

We train our visual memory module with the back propagation through time algorithm [149], which unrolls the recurrent network for n time steps and keeps all the intermediate activations to compute the gradients. Thus, our ConvGRU model, which has six internal convolutional layers, trained on a video sequence of length n, is equivalent to a $6n$ layer CNN for the unidirectional variant, or $12n$ for the bidirectional model at training time. This memory requirement makes it infeasible to train the whole model, including appearance and motion streams, end-to-end. We resort to using pretrained versions of the appearance and motion networks, and train the ConvGRU.

In contrast to our motion segmentation model, which is learned on synthetic videos, we use the training split of the DAVIS dataset [112] for learning the ConvGRU weights. Despite being an order of magnitude smaller, DAVIS consists of realistic videos, which turns out to be crucial for effective use of appearance stream to correct motion estimation errors (see §5.4.4). Since objects move in all the frames in DAVIS, it biases the memory module towards the presence of an uninterrupted motion stream. This results in the ConvGRU learned from this data failing, when an object stops to move in a test sequence. We augment the training data to simulate such *stop-and-go* scenarios to learn a more robust model for realistic videos. To this end, we create additional sequences where ground truth moving object segmentation (instead of responses from the motion network) is provided for all the frames, except for the last five frames. No motion input is used for these last five frames. This ensures that the model does not have access to motion input towards the end of the sequence, simulating a case where objects stop moving. Given that ground truth segmentation determines the loss for training, i.e., it is used for all the frames, ConvGRU explicitly memorizes the moving object in the initial part of the sequence, and then segments it in frames where motion input is missing. We do a similar train-

ing set augmentation by using ground truth segmentation as motion input for all but the first five frames.

5.4 Experiments

5.4.1 Datasets and Evaluation

We now review the metrics and datasets used for evaluation of video segmentation methods. Similarly to semantic segmentation, the accuracy of video segmentation methods is usually measured with intersection over union (see Section 3.2.1 for details). The only exception to this is the FBMS dataset, where performance is measured with precision, recall and F-measure scores. Precision captures the fraction of true positives among all the predicted positives:

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}, \tag{5.5}$$

and recall captures the fraction of the positive examples correctly identified by the model to all the positives in the data:

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{5.6}$$

F-measures then combines these complementary statistics with a harmonic mean:

$$F = \frac{2 \cdot precision \cdot recall}{recision + recall}. \tag{5.7}$$

The goal is, similarly to IoU, to assign a high score to models that both accurately capture the objects and do not segment background regions as moving. Another property in which FBMS differs from the other datasets used in our experiments (except for SegTrack-v2) is that its annotations are instance-level, whereas we are treating video object segmentation as a foreground-background classification problem. Thus, we convert FBMS and SegTrack-v2 annotation to binary ones by merging all the foreground labels into a single category, as in [134].

Measuring accuracy on the frame level is important for evaluating the performance of video segmentation methods, but using frame-level IoU or F-measure as the only criterion completely ignores the temporal dimension of the data. To address this issue, a temporal stability metric $\mathcal{T}$ was introduced by [112]. It captures the smoothness of segmentation over time. This measure is hard to formalize, however, since some mask deformations

are acceptable, whereas others are not. In practice the authors resort to transforming the mask M_t into polygons representing its contours $P(M_t)$. They then describe each point $p_i^t \in M_t$ using the Shape Context Descriptor (SCD) [11] and look for a matching between points p_t^i, p_{t+1}^j that minimizes the SCD distances between the matched points, while preserving the order of points in shapes. The resulting mean cost per matched point is used as the measure of temporal stability. Intuitively, matching compensates motion and small deformations, but it does not compensate the oscillations and inaccuracies of the mask contours, which are captured by this measure.

Finally, [112] introduce an accuracy measure complementary to the intersection over union. IoU gives equal importance to all the pixel in an image, whereas for segmentation problems, arguably, pixels on the boundaries of the objects are more important than the rest. Thus, the authors suggest to measure the accuracy of the segmentation on the contours separately. To this end they compute a bipartite matching between the contour points of the predicted and groundtruth masks $c(M)$ and $c(G)$ respectively. The contour accuracy is then evaluated with the F-measure on the obtained set of matches.

We use five datasets in the experimental analysis: FT3D and DAVIS for training and test, FusionSeg only for training, and FBMS and SegTrack-v2 only for test.

FlyingThings3D (FT3D). We train our motion segmentation network with the synthetic FlyingThings3D dataset [97]. It contains videos of various objects flying along randomized trajectories, in randomly constructed scenes. The video sequences are generated with complex camera motion, which is also randomized. FT3D comprises 2700 videos, each containing 10 stereo frames. The dataset is split into training and test sets, with 2250 and 450 videos respectively. Ground-truth optical flow, disparity, intrinsic and extrinsic camera parameters, and object instance segmentation masks are provided for all the videos. No annotation is directly available to distinguish moving objects from stationary ones, which is required to train our network. We extract this from the data provided as follows. With the given camera parameters and the stereo image pair, we first compute the 3D coordinates of all the pixels in a video frame t. Using ground-truth flow between frames t and $t+1$ to find a pair of corresponding pixels, we retrieve their respective 3D scene points. Now, if the pixel has not undergone any independent motion between these two frames, the scene coordinates will be identical (up to small rounding errors). We have made these labels publicly available on our project website [4]. Performance on the test set is mea-

sured as the standard intersection over union score between the predicted segmentation and the ground-truth masks.

DAVIS. We use the densely annotated video segmentation dataset [112] for evaluation as well as for training our visual memory module. DAVIS contains 50 full HD videos, featuring diverse types of object and camera motion. It includes challenging examples with occlusion, motion blur and appearance changes. Accurate pixel-level annotations are provided for the moving object in all the video frames. A single object is annotated in each video, even if there are multiple moving objects in the scene. Following the 30/20 training/validation split provided with the dataset, we train the visual memory module on the 30 sequences, and test on the 20 validation videos. Note that our motion segmentation model is also evaluated spearately on the entire trainval set, as it is trained exclusively on FT3D. We follow the protocol in [112] and use images downsampled by a factor of two.

FusionSeg. This dataset was introduced recently in [63], and consists of 84929 pairs of frames extracted from the ImageNet-Video dataset [124]. The frames are annotated with an automatic segmentation method, which combines a foreground-background appearance-based model with ground truth bounding box annotations available in ImageNet-Video. Annotations obtained in this way may be inaccurate, but are useful for analyzing the impact of learning the motion network on these realistic examples, in contrast to using synthetic examples; see Section 5.4.3. We will refer to this dataset as FusionSeg in the rest of the chapter.

FBMS. The Freiburg-Berkeley motion segmentation dataset [105] is composed of 59 videos with ground truth annotations in a subset of frames. In contrast to DAVIS, it has multiple moving objects in several videos with instance-level annotations. Also, objects may move only in a fraction of the frames, but are annotated in frames where they do not exhibit independent motion. The dataset is split into training and test sets. Following the standard protocol on this dataset [68], we do not train on any of these sequences, and evaluate separately on both these splits.

SegTrack-v2. It contains 14 videos with instance-level moving object annotations in all the frames. We convert these annotations into a binary form for evaluation and use intersection over union as the performance

measure. Note that some videos in this dataset are of very low resolution, which appears to favor traditional forms of appearance representation.

5.4.2 Implementation details

Appearance stream. For the experiments using DeepLab-v1, we extract features from the fc6 layer of the network, which has a dilation of 12. This approach cannot be followed for DeepLab-v2 however, since dilation is applied to fc8, the prediction layer, in this improved model. Thus, extracting fc6 or fc7 features of DeepLab-v2 would result in a decreased field of view compared to the baseline v1 model. Moreover, there are four independent prediction layers in v2 with dilations 6, 12, 18 and 24, whose outputs are averaged. To make the feature representation derived from the two architectures compatible, we introduce four new penultimate convolutional layers with corresponding dilations, kernel size 3 and feature dimension 512 to the DeepLab-v2 architecture. The maximum response over these four feature maps is then passed to a single prediction layer. We finetune this model on PASCAL VOC 2012 for semantic segmentation. The features after the max operation are used as the appearance representation in our final model, and correspond to an improved version of fc6 features from DeepLab-v1. This representation is further passed through two 1×1 convolutional layers, interleaved with *tanh* nonlinearities, to reduce the dimension to 128.

Training MP-Net. We use mini-batch SGD with a batch size of 13 images—the maximum possible due to GPU memory constraints. The network is trained from scratch with learning rate set to 0.003, momentum to 0.9, and weight decay to 0.005. Training is done for 27 epochs, and the learning rate and weight decay are decreased by a factor of 0.1 after every 9 epochs. We downsample the original frames of the FT3D training set by a factor 2, and perform data augmentation by random cropping and mirroring. Batch normalization [62] is applied to all the convolutional layers of the network.

Training visual memory module. We minimize the binary cross-entropy loss using back-propagation through time and RMSProp [135] with a learning rate of 10^{-4}. The learning rate is gradually decreased after every epoch. Weight decay is set to 0.005. Initialization of all the convolutional layers, except for those inside the ConvGRU, is done with the standard *xavier* method [44]. We clip the gradients to the $[-50, 50]$ range before each parameter update, to avoid numerical issues [45]. We form batches of size 14

by randomly selecting a video, and a subset of 14 consecutive frames in it. Random cropping and flipping of the sequences is also performed for data augmentation. Our full model uses 7×7 convolutions in all the ConvGRU operations. The weights of the two 1×1 convolutional (dimensionality reduction) layers in the appearance network and the final 1×1 convolutional layer following the memory module are learned jointly with the memory module. The model is trained for 30000 iterations and the proportion of batches with additional sequences (see Section 5.3.2) is set to 20%.

Other details. We perform zero-mean normalization of the flow field vectors, similar to [128]. When using flow angle and magnitude together (which we refer to as flow angle field), we scale the magnitude component, to bring the two channels to the same range. Our final model uses a fully-connected CRF [76] to refine boundaries in a post-processing step. The parameters of this CRF are set to values used for a related pixel-level segmentation task [23]. Many sequences in FBMS are several hundred frames long and do not fit into GPU memory during evaluation. We apply our method in a sliding window fashion in such cases, with a window of 130 frames and a step size of 50. Our model is implemented in the Torch framework.

5.4.3 Motion pattern network

We first analyze the different design choices in our MP-Net, and then study the influence of training data and optical flow representation on the motion prediction performance.

Influence of input modalities

We analyze the influence of different input modalities, such as RGB data (single frame and image pair), optical flow field (ground truth and estimated one) directly as flow vectors, i.e., flow in x and y axes, or as angle field (flow vector angle concatenated with flow magnitude), and a combination of RGB data and flow, on training MP-Net. These results are presented on the FT3D test set and also on DAVIS, to study how well the observations on synthetic videos transfer to the real-world ones, in Table 5.1. For computational reasons we train and test with different modalities on a smaller version of our MP-Net, with one decoder unit instead of four. Then we pick the best modality to train and test the full, deeper version of the network.

From Table 5.1, the performance on DAVIS is lower than on FT3D. This is expected as there is domain change from synthetic to real data, and

# dec.	Trained on FT3D with ...	FT3D	DAVIS
1	RGB single frame	68.1	12.7
	RGB pair	69.1	16.6
	GT flow	74.5	44.3
	GT angle field	73.1	46.6
	RGB + GT angle field	74.8	39.6
	LDOF angle field	63.2	38.1
4	GT angle field	85.9	52.4

Table 5.1 – Comparing the influence of different input modalities on the FlyingThings3D (FT3D) test set and DAVIS. Performance is shown as mean intersection over union scores. # dec. refers to the number of decoder units in our MP-Net. Ground-truth flow is used for evaluation on FT3D and LDOF flow for DAVIS.

that we use ground truth optical flow as input for FT3D test data, but estimated flow [19, 133] for DAVIS. As a baseline, we train on single RGB frames ('RGB single frame' in the table). Clearly, no motion patterns can be learned in this case, but the network performs reasonably on FT3D test (68.1), as it learns to correlate object appearance with its motion. This intuition is confirmed by the fact that 'RGB single frame' fails on DAVIS (12.7), where the appearance of objects and background is significantly different from FT3D. MP-Net trained on 'RGB pair', i.e., RGB data of two consecutive frames concatenated, performs slightly better on both FT3D (69.1) and DAVIS (16.6), suggesting that it captures some motion-like information, but continues to rely on appearance, as it does not transfer well to DAVIS.

Training on ground-truth flow vectors corresponding to the image pair ('GT flow') improves the performance on FT3D by 5.4% and on DAVIS significantly (27.7%). This shows that MP-Net learned on flow from synthetic examples can be transferred to real-world videos. We then experiment with flow angle as part of the input. As discussed in [101], flow orientations are independent of depth from the camera, unlike flow vectors, when the camera is undergoing only translational motion. Using the ground truth flow angle field (concatenation of flow angles and magnitudes) as input ('GT angle field'), we note a slight decrease in IoU score on FT3D (1.4%), where strong camera rotations are abundant, but in real examples, such motion is usually mild. Hence, 'GT angle field' improves IoU on DAVIS by 2.3%. We use angle field representation in all further experiments.

Flow in test	FT3D	DAVIS
LDOF	58.7	52.4
EpicFlow	52.5	56.9
FlowNet 2.0	66.3	62.6

Table 5.2 – Performance of the best MP-Net variant (4 decoder units trained on GT angle field) with different flow inputs (LDOF, EpicFlow, FlowNet 2.0) on DAVIS and FT3D.

Using a concatenated flow and RGB representation ('RGB + GT angle field') performs better on FT3D (by 1.7%), but is poorer by 7% on DAVIS, re-confirming our observation that appearance features are not consistent between the two datasets. Finally, training on computed flow [19] ('LDOF angle field') leads to significant drop on both the datasets: 9.9% on FT3D (with GT flow for testing) and 8.5% on DAVIS, showing the importance of high-quality training data for learning accurate models. The full version of our MP-Net, with 4 decoder units, improves the IoU by 12.8% on FT3D and 5.8% on DAVIS over its shallower one-unit equivalent.

Notice that the performance of our full model on FT3D is excellent, with the remaining errors mostly due to inherently ambiguous cases like objects moving close to the camera (see third row in Figure 5.3), or very strong object/camera motion. On DAVIS, the results are considerably lower despite less challenging motion. To investigate the extent to which this is due to errors in flow, we study the effect of flow quality in the following section.

Effect of the flow quality

We evaluate the performance of MP-Net using two recent flow estimation methods, EpicFlow [120] and FlowNet 2.0 [61], and LDOF [19, 133], a more classical approach, on the FT3D test and DAVIS datasets in Table 5.2. We observe a significant drop in performance of 27.2% (from 85.9% to 58.7%) on FT3D when using LDOF, compared to evaluation with the ground truth in Table 5.1. This confirms the impact of optical flow quality and suggests that improvements in flow estimation can increase the performance of our method on real-world videos, where no ground truth flow is available.

We experimentally demonstrate this improvement, by utilizing state-of-the-art flow estimation methods, instead of LDOF. EpicFlow, which leverages motion contours, produces more accurate object boundaries, and improves over MP-Net using LDOF by 4.5% on DAVIS. On FT3D though it

Trained on	FT3D	DAVIS
FT3D	85.9	62.6
FusionSeg	40.8	60.4
FT3D + FusionSeg	43.0	63.9
DAVIS	34.0	62.3
FT3D + DAVIS	45.7	66.7
FT3D + FusionSeg + DAVIS	40.8	68.6

Table 5.3 – Performance of the best MP-Net variant trained with different datasets on FT3D test and DAVIS validation sets. FlowNet 2.0 is used for flow estimation on DAVIS both in training and in testing in all these experiments.

leads to a 6.2% decrease in performance. We observe that this is due to EpicFlow, which does produce more accurate object boundaries, but also smooths out small objects and objects with tiny motions. This smoothing appears to be beneficial on real videos, but degrades the performance on synthetic FT3D videos. FlowNet 2.0, which is a CNN-based method trained on a mixture of synthetic and real videos to estimate optical flow from a pair of frames, further improves the performance on DAVIS by 5.7%. It also achieves better results on FT3D, with a 7.6% improvement over LDOF. The remaining gap of 19.6% between the ground truth flow and FlowNet 2.0 performance on FT3D shows the potential for future improvement of flow estimation methods.

Training on real videos

We also experiment with training out MP-Net on FusionSeg and DAVIS, in order to explore the value real videos can bring in learning a motion segmentation model, compared to training exclusively on synthetic videos. On one hand, real videos contain motion patterns that have similar statistics to those encountered in the testing phase, but on the other hand, no ground truth flow is available, which was shown to be crucial for obtaining top performance when training on FT3D (see §5.4.3).

All the models in this experiment are trained on flow extracted with the state-of-the-art FlowNet 2.0 optical flow estimation approach, in order to minimize the influence of errors in flow. FlowNet 2.0 is also used for evaluation on the DAVIS validation set, whereas ground truth flow is used for FT3D test set. As shown in Table 5.3, the model trained on FusionSeg is 2.2% below the one trained on synthetic data in the case of DAVIS.

On FT3D, its performance drops by 45.1%. This shows that the synthetic dataset contains a lot more challenging motions than those typically encountered in real videos, and although a model learned on synthetic data can generalize to real data, the converse does not hold. Learning the model only on real videos also does not bring any improvement on DAVIS, due to errors in flow estimation and labels in FusionSeg outweighing the potential benefits. We then finetune the FT3D-trained model on FusionSeg to leverage the benefits of the two domains. This leads to a notable improvement on both datasets, e.g., 3.5% on DAVIS compared to the model trained on FusionSeg alone. The results on synthetic FT3D videos, despite the improvement over the FusionSeg-trained model, remain low however, showing the significant difference between the two domains.

To further explore the use of real videos, we train our motion estimation model on the DAVIS training set. This dataset contains only 2079 frames, compared to 84929 in FusionSeg, but they are manually annotated, removing one source of errors due to incorrect labels from training. The performance on DAVIS increases by 1.9% with this, compared to training on FusionSeg. On FT3D, though, IoU decreases by 6.8%, because the variety of motions in DAVIS is even smaller than that seen in FusionSeg. Combining the synthetic and real datasets, i.e., training on FT3D and finetuning on DAVIS, improves the performance on both FT3D and DAVIS. Finetuning the FT3D-trained model with FusionSeg and then DAVIS training data further improves the performance on the DAVIS validation set, but results in a drop in the case of FT3D, as the model further away from synthetic data.

5.4.4 Video object segmentation framework

We use the motion network trained on FT3D with LDOF optical flow, unless stated otherwise, and study the influence of design choices on our overall segmentation framework.

Ablation study

Table 5.4 demonstrates the influence of different components of our approach on the DAVIS validation set. First, we study the role of the appearance stream. As a baseline, we remove it completely ("no" in "App stream" in the table), i.e., the output of the motion stream is the only input to our visual memory module. In this setting, the memory module lacks sufficient information to produce accurate segmentations, which results in an 26.6% drop in performance compared to the method where the

Aspect	Variant	Mean IoU
Ours (fc6, ConvGRU, Bidir, DAVIS)		70.1
App stream	no	43.5
	RGB	58.3
	2-layer CNN	60.9
	DeepLab fc7	69.8
	DeepLab conv5	67.7
	DeepLab-v2	72.5
App pretrain	ImageNet only	64.1
Motion stream	no	59.6
Memory module	no	64.1
	ConvRNN	68.7
	ConvLSTM	68.9
Bidir processing	no	67.2
Train data	FT3D GT Flow	55.3
	FT3D LDOF Flow	59.6

Table 5.4 – Ablation study on the DAVIS validation set showing variants of appearance and motion streams and memory module. "Ours" refers to the model using fc6 appearance features together with a motion stream, and a bidirectional ConvGRU trained on DAVIS.

appearance stream with fc6 features is used ("Ours" in the table). We then provide raw RGB frames, concatenated with the motion prediction, as input to the ConvGRU. This simplest form of image representation leads to a 14.8% improvement, compared to the motion only model, showing the importance of the appearance features. The variant where RGB input is passed through two convolutional layers, interleaved with tanh nonlinearities, that are trained jointly with the memory module ("2-layer CNN"), further improves this. This shows the potential of learning appearance representation as a part of the video segmentation pipeline. Next, we compare features extracted from the fc7 and conv5 layers of the DeepLab model to those from fc6 used by default in our method. Features from fc7 and fc6 show comparable performance, but fc7 ones are more expensive to compute. Conv5 features perform significantly worse, perhaps due to a smaller field of view. Finally, we replace the VGG16-based DeepLab architecture with the ResNet101-based DeepLab-v2, as described in Section 5.1. This improves the performance over DeepLab-v1 by 2.4%, which is consistent with our previous observations that better representations directly affect the overall

performance of the method. We thus use DeepLab-v2 appearance stream in our final model.

The importance of appearance network pretrained on the semantic segmentation task is highlighted by the "ImageNet only" variant in Table 5.4, where the PASCAL VOC pretrained DeepLab segmentation network is replaced with a network trained on ImageNet classification. Although ImageNet pretraining provides a rich feature representation, it is less suitable for the video object segmentation task, which is confirmed by an 6% drop in performance. Discarding the motion information ("no" in "Motion stream"), although being 10.5% below our complete method, still outperforms most of the motion-based approaches on DAVIS (see Table 5.6). This variant learns foreground/background segmentation, which is sufficient for videos with a single dominant object, but fails in more challenging cases. Section 5.4.4 presents additional experiments to explore the quality of motion estimation during the training and testing phases.

Next, we evaluate the design choices in the visual memory module. We replaced the memory module (ConvGRU) with a stack of six convolutional layers to obtain 'no memory' variant of our model ("no" in "Memory module" in Table 5.4), but with the same number of parameters. This variant results in a 6% drop in performance compared to our full model with ConvGRU on the DAVIS validation set. The performance of the 'no memory' variant is comparable to 63.3, the performance of "MP-Net+Obj," the approach without any memory (see Table 2 in [137]). Using a simple recurrent model (ConvRNN) results in a slight decrease in performance. Such simpler architectures can be used in case of a memory vs segmentation quality trade off. The other variant using ConvLSTM is comparable to ConvRNN, possibly due to the lack of sufficient training data. Performing unidirectional processing instead of a bidirectional one decreases the performance by nearly 3% ("no" in "Bidir processing").

Lastly, we train two variants ("FT3D GT Flow" and "FT3D LDOF Flow") on the synthetic FT3D dataset [97] instead of DAVIS. Both of them show a significantly lower performance than our method trained on DAVIS. This is due to the appearance of synthetic FT3D videos being very different from the real-world ones. The variant trained on ground truth flow (GT Flow) is inferior to that trained on LDOF flow because the motion network (MP-Net) achieves a high performance on FT3D with ground truth flow, and thus our visual memory module learns to simply follow the motion stream output.

Train	Test	Mean IoU	+ CRF
FT3D + LDOF	FT3D + LDOF	72.5	76.8
FT3D + LDOF	FSeg + FNet	72.0	75.3
FSeg + FNet	FSeg + FNet	73.3	78.2
FSeg + FNet	FT3D + LDOF	70.2	76.2

Table 5.5 – Influence of motion stream variants, used in training and test phases, on the DAVIS validation set. 'FT3D + LDOF' corresponds to the segmentation model with baseline MP-Net, trained on FT3D, and ConvGRU module trained on DAVIS with LDOF. 'FSeg + FNet' is the variant with improved MP-Net finetuned on FusionSeg, and ConvGRU module trained on DAVIS with FlowNet 2.0.

Influence of the motion network

In Sections 5.4.3 and 5.4.3 we have demonstrated that the performance of MP-Net can be improved by using more accurate optical flow estimation methods, and finetuning the network on FusionSeg and DAVIS. Here we explore the influence of these improvements in motion estimation on our video object segmentation framework. In Table 5.5 we evaluate the best version of our framework so far (DeepLab-v2 appearance stream, ConvGRU memory module trained on DAVIS, Bi-directional processing) with baseline (trained on FT3D only, and uses LDOF for flow estimation) and improved versions of MP-Net. The improved versions of MP-Net are finetuned on FusionSeg and use FlowNet 2.0 for flow estimation. Note that the variant finetuned on FusionSeg and then on DAVIS (see Section 5.4.3) leads to a drop in performance due to overfitting of the segmentation model on the small number of sequences in the DAVIS training set.

The main observation from the results in Table 5.5 is that our approach is fairly robust to the motion model being used. The performance differs by at most 3% here, whereas these component motion models differ by 11.5%, as seen in Tables 5.2 and 5.3. This shows that the visual memory module learns to use appearance and temporal consistency cues to overcome variations in quality of motion estimation.

The performance on the DAVIS validation set is best when the same motion model is used in the training and the test phases; see the second and the third rows in Table 5.5 for a comparison. This is expected because ConvGRU adapts to the motion model used in training, and suffers from a domain shift problem, if this model is replaced during the test phase. The variant trained and tested with the 'FSeg + FNet' model (row 3 in

Measure		CVOS	KEY	MSG	NLC	CUT	FST	MP-Net	FSG	ARP	Ours
$\mathcal{J}$	Mean	48.2	49.8	53.3	55.1	55.2	55.8	70.0	70.7	76.2	78.2
	Recall	54.0	59.1	61.6	55.8	57.5	64.9	85.0	83.5	91.1	89.1
	Decay	10.5	14.1	2.4	12.6	2.3	0.0	1.4	1.5	7.0	4.1
$\mathcal{F}$	Mean	44.7	42.7	50.8	52.3	55.2	51.1	65.9	65.3	70.6	75.9
	Recall	52.6	37.5	60.0	51.9	61.0	51.6	79.2	73.8	83.5	84.7
	Decay	11.7	10.6	5.1	11.4	3.4	2.9	2.5	1.8	7.9	3.5
$\mathcal{T}$	Mean	24.4	25.2	29.1	41.4	26.3	34.3	56.3	32.8	39.3	20.2

Table 5.6 – Comparison to the state-of-the-art methods on DAVIS with intersection over union ($\mathcal{J}$), F-measure ($\mathcal{F}$), and temporal stability ($\mathcal{T}$).

the table), which shows the best performance, with or without the CRF post-processing is used in the remainder of the chapter.

5.4.5 Comparison to the state-of-the-art

DAVIS. Table 5.6 compares our approach to the state-of-the-art methods on DAVIS. In addition to comparing our results to the top-performing unsupervised approaches reported in [112], we included the results of recent methods from the benchmark website:[1] CUT [68], FSG [63] and ARP [74], as well as the frame-level variant of our method: MP-Net [137]. Our method outperforms ARP [74], the previous state of the art by 2% on the mean IoU measure. We also observe an 8.2% improvement over MP-Net in mean IoU and 36.1% in temporal stability, which clearly demonstrates the significance of the visual memory module.

Figure 5.7 shows qualitative results of our approach, and the next three top-performing methods on DAVIS: MP-Net [137], FSG [63] and ARP [74]. In the first row, our method fully segments the dancer, whereas MP-Net and FSG miss various parts of the person and ARP segments some of the people in the background. All these approaches use heuristics to combine motion and appearance cues, which become unreliable in cluttered scenes with many objects. Our approach does not include any heuristics, which makes it robust to this type of errors. In the second row, all the methods segment the car, but only our approach does not leak into other cars in the video, showing high discriminability. In the next row, our approach is able to fully segment a complex object, whereas the other methods either miss parts of it (MP-Net and FSG) or segment background regions as moving (ARP). In the last row, we illustrate a failure case of our method. The people in the background move in some of the frames in this example. MP-

1. http://davischallenge.org/soa_compare.html

Figure 5.7 – Qualitative comparison with the top-performing methods on DAVIS. Left to right: ground truth, results of MP-Net [137], FSG [63], ARP [74], and our method.

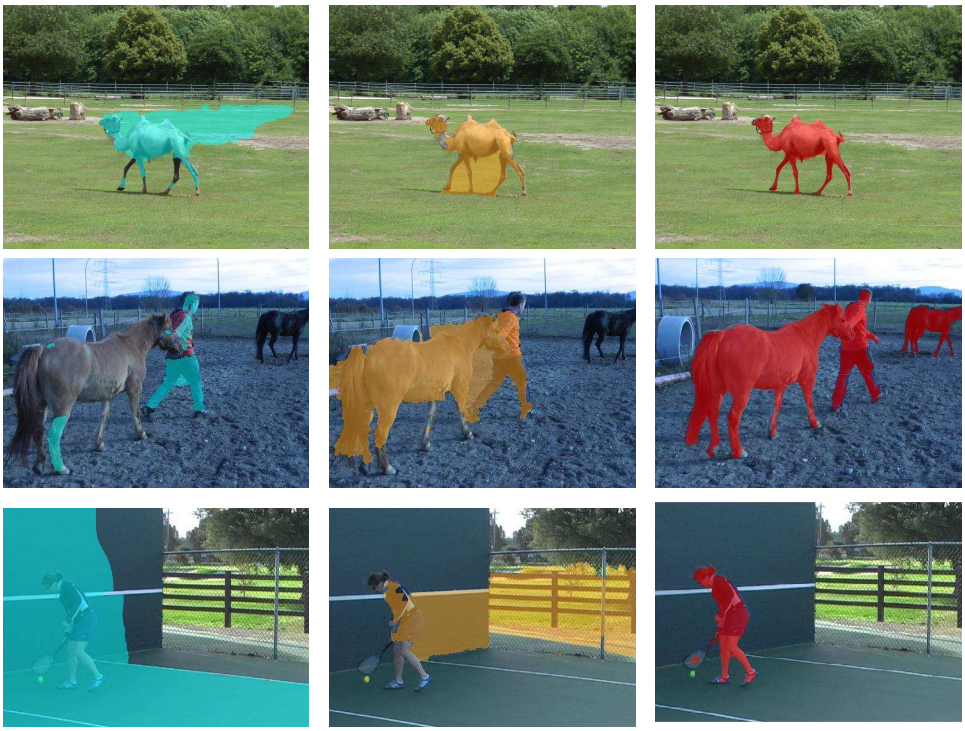
Net, FSG and our method segment them to varying extents. ARP focuses on the foreground object, but misses a part of it.

FBMS. As shown in Table 5.7, MP-Net [137] is outperformed by most of the methods on this dataset. Our approach based on visual memory outperforms MP-Net by 21.3% on the test set and by 21.0% on the training set according to the F-measure. FST [108] based post-processing ("MP-Net-V" in the table) significantly improves the results of MP-Net on FBMS, but it remains below our approach on all the measures. We compare with ARP [74] using masks provided by the authors on the test set. Our method outperforms ARP on this set by 12.2% on the F-measure. Overall, our method shows a significantly better performance than all the other approaches in terms of precision, recall and F-measure. This demonstrates that the visual memory module, in combination with a strong appearance representation, handles complex video segmentation scenarios, where objects move only in a fraction of the frames.

Figure 5.8 shows qualitative results of our method and the two next-best methods on FBMS: MP-Net-V [137] and CUT [68]. MP-Net-V relies highly on FST's [108] tracking capabilities, and thus demonstrates the same background leaking failure mode, as seen in all the three examples. CUT misses parts of objects and incorrectly assigns background regions to the

Measure	Set	KEY	MP-Net	FST	ARP	CVOS	CUT	MP-Net-V	Ours
$\mathcal{P}$	Training	64.9	83.0	71.3	-	79.2	86.6	69.3	89.9
	Test	62.3	84.0	76.3	76.1	83.4	83.1	81.4	93.8
$\mathcal{R}$	Training	52.7	54.2	70.6	-	79.0	80.3	80.8	83.5
	Test	56.0	49.4	63.3	66.9	67.9	71.5	73.9	75.3
$\mathcal{F}$	Training	58.2	65.6	71.0	-	79.3	83.4	74.6	86.6
	Test	59.0	62.2	69.2	71.3	74.9	76.8	77.5	83.5

Table 5.7 – Comparison to the state-of-the-art methods on FBMS with precision ($\mathcal{P}$), recall ($\mathcal{R}$), and F-measure ($\mathcal{F}$).



CUT MP-Net-V Ours

Figure 5.8 – Qualitative comparison with the top-performing methods on FBMS. Left to right: results of CUT [68], MP-Net-Video [137], and our method.

foreground in some cases, whereas our method demonstrates very high precision.

SegTrack-v2. The performance of our method on SegTrack is presented in the Table 5.8. NLC [35] is the top-performing method, followed by FSG [63], on this dataset. Note however, that these methods are both

Method	Mean IoU
CUT	47.8
FST	54.3
FSG	61.4
NLC	67.2
Ours	53.7
Ours w/o CRF	59.1

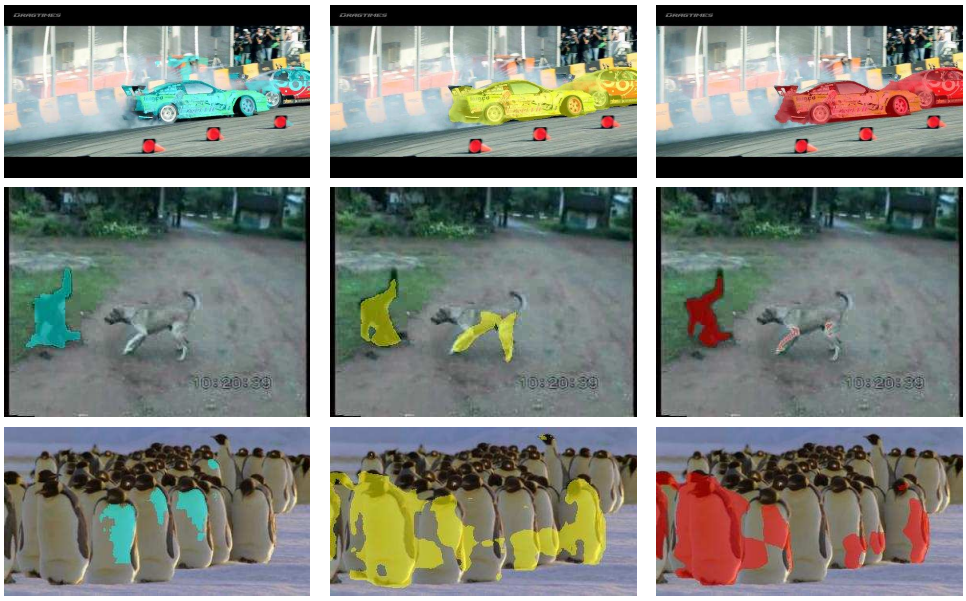
Table 5.8 – Comparison to the state-of-the-art methods on SegTrack-v2 with mean IoU.

tuned to SegTrack. FSG is trained directly on a subset of SegTrack sequences, and the parameters of NLC are set manually for this dataset. In contrast, our method is not adapted to the test domain, which is one of the reasons for the relatively lower performance than NLC and FSG. As shown recently [63, 72], the low resolution of some of the SegTrack videos poses a significant challenge for deep learning-based video segmentation methods. Being trained on datasets like PASCAL VOC or COCO, which are composed of high-quality images, these models suffer from the well-known domain shift problem, when applied to low-resolution videos. Our method, with its appearance stream trained on VOC, is subject to this issue as well. Additionally, CRF post-processing decreases the performance of our method on SegTrack; see 'Ours w/o CRF' in Table 5.8.

A qualitative comparison of our method and the variant without CRF post-processing ('Ours w/o CRF') with NLC is presented in Figure 5.9. In the first row, all the three approaches are segment the moving cars in the challenging racing scene, but NLC is less precise than the two variants of our method. In the second example, the monkey is extracted by all the methods, but only our method (w/o CRF) also segments the dog. The segmentation of the dog is however incomplete, and refinement with CRF worsens this further due to similarity with the background colours. In the last row, none of the methods captures the group of penguins. Our results are further diminished by the CRF, due to unreliability of the initial prediction (w/o CRF).

5.4.6 ConvGRU visualization

We present a visualization of the gate activity in our ConvGRU unit on two videos from the DAVIS validation set. We use the unidirectional model in the following for better clarity. The reset and update gates of the Con-



NLC Ours w/o CRF Ours

Figure 5.9 – Qualitative comparison of two variants of our method with the top-performing approach on SegTrack. Left to right: results of NLC [35], our method without CRF post-processing, and our full method.

vGRU, r_t and z_t respectively, are 3D matrices of $64 \times \mathrm{w}/8 \times \mathrm{h}/8$ dimension. The overall behavior of ConvGRU is determined by the interplay of these 128 components. We use a selection of the components of r_t and $(1 - z_t)$ to interpret the workings of the gates. Our analysis is shown on two frames which correspond to the middle of the *goat* and *dance-twirl* sequences in (a) and (b), respectively in Figure 5.10.

The outputs of the motion stream alone (left) and the final segmentation result (right) of the two examples are shown in the top row in the figure. The five rows below correspond to one of the 64 dimensions of r_t and $(1-z_t)$, with i denoting the dimension. These activations are shown as grayscale heat maps. High values for either of the two activations increases the influence of the previous state of a ConvGRU unit on the new state matrix computation. If both values are low, the state in the corresponding locations is rewritten with a new value; see equations ((5.3)) and ((5.4)).

For $i = 8$, we observe the update gate being selective based on the appearance information, i.e., it updates the state for foreground objects and duplicates it for the background. Note that motion does not play a role in this case. This can be seen in the example of stationary people (in the background) on the right, that are treated as foreground by the update

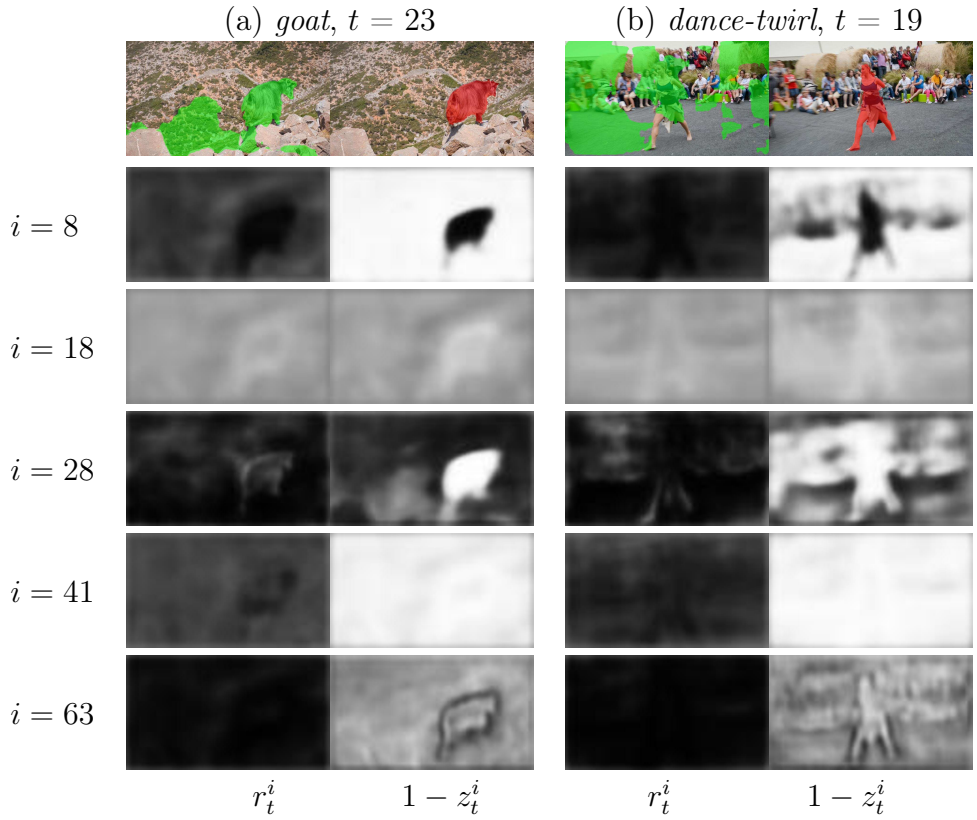


Figure 5.10 – Visualization of the ConvGRU gate activations for two sequences from the DAVIS validation set. The first row in each example shows the motion stream output and the final segmentation result. The other rows are the reset (r_t) and the inverse of the update ($1 - z_t$) gate activations for the corresponding ith dimension. These activations are shown as grayscale heat maps, where white denotes a high activation.

gate. In the second row, showing responses for $i = 18$, both heatmaps are uniformly close to 0.5, which implies that the new features for this dimension are obtained by combining the previous state and the input at time step t.

In the third row for $i = 28$, the update gate is driven by motion. It keeps the state for regions that are predicted as moving, and rewrites it for other regions in the frame. For the fourth row, where $i = 41$, r_t is uniformly close to 0, whereas $(1-z_t)$ is close to 1. As a result, the input is effectively ignored and the previous state is duplicated. In the last row showing $i = 63$, a more complex behavior can be observed, where the gates rewrite the memory for regions in object boundaries, and use both the previous state and the

current input for other regions in the frame.

5.5 Summary

This chapter introduces a novel approach for video object segmentation. Our method combines two complementary sources of information: appearance and motion, with a visual memory module, realized as a bidirectional convolutional gated recurrent unit. To separate object motion from camera motion we introduce a CNN-based model, which is trained using synthetic data to segment independently moving objects in a flow field. The ConvGRU module encodes spatio-temporal evolution of objects in a video based on a state-of-the-art appearance representation, and uses this encoding to improve motion segmentation. The effectiveness of our approach is validated on three benchmark datasets. We plan to explore instance-level video object segmentation as part of future work.

Part III

End-To-End Approach

Chapter 6

Integrating semantic and video segmentation

Contents

In this final part of the thesis we propose a joint architecture for learning video and semantic segmentation. It combines our weakly-supervised semantic segmentation approach (M-CNN) described in Chapter 3 with the video segmentation method from Chapter 5. The resulting fully-trainable architecture addresses some of the limitations of the heuristic-based M-CNN, leading to an improved performance.

Recall that M-CNN capitalized on weak semantic labels and motion cues to infer pixel-level labels of the video frames. These labels were then used to update a semantic model of the dominant object in the video. The updated model, in turn, helped to improve label estimation in the next iterations. This approach demonstrated that semantic and motion information can indeed benefit each other and that motion is an indispensable cue for weakly-supervised semantic segmentation. It was, however, limited

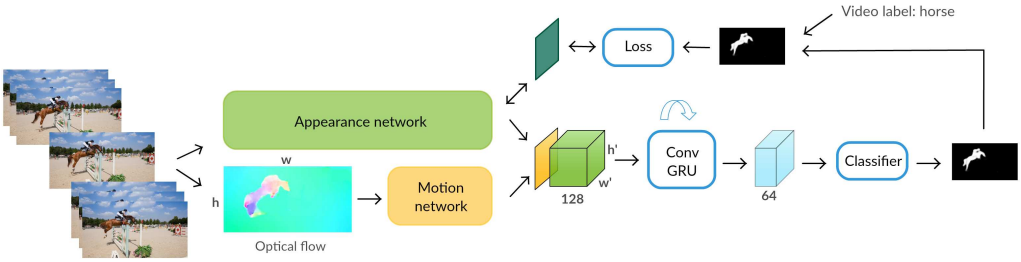


Figure 6.1 – Our joint framework for semantic and video object segmentation. Each video frame is processed by the appearance (green) and the motion (yellow) networks to produce an intermediate two-stream representation. The ConvGRU module combines this with the learned visual memory to compute the final segmentation result. This binary segmentation is converted to a semantic one by utilizing the weak label of a video sequence and used to train the appearance stream.

in that the moving object segmentation, as well as the label inference components of the framework were heuristic-based, thus their performance did not improve as learning progressed and could become a bottleneck of the method.

The second part of this manuscript addressed the problem of moving object segmentation in isolation. In particular, a video object segmentation method was developed in Chapter 5 that combined a frame-level motion segmentation and a semantic encoding of that frame with a convolutional recurrent model, which aggregated a spatio-temporal representation of the moving object. This fully trainable method has set the new state-of-the-art performance in unsupervised video object segmentation on DAVIS and FBMS.

To combine the two approaches described above we augment our video object segmentation model with a semantic segmentation layer on top of the appearance stream and use the predicted moving object masks to train it (see Figure 6.1). A large dataset of weakly labeled videos is used in this stage. Notice that training the appearance stream results in a change in the feature encoding used as input to the memory module (shown in green in Figure 6.1). This transformation of the feature space quickly leads to a degradation of the video segmentation performance, as we demonstrate in Section 6.2. To adapt the memory module to the changing feature space of the appearance stream, we finetune the ConvGRU on DAVIS in a fully-supervised way. As a result, our approach jointly learns semantic and video object segmentation models with only a small set of fully-labeled and a large set of weakly-labeled videos.

Several methods have recently demonstrated the value of class activation maps (CAMs) [162] for weakly-supervised learning (see [57, 75, 106]). They provide approximate object locations that are category-specific, and thus are indispensable for inferring pixel-level labels. We show that CAMs can be seamlessly integrated into our approach as well and result in a significantly improved performance.

The rest of this chapter is organized as follows: in Section 6.1 we describe our joint approach for learning semantic and video-object segmentation, as well as demonstrate how CAMs can be integrated into the framework. Then, in Section 6.2 we outline the experimental protocol and present the results of the joint learning approach on both weakly-supervised semantic segmentation and video object segmentation. We also provide an ablation study to investigate the different design choices that were made and compare to several baselines. Finally, in Section 6.3 we conclude and provide potential directions for future work.

6.1 Joint learning of semantics and motion

Our model for joint learning of semantic and video object segmentation, shown in Figure 6.1, is an extension of the video object segmentation architecture presented in Chapter 5. In particular, the model takes video frames together with their estimated optical flow as input, and outputs a binary segmentation of moving objects. This binary segmentation is then converted to a semantic segmentation by utilizing the weak label of the video. The inferred semantic masks are, in turn, used to train the appearance stream, which is augmented with a segmentation layer shown in dark green. To adapt the memory module to the constantly changing semantic space, ConvGRU is also jointly trained on DAVIS. Next we describe this approach in more detail.

6.1.1 Learning framework

We train our model on two sets of videos: a small fully supervised dataset $\mathcal{V} = \{(\mathbf{x}_i^{\mathcal{V}}, \mathbf{y}_i^{\mathcal{V}})\}_{i=1}^{N}$ for video object segmentation and a large weakly-labeled dataset $\mathcal{S} = \{(\mathbf{x}_i^{\mathcal{S}}, \mathbf{z}_i^{\mathcal{S}})\}_{i=1}^{M}$ for semantic segmentation, where $\mathbf{x}_i^{\mathcal{V}}$ and $\mathbf{x}_i^{\mathcal{S}}$ are sets of frames corresponding to video sequences, $\mathbf{y}_i^{\mathcal{V}}$ represent binary segmentations of moving objects and $\mathbf{z}_i^{\mathcal{S}}$ are weak semantic labels indicating presence of objects of certain categories in the video. Our main goal is to learn a semantic segmentation model from the set $\mathcal{S}$ using only the weak labels $\mathbf{z}_i^{\mathcal{S}}$ and motion cues. Unlike our previous weakly-supervised semantic

segmentation method described in Chapter 3, we propose to unify the label inference and video segmentation steps. We integrate the semantic model into our moving object segmentation approach as an appearance stream, thus training it should directly results in an improved label estimation. To update the appearance model we treat the predicted moving object masks $\mathbf{m}_i^{\mathcal{S}}$ as ground truth segmentations. Let's assume that a video $\mathbf{x}_i^{\mathcal{S}}$ contains only objects of a single semantic category c and that all of these objects exhibit independent motion. An estimate of the semantic labels $\hat{\mathbf{y}}_i^{\mathcal{S}}$ can then be obtained in a straightforward way:

$$\hat{y}_{ij}^{\mathcal{S}} = \begin{cases} c, \text{if } m_{ij}^{\mathcal{S}} = 1, \\ 0, \text{otherwise}, \end{cases} \tag{6.1}$$

and a semantic segmentation model $P(\mathbf{y}|\mathbf{x}, \theta)$ can be learned by maximizing the parameters of the appearance stream θ with respect to the estimated labels:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P(\hat{\mathbf{y}}^{\mathcal{S}}|\mathbf{x}^{\mathcal{S}}, \theta). \tag{6.2}$$

In practice this global maximization step is approximated with stochastic gradient descent.

As mentioned above, improved appearance representation should result in an improved performance of the video segmentation approach. Indeed, it was shown in the ablation study in Section 5.4.4, that the quality of the appearance representation is directly related to the segmentation quality. Motivated by this observation, similarly to M-CNN, we switch between updating the semantic model and using the updated model to re-estimate moving object segmentations. Unlike M-CNN, our new model integrates the semantic network more intimately into the video segmentation and thus requires a different training protocol.

Recall that in M-CNN the scores $\mathbf{s}_i^c$ for the category c predicted by the appearance network were combined with the motion cues to infer the segmentation masks. In contrast, our proposed approach extracts a feature encoding of a frame $\phi(f)$ from an intermediate layer of the appearance network and feeds it to the memory module. This allows the model to not only reason about the location of the object in the current frame, but also to aggregate a spatio-temporal representation of the object throughout the video, which is crucial for the success of the approach. Such a tight coupling, however, comes at a cost: training the appearance stream in a weakly-supervised way results in a change in the feature representation $\phi(f)$. The updated feature representation, in turn, changes the behavior of the memory module and it does not necessarily produce accurate video

object segmentation anymore. This can be seen as an instance of the domain shift problem. A common way to adapt a model to the changed inputs is to finetune it on the new data. We follow this path and interleave between training the appearance stream on $\mathcal{S}$ and finetuning the memory module on $\mathcal{V}$ - a small dataset with ground truth moving object annotations.

In practice we first train our video segmentation method on DAVIS in the same way as described in Chapter 5, with appearance stream pretrained on ImageNet. We then augment the model with a semantic segmentation layer on top of the appearance stream and interleave between k iteration of weakly-supervised semantic segmentation training and k iterations of training the video segmentation model in a fully-supervised way. In Section 6.2 we demonstrate that this joint training is essential to avoid degenerate solutions.

6.1.2 Incorporating CAMs into training

Class activation maps are approximate estimates of location of an object of a given category in an image that can be computed by a model trained for image classification (see Section 2.4 for details). They have been successfully exploited by several weakly-supervised semantic segmentation approaches recently [57, 75, 106], allowing them to achieve state-of-the-art results. In this section we show that our video-based method can benefit from CAMs as well.

Indeed, knowing the approximate location of an object in a video can be useful when the object does not exhibit independent motion for a large number of frames. Our video segmentation approach is based on a recurrent memory module, whose state is decayed in every time step by the forget gate (see Section 5.3 for details). Due to this update rule, if an object becomes static the model starts to gradually forget it and eventually segments it as background. Class activations maps, despite being inaccurate can help to better handle such scenarios.

Following [106] we train a slightly modified version of VGG model on the trainval set of PASCAL VOC using only image-level labels. The network is modified by removing the last max pooling layer to increase its output resolution by a factor of 2, which was show to be important for the localization performance of CAMs. We then precompute the maps $\mathbf{u}_i$ for all the videos in $\mathcal{S}$, where a map for a frame j in a video i is encoded as $\mathbf{u}_{ij} = \{cam_c\}_{c=1}^{K}$ and cam_c is the corresponding activation map for the category c. These maps are added to the training set $\mathcal{S} = \{(\mathbf{x}_i^{\mathcal{S}}, \mathbf{z}_i^{\mathcal{S}}, \mathbf{u}_i^{\mathcal{S}})\}_{i=1}^{M}$ and used in the label inference as described below. Notice that most of the approaches employing CAMs turn them into binary maps by thresholding category scores

with a certain predefined value τ. This is due to the fact that activation maps are used as seeds in these methods and thus it is enough to limit them to small confident regions. In our approach, in contrast, we treat CAMs as probabilities and thus need to convert them to continuous values in $[0, 1]$ range. To this end, we apply the sigmoid function to each class map individually to obtain $cam_c = sigmoid(\mathbf{s}_c)$, where $\mathbf{s}_c$ is the raw output of the network.

To integrate CAMs into our video segmentation approach we propose to treat them in the same way as motion segmentations shown in yellow in Figure 6.1. Indeed, both the motion segmentation and class activation maps are in the $[0, 1]$ range, where high values indicate the location of the object. We compute the new inputs to the memory module with

$$\omega' = \lambda\omega + (1 - \lambda)cam_c, \tag{6.3}$$

where ω is the predicted motion segmentation, ω' is the final input to the memory module and c is the category of the video. In practice we simply set $\lambda = 0.5$, giving equal importance to the motion and category cues. Notice that we use CAMs only during inference of a segmentation for weakly labeled videos. In training of the video segmentation component no semantic information about the videos is available, thus we set $\omega' = \omega$. Nevertheless, the learned model can still exploit class localization information in testing, since both cues are encoded in the same form and have a similar meaning.

6.2 Experiments

6.2.1 Implementation details

Datasets and evaluation We use DAVIS [113] as a fully-supervised video segmentation dataset. We follow the experimental protocol described in Section 5.4.1 both in training and evaluation of the video segmentation performance. For learning weakly supervised semantic segmentation we utilize two datasets: YouTube-Objects [116] and ImageNet-VID [1] to demonstrate that our approach is independent of the particular set of videos. As was noted in Section 3.2.1, a fraction of sequences in these datasets either do not contain the objects indicated in the video labels or these objects are static, thus, some form of data preprocessing is necessary. We explore two possibilities: first, like in the M-CNN experiments, we filter the videos based on the area of the predicted motion segment (see Section 3.2.1 for details). More recently, [57] proposed to use CAMs to filter out the sequences which do not contain the object of interest. This approach results

in a high precision, but does not handle static videos. Using CAM maps in training, however, as we proposed in Section 6.1.2 can allow to segment static objects as well (although the segmentations are less accurate). We compare the two filtering strategies in Section 6.2.3. To evaluate the semantic segmentation performance we utilize the validation set of PASCAL VOC 2012 [34]. The details of all three datasets and the metrics used for evaluation are described in Section 3.2.1.

Model In Section 5.4 we have thoroughly evaluated different design choices in our video segmentation framework and concluded that the configuration with the appearance stream based on DeepLab-v2 and motion stream finetuned on FusionSeg with FlowNet2 optical flow performs best. Almost all the other methods for weakly-supervised semantic segmentation report their results with the DeepLab-v1 architecture, however. Hence, to fairly compare to these methods, we use the configuration with appearance stream based on DeepLab-v1 (trained on ImageNet) and motion stream finetuned on FusionSeg with FlowNet2 optical flow in all our experiments.

Training We start by pretraining our video segmentation model on DAVIS, as described in Section 5.3.2. To finetune it jointly for the tasks of semantic and video segmentation then, we interleave between 10 steps of weakly-supervised semantic segmentation training and 10 fully-supervised steps for video object segmentation for a total of 20000 iterations. For weakly-supervised iterations we use a batch size 10, with frames equally sampled from a video sequence. We use SGD with a learning rate of 0.001, momentum 0.9 and weight decay of 0.0005. Learning rate is decreased in every iteration with a "poly" policy with $power = 0.9$. For video segmentation iterations we use batches of 12 consecutive frames. We also include 20% of batches with simulated discontinuous motion, as suggested in Section 5.3.2. The optimization algorithm used is RMSProp with learning rate 0.0001, 0.9 momentum and weight decay set to 0.005. As in the experiments in Chapter 5, the learning rate is decreased after each epoch and the gradients are clipped to the $[-50, 50]$ range before each parameter update, to avoid numerical issues [45]. In both cases we use random flipping and spatio-temporal cropping for data augmentation.

6.2.2 Ablation study

We start by evaluating the importance of the different components of our label inference approach in Table 6.1. To this end, we compare several

Dataset	Inference	Joint train	Filtering	CAM	Mean IoU
YTube	no	no	motion 50	no	19.3
YTube	yes	no	motion 50	no	7.2
YTube	yes	yes	motion 50	no	37.5
YTube	yes	yes	motion 10	no	39.1
YTube	yes	yes	CAM 10	yes	47.7
ImNet	yes	yes	CAM 10	yes	47.0

Table 6.1 – Evaluating different variants of our label inference approach on the 10 classes of PASCAL VOC 2012.

versions of our method on the 10 classes of the validation set of PASCAL VOC 2012 (the same ones as used in the M-CNN experiments). The first variant simply uses our video segmentation approach trained on DAVIS to infer the moving object segmentations for all the videos in YouTubeObjects dataset. The videos are filtered based on the size of the predicted motion segment and only subsequences with at least 50 consecutive valid frames are used for training (denoted by 'motion 50' in the Filtering column). These segmentations are then fixed (denoted by 'no' in the Inference column) and used together with the semantic tags of the videos to train the appearance stream in a fully-supervised way. We observe that using fixed motion labels leads to a very low performance on VOC. We conclude that this is due to a low performance of our video segmentation method, with its appearance stream trained on ImageNet, on the challenging videos in the Youtube-Objects dataset. These videos are of lower quality than those in DAVIS and, in addition, contain irrelevant sequences. As a result, the quality of the labels estimated by the unadapted video segmentation method is too low to be useful.

Next, we experiment with employing our visual memory module to segment the moving objects in a video in every training iteration (denoted with 'yes' in the Inference column). In this variant, shown in the second row of the table, the improved appearance representation can result in improved label estimates. This baseline, however, does not utilize joint training, that is the parameters of the memory module are not jointly finetuned on DAVIS to adapt to the changing appearance representation. As a result, our video segmentation approach quickly breaks down, and resorts to predicting all the pixels in a video as background. The semantic segmentation model then learns the background category only as well. In contrast, our joint training approach (row three, denoted with 'yes' in the Joint train column) is able

to successfully adapt the memory module as the appearance representation progresses, which not only leads to non-degenerate segmentations, but also substantially outperforms the static labels baseline from the first row. The importance of an appearance model trained on the task of semantic segmentation for the performance of our video segmentation approach has been already demonstrated in Section 5.4.4. Here, in addition, the appearance stream gets tuned to the specific videos that are being segmented, resulting in a further improvement in precision.

In M-CNN, as well as in these ablation experiments, only sequences of at least 50 frames were used for training. Batches of 10 frames were then sampled from them at equal distance to increase diversity. We now experiment with including shorter sequences and forming batches of 10 consecutive frames in the fourth row of Table 6.1. Intuitively, this will decrease diversity within batches but will provide additional training examples, as well as improve video segmentation quality for sequences with rapid motions. Indeed, we observe a 1.6% improvement over the variant with sequences of length at least 50. Based on this observation we relax the sequence length constraint to 10 in the remaining experiments.

Next, we evaluate the importance of class activation maps, introduced in Section 6.1.2, in the row five of Table 6.1. We use CAMs both for filtering the videos to avoid irrelevant sequences, and to aid in label inference in cases where the objects are static, or there are several moving objects in a video, some of which are irrelevant. In line with some recent publications [57, 75, 106], integrating CAMs indeed results in a significant improvement of our method's performance. Despite being trained with no semantic information, our video segmentation approach is able to capitalize on soft semantic cues at test time, demonstrating its flexibility.

Finally, to show that our method is general and can be applied to different video collections, we train the final variant (with joint training and class activation maps) on the ImageNet-VID dataset (denoted as ImNet). This datasets has different statistics compared to Youtube-Objects, nevertheless our method is able to achieve a similar performance. In the next sections we compare the performance of our joint approach to the state-of-the-art methods in weakly-supervised semantic segmentation.

6.2.3 Evaluation on weakly-supervised semantic segmentation

In this section we compare our full method to M-CNN and the top-perfroming weakly-supervised approaches on the 10 categories of the vali-

dation set of Pascal VOC 2012 dataset. In addition to EM-Adapt [107], we include two more recent publications: an image-based method of Kolesnikov and Lampert [75] and a video-based method of Hong et al. [57]. Finally, we report the performance of the DeepLab-v1 model trained on the augmented training set of PASCAL VOC in a fully-supervised way for a reference. The results are presented in Table 6.2.

First of all, we observe that our fully-learnable method significantly outperforms M-CNN when trained on videos only, both on Youtube-Objects and ImageNet-VID datasets. This is due to the joint training approach, which allowed to achieve higher quality of estimated labels, compared to the heuristic-based approach used by M-CNN. Class activation maps also served as an important cue for label inference. This is confirmed by the fact that our models, trained on videos only, perform on par with M-CNN trained on videos and images from VOC. Indeed, our approach also utilized VOC images to learn a CAM model, but not to directly train the semantic segmentation network. Incorporating images into training in a principled way could further boost our results. This can be seen by comparing our results to those of the heuristic-based approach of Kolesnikov and Lampert [75]. They directly infer the labels of the images in the training set of VOC, thus avoiding the domain shift problem.

The very recent video-based method of Hong et al. [57] achieves top performance on VOC. They propose to utilize a specialized network architecture with independent classification and segmentation networks. The segmentation network is category-agnostic, and is learned on a large set of web-crawled videos, whereas the classification network is trained on VOC. This allows them to achieve high recall and avoid the domain shift issue. The specialized network architecture is indeed a critical component of their approach, making it non-trivial to apply to state-of-the-art models like ResNet [54]. In addition, although they do utilize motion during label inference on videos, it has a relatively low weight in the objective and the method mostly relies on CAMs. As a result, it remains heuristic-based, and could further benefit from a more principled way of combining label inference in images and videos.

Method	Dataset	bkg	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Average
EM-Adapt [107]	VOC aug.	77.4	32.1	30.8	26.4	42.6	40.7	32.8	37.8	35.1	45.2	41.1	40.2
Kolesnikov [75]	VOC aug.	82.4	62.9	61.6	27.6	62.7	75.2	53.5	65.8	57.8	62.3	45.4	59.7
M-CNN	YTube	86.3	46.5	43.5	27.6	34.0	47.5	28.7	31.0	30.8	32.4	43.4	41.2†
M-CNN	VOC aug.+YTube	82.5	47.8	35.3	29.6	45.6	54.6	40.3	46.6	44.8	52.2	56.6	48.7
M-CNN	ImNet	85.6	41.4	45.3	23.2	38.6	42.3	36.0	35.1	21.1	15.3	44.8	39.0
M-CNN	VOC aug.+ImNet	83.1	47.6	40.3	26.4	44.1	51.1	41.7	51.0	34.9	44.6	52.7	47.0
Hong [57]	VOC aug.+Crawl	87.0	69.3	70.2	31.2	68.5	76.5	63.8	73.5	69.5	66.5	57.4	66.7
Ours joint	YTube	87.3	49.4	39.2	32.4	59.4	49.5	41.1	42.8	32.8	47.7	43.3	47.7
Ours joint	ImNet	87.0	45.5	42.2	36.4	54.7	46.0	37.0	45.4	26.2	45.2	50.9	47.0
DeepLab-v1	VOC aug.	88.9	76.0	74.9	60.0	78.2	78.2	63.9	72.4	61.7	72.1	77.0	73.0

Table 6.2 – Performance of our joint learning method on the VOC 2012 validation set is shown as IoU scores. MCNN is our previous, heuristic-based approach described in Chapter 3. We also compare with the best variants of EM-Adapt [107] trained on YouTube-Objects (YTube), ImageNet-VID (ImNet), and augmented VOC (VOC aug.) datasets, as well as to more recent image-based [75] and video-based weakly-supervised approaches [57] and the fully-supervised DeepLab-v1 [23]. † denotes the average result of 5 trained models.

In Figure 6.2 we present a qualitative comparison of our fully-trainable, weakly-supervised segmentation approach to M-CNN and the method of Hong et al. [57] on the validation set of Pascal VOC 2012. Note that the latter two methods are trained on videos and images, whereas ours is trained on videos only. First of all, due to the principled way to learn video segmentation and weakly-supervised semantic segmentation models together, our approach produces more accurate predictions compared to M-CNN. This is especially noticeable in the bird example in the first row, where M-CNN incorrectly labels the feeder as bird, but our method is able to avoid this mistake. Indeed, the motion cues used in M-CNN are fixed and can result in noisy labels for learning the semantic model, whereas our method is able to estimate labels more accurately. The segmentations of [57] are even more precise, due to their task-specific architecture, where the segmentation branch is built with deconvolutional layers, reconstructing the fine details of the objects. This architecture is inefficient, however, taking twice as much memory compared to the sate-of-the-art models. In addition, it requires a separate network pass for each object in an image in contrast to FCNNs which segment the whole image in a single pass.

6.3 Summary

In this chapter we have integrated our motion-based, weakly-supervised semantic segmentation approach from Chapter 3 with our learning-based video segmentation approach described in Chapter 5. The resulting architecture was jointly trained for the two tasks, demonstrating an improved performance, compared to the heuristic-based M-CNN on the PASCAL VOC 2012 benchmark dataset. It, however, remained behind some of the more recent weakly-supervised methods.

Despite the impressive progress achieved by the field, the gap with the fully-supervised methods is not fully bridged yet, and the top-performing methods have important limitation. In the next chapter we review some of the relevant challenges and propose ideas that could help to address them.

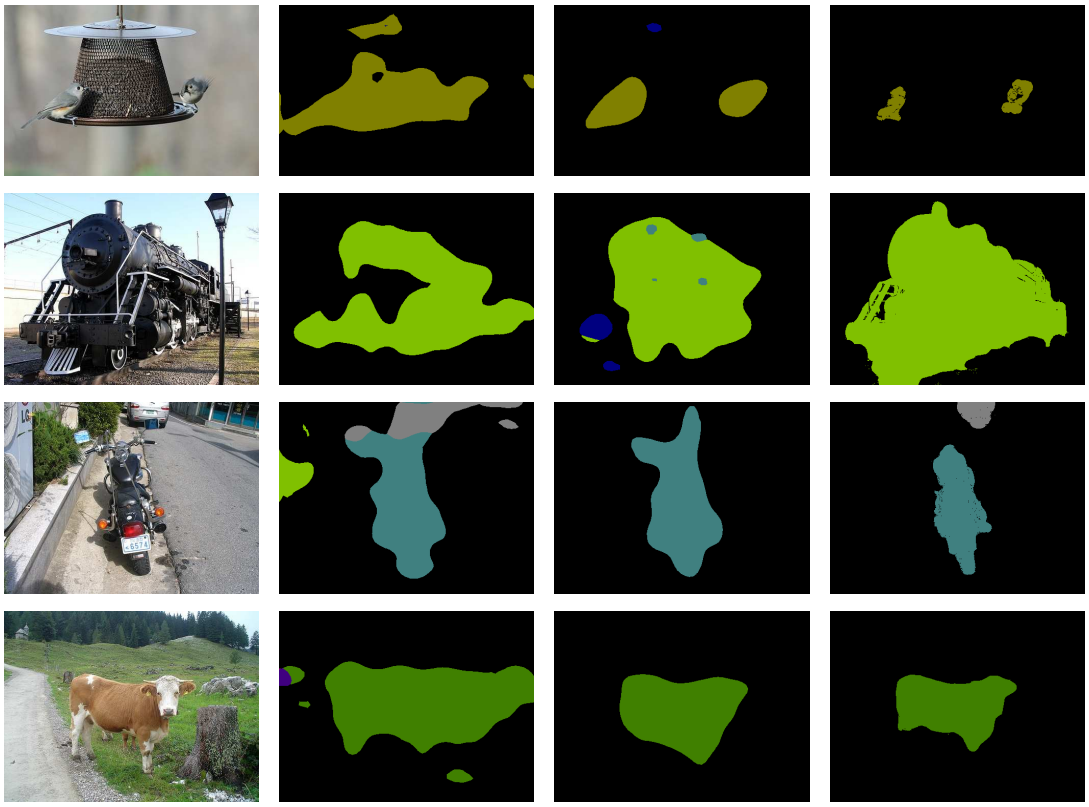


Image M-CNN (Im+Vid) Ours joint (Vid) [57] (Im+Vid)

Figure 6.2 – Sample results on the VOC 2012 validation set. Results of M-CNN (our previous method described in Chapter 3) trained on augmented VOC and Youtube-Objects, our joint approach trained only on Youtube-Objects, and the method of Hong et al. [57] trained on augmented VOC and a dataset of web-crawled videos are shown in 2nd, 3rd and 4th columns respectively.

Chapter 7

Conclusion

Contents

In this chapter we summarize our contributions on weakly-supervised semantic segmentation and video segmentation in Section 7.1. We then conclude with directions for future research in Section 7.2.

7.1 Summary of contributions

7.1.1 Weakly-supervised semantic segmentation

We have introduced a method for learning deep semantic segmentation models from weakly-labeled videos, capitalizing on motion cues (see Chapter 3). Our model, motion-CNN (M-CNN), integrates a fully-convolutional network (FCNN) with motion information into an expectation maximization-like framework. This framework allows to estimate accurate pixel-level labels for training the FCNN from videos by combining unsupervised motion segmentations with predictions of the semantic model. Our approach achieved state-of-the-art results on PASCAL VOC at the time of publication, while being trained from only 150 videos and 1462 images with tag-level labels.

With the deep-learning-based method for video object segmentation summarized in the next section, we have addressed one of the main limitation of M-CNN in Chapter 6. In particular, the moving object segmentation and label inference algorithms used in M-CNN are heuristic-based, and thus their performance does not improve during training. In our fully-trainable method for weakly-supervised learning from videos we have extended the video segmentation model from Chapter 5 with a semantic segmentation branch. The resulting network was trained on a large set of videos with weak semantic labels and a small set of videos with pixel-level moving object labels, allowing to incrementally improve performance on both tasks. This fully-trainable architecture indeed achieved a significant improvement over the heuristic-based M-CNN.

7.1.2 Video object segmentation

Motivated by the importance of motion information for weakly-supervised semantic segmentation, we studied the problems of motion and video object segmentation, developing one of the first deep-learning based approaches to these problems (see Chapter 5). Our model for video object segmentation is a two-stream network with an appearance stream and a motion stream. Their outputs are combined, and passed to the visual memory module, which is trained to aggregate a spatio-temporal representation of the moving object. This architecture allowed our model to produce accurate segmentations and handle discontinuous motion.

Our motion segmentation network is a CNN trained for the task of segmenting the objects that move independently from the camera in a fully-supervised way. With this learning-based approach to motion segmentation we demonstrated that motion patterns in the optical flow field can be treated in the same way as appearance patterns in RGB images. We believe that this insight has opened the way for learning-based solutions to other motion-analysis problems, such as camera pose estimation. Our full method has set the new state-of-the-art for video object segmentation on several benchmark datasets, significantly outperforming the classical, heuristic-based, as well as concurrent, deep-learning-based approaches.

Frame t Frame $t+1$ Optical flow

Figure 7.1 – An example of a video from the FT3D dataset with strong camera motion. Despite the fact that the objects in the background are static, their shapes are captured precisely by optical flow. This information could be used to learn accurate segmentation models for static categories.

7.2 Perspectives for future research

7.2.1 Weakly-supervised semantic segmentation

Handling static objects. In this dissertation we have proposed a method that utilized object motion to learn accurate semantic segmentation models from videos. Our approach, however, relied on an assumption that the object exhibits independent motion in at least one frame of the video. While this assumption generally holds for categories like *car* or *dog*, it does not for other types of objects like *table* or *bottle*. Moreover, a video of a moving person can have a static car in the background, introducing noise in the label estimation. Thus, clearly, motion cues alone are not sufficient for learning to segment all the categories of the visual world. In our experiments we utilized an heuristic approach to infer pixel labels for static categories. It allowed us to achieve state-of-the-art performance at the time of publication, but inevitably resulted in an overall loss of precision.

Recent work of Hong et al. [57] extended our M-CNN for learning semantic segmentation models from videos with both dynamic and static objects. Unlike our approach, which learns a single network for semantic segmentation, they propose to learn separate models for approximate object localization and precise segmentation. The segmentation network is category-agnostic, which allows them to transfer knowledge from moving categories to the static ones. This problem-specific architecture, however makes applying their approach to state-of-the-art networks like ResNet non-trivial. Their label inference algorithm is the same as in our M-CNN, but with an additional unary term that encodes the class activation scores of Zhou et al. [162], and a decreased weight of the motion term. Thus, despite achieving top results on PASCAL VOC 2012, the method of Hong et al. remains heuristic-based, and requires careful hyper-parameter tuning. An

alternative could be utilizing ego-motion [7] to obtain information about shape of objects in a scene irrespective of their dynamics. For examples, Figure 7.1 shows a sequence from the FT3D dataset with a moving camera and two dynamic foreground objects in the top right and bottom left corners. The objects in the background are static, but strong ego-motion allows to reveal their shapes nevertheless.

Incorporating other physical constraints. We drew inspiration for our motion-based approach from the seminal work of Elizabeth Spelke [130], which showed that independent object motion is indeed the primary cue used by infants to learn the notion of objects. In her research, however, other principles were identified, that are as important for object perception. These principles can be described as physics-based in that they correspond to the basic constraints for rigid bodies in the physical world. In particular, they include constraints such as *boundedness*: two distinct objects can not occupy the same place at the same time; or *no action at distance*: two surfaces that are visually separated constitute independent objects even if their motion is consistent. Incorporating such constraints into weakly-supervised learning methods, explicitly or implicitly, could help to resolve remaining ambiguities in the label inference. For instance, a method that respects the *no action at distance* constraint, would be able to better handle multi-object videos. The first step towards achieving such capabilities is to design a label inference algorithm that operates on the level of surfaces, not pixels, where surfaces can be extracted with a depth estimation approach [94].

Weakly-supervised instance segmentation. Instance segmentation can be seen as the most general setting for object recognition. It combines the challenges of object detection and semantic segmentation, as it requires the model to precisely segment each object instance in an image. As in other areas of computer vision, the performance on this task improved significantly in recent years due to the introduction of deep learning based methods (see for instance [53, 90]). These methods, however, remain fully-supervised. Weakly-supervised instance segmentation, especially in the case when image-level labels only are available, is an extremely challenging problem. It exhibits all the issues of weakly-supervised semantic segmentation, described in Section 1.1.1, and in addition requires the label inference to distinguish between different instances of the same category. Motion cues can be helpful here as well. In particular, distinct motion directions can serve as an indicator that certain pixels belong to distinct objects instances even when their appearance is consistent [13, 101]. Other physical constraints mentioned above can be also useful for this setting. For instance, incorporating the *no action at distance* constraint can help to distinguish between objects that exhibit similar motion patterns.

Data-efficient learning. Reducing the amount of annotations required for learning segmentation models is a necessary but not a sufficient condition to capture all the categories in the visual world. For many categories a large number of image or, especially, video examples is hard to obtain in the first place – a problem commonly referred to as long tail category distribution [142, 164]. It implies that for a small number of categories many training examples can be easily obtained, whereas for a much larger number of categories (so called *long tail* of the distribution) training examples are sparse. To handle the categories in the tail the model has to be data efficient, that is, it has to be able to build a robust semantic model of a category from just a few examples. This property is often achieved by sharing information between categories. For instance, Wang et al. [147] pose this problem as transfer learning. They use data-rich categories in the head of the distribution to train a deep network that adapts the weights of a classifier learned from a small number of samples. This network captures what makes a classifier generalize better, and attempts to transfer this knowledge to the categories in the tail of the distribution. In contrast, Zhao et al. [158] propose to explicitly utilize semantic relations between concepts, by embedding image representations in a space that respects the wordnet hierarchy. As a result, images that are semantically similar are closer in the feature space, facilitating information sharing between the categories. Learning representations that are disentangled by design and thus encourage information sharing [161] is another promising direction. Sparsity-inducing regularizers, such as L1-loss, are traditionally used to enforce disentanglement [96], but other forms of regularization can be explored as well.

7.2.2 Video object segmentation

Optical flow-free solutions. All the state-of-the-art methods for motion segmentation and unsupervised video object segmentation use optical flow to extract raw motion information from pairs of RGB frames. This preprocessing step simplifies the more abstract task of estimating independent object motion. In addition, for learning-based approaches like our MP-Net, it provides a natural separation of the motion estimation problem from appearance. This separation allowed us to train a model on unrealistic, synthetic videos and then apply it in real scenarios with only a moderate loss in performance. Optical flow computation is expensive, however, and its complexity might be unnecessary for the task of motion segmentation. Indeed, optical flow strives to estimate the exact translation of each pixel between two consecutive frames, whereas to classify whether a pixel has

moved with respect to the background a rough approximation would be sufficient. This has been demonstrated recently by Dave et al. [29] for the problem of action recognition, where optical flow is traditionally used for data preprocessing as well. They instead propose to utilize a series of recurrent neural networks that operate on RGB frames and sequentially make top-down predictions about the future. The predictions are then corrected with bottom-up observations and the resulting residual term is used for action classification. This approach allowed their model to achieve state-of-the-art performance on several benchmark datasets without expensive optical flow computation. Residual video representations naturally focus on the moving regions, and can be seen as an approximation of motion segmentation. Thus, they could be used to design efficient, optical flow-free video segmentation methods.

Video instance segmentation. As in the case of image recognition, instance segmentation is desirable and challenging for video segmentation. Traditional approaches for this problem rely on trajectory clustering [18, 68] and have shown state-of-the-art performance in the past. They, however, have been recently outperformed by our learning-based method in the foreground/background setting. Extending our approach to the instance segmentation scenario is yet to be explored. The main limitation here is that state-of-the-art deep-learning methods for instance segmentation on images [53, 90] are computationally expensive, relying on object detection pipelines. They produce hundreds of proposals for each frame and classify them individually, thus, applying these methods directly to videos is not practical. In addition, image-based approaches do not exploit the temporal consistency property of the videos. Romera-Paredes et al. [121] have proposed a recurrent memory-based method for instance segmentation recently, which is more suitable for the video problems. Their method produces a single output for every object instance and can memorize the objects detected in the past, to simplify their segmentation in the future frames. Thus, it would be natural to extend our approach to the instance-level setting by replacing the ConvGRU trained for foreground/background segmentation with the memory module of Romera-Paredes et al. The model can then be trained on a video dataset with instance-level annotations, like FBMS [105].

Interactive video object segmentation. Unsupervised video object segmentation is limited by the fact that it captures all the moving objects. This might be desirable in some cases, but in many scenarios the user might want to focus on a subset of the salient video objects. Consider, for instance, the *breakdance* video from the DAVIS dataset, shown in Figure 7.2. Both the dancer and some of the people in the background are in motion, and thus segmented by our method. This, however, contradicts with ground truth

Ground truth Ours full

Figure 7.2 – A video from the DAVIS dataset, where several people are in motion. They are all segmented by our method, shown on the right, to some extent. The ground truth annotation, however, focuses on the dancer. Introducing some form of interaction with the user into our method could help to resolve such ambiguous cases.

annotations in the dataset. One way of addressing this limitation is by asking the user to mark the relevant objects in advance, but it is preferable to allow him to select the desired instances from all the segmented ones instead. Our approach does not provide any interaction with the user at the moment. To implement such a functionality, the notion of instances has to be introduced first. This will allow the user to provide binary tags for all the segmented instances, indicating his preference. The model will then re-segment the video using the selected instances for guidance. This can be achieved by integrating our method with one of the recently published semi-supervised approaches [70, 80, 88]. They are often based on finetuning the network on test sequences, however, which is computationally expensive. A more efficient solution could rely on a permanent memory module, described in the next paragraph.

Long term memory. One of the main limitations of our method is the short time span of the recurrent memory module. In particular, if the object stops moving, like in the case of the lion in Figure 7.3, the model can still segment it for several seconds, but, if it does not resume motion, the memory state decays and the object gets lost. This is in part due to the short sequences on which the model is trained, due to memory limitations, and in part an inherent property of GRUs. The memory state gets multiplied by the forget gate at every time step, its values being less than one, thus the state inevitably decays in time. Memory networks [78, 132] are commonly used when a permanent, fully-differentiable information storage is needed. They feature an explicit storage unit which is updated with the new observation at every time step and older observation are only erased

Frame t Frame $t + 20$ Frame $t + 40$ Frame $t + 60$

Figure 7.3 – An example of a video from the FBMS dataset where the lion remains mostly static for several frames. The state of our visual memory module is decayed at every time step, thus by the end of the sequence only the parts of the animal that moved (head, paws and tail) remain segmented, but the body is lost. Integrating a memory network into our architecture could help to handle such scenarios.

if the memory capacity is exceeded. Both memory update and read operations are implemented through linear and pooling layers, thus the desired behavior can be learned jointly with the rest of the model through back propagation. In addition to mitigating the forgetting problem, introducing such a memory module into our framework would facilitate interactive video segmentation. The object instances selected by the user can be stored in it and accessed during re-segmentation of the video, thus avoiding the need to finetune the network.

Publications

This thesis has led to the following publications.

International Conferences

- P. Tokmakov, K. Alahari, C. Schmid.
  Weakly-Supervised Semantic Segmentation using Motion Cues.
  Proceedings of the European Conference on Computer Vision (ECCV) 2016.

- P. Tokmakov, K. Alahari, C. Schmid.
  Learning Motion Patterns in Videos.
  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

- P. Tokmakov, K. Alahari, C. Schmid.
  Learning Video Object Segmentation with Visual Memory.
  Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2017.

International Journals - under review

- P. Tokmakov, C. Schmid, K. Alahari.
  Learning to Segment Moving Objects.
  International Journal on Computer Vision (IJCV) - submitted in December 2017.

Software

The following software packages produced in this thesis are available online.

- Source code and pretrained models for M-CNN: http://thoth.inrialpes.fr/research/weakseg/.
- Source code and pretrained models for MP-Net, as well as moving object labels for FT3D: http://thoth.inrialpes.fr/research/mpnet/.
- Source code and pretrained models for the video object segmentation with visual memory approach: http://thoth.inrialpes.fr/research/lvo/.

Bibliography

[1] http://vision.cs.unc.edu/ilsvrc2015/download-videos-3j16.php#vid. 32, 39, 105

[2] http://calvin.inf.ed.ac.uk/datasets/youtube-objects-dataset. 39

[3] http://thoth.inrialpes.fr/research/weakseg. 40

[4] Learning motion patterns in videos. http://thoth.inrialpes.fr/research/mpnet. 81

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 47

[6] E. H. Adelson. On seeing stuff: the perception of materials by humans and machines. In *HVEI*, 2001. 74

[7] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 116

[8] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 28

[9] N. Ballas, L. Yao, C. Pal, and A. Courville. Delving deeper into convolutional networks for learning video representations. *ICLR*, 2016. 67

[10] A. Bearman, O. Russakovsky, V. Ferrari, and F.-F. Li. What's the point: Semantic segmentation with point supervision. In *ECCV*, 2016. 27, 28, 29

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IIEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002. 81

[12] L. Bergen and F. Meyer. Motion segmentation and depth ordering based on morphological segmentation. *ECCV*, 1998. 57

[13] P. Bideau and E. G. Learned-Miller. It's moving! A probabilistic model for causal motion segmentation in moving camera videos. In *ECCV*, 2016. 55, 74, 116

[14] M. H. Bornstein, K. Ferdinandsen, and C. G. Gross. Perception of symmetry in infancy. *Developmental Psychology*, 17(1):82, 1981. 3

[15] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *ICCV*, 2001. 11, 36

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001. 37

[17] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *ICCV*, 2009. 61

[18] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 11, 16, 47, 62, 63, 118

[19] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011. 6, 7, 74, 76, 85, 86

[20] S. Caelles, K.-K. M. andJ. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video segmentation. In *CVPR*, 2017. 52, 60

[21] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012. 18

[22] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012. 18

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 18, 31, 34, 44, 45, 46, 69, 70, 84, 110

[24] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2, 69, 71

[25] M.-M. Cheng, N. J. Mitra, X. Huang, P. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 28

[26] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 65, 66

[27] R. G. Cinbis, J. Verbeek, and C. Schmid. Multi-fold MIL training for weakly supervised object localization. In *CVPR*, 2014. 25

[28] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, 2015. 23

[29] A. Dave, O. Russakovsky, and D. Ramanan. Predictive-corrective networks for action detection. In *CVPR*, 2017. 118

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. 20

[31] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 67

[32] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 69, 73

[33] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, 2002. 2, 8, 19, 25

[34] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 32, 38, 39, 70, 106

[35] A. Faktor and M. Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014. 6, 11, 52, 64, 94, 96

[36] R. L. Fantz, J. Fagan, and S. B. Miranda. Early visual selectivity. *Infant perception: From sensation to cognition*, 1:249–346, 1975. 3

[37] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. 18, 31

[38] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016. 67

[39] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 55

[40] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975. 65

[41] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video cnns through representation warping. In *ICCV*, 2017. 6

[42] F. Galasso, M. Keuper, T. Brox, and B. Schiele. Spectral graph reduction for efficient image and streaming video segmentation. In *CVPR*, 2014. 61

[43] F. Galasso, N. S. Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *ICCV*, 2013. 62

[44] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 83

[45] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 83, 106

[46] A. Graves, N. Jaitly, and A. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Workshop on Automatic Speech Recognition and Understanding*, 2013. 78

[47] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013. 67

[48] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. 66, 78

[49] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. In *CVPR*, 2010. 61

[50] U. Güçlü and M. A. van Gerven. Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27):10005–10014, 2015. 1

[51] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 39, 44

[52] G. Hartmann, M. Grundmann, J. Hoffman, D. Tsai, V. Kwatra, O. Madani, S. Vijayanarasimhan, I. Essa, J. Rehg, and R. Sukthankar. Weakly supervised learning of object segmentations from web-scale video. In *ECCV*, 2012. 16

[53] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 116, 118

[54] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1, 109

[55] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91, 1991. 65

[56] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 66

[57] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han. Weakly supervised semantic segmentation using web-crawled videos. In *CVPR*, 2017. 48, 102, 104, 105, 108, 109, 110, 111, 112, 115

[58] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. National Academy of Sciences*, 79(8):2554–2558, 1982. 65

[59] B. Horn. *Robot vision*. MIT press, 1986. 54

[60] J. F. Hughes and J. D. Foley. *Computer graphics: principles and practice*. Pearson Education, 2014. 57

[61] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 74, 86

[62] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 83

[63] S. D. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. In *CVPR*, 2017. 52, 58, 64, 82, 92, 93, 94, 95

[64] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014. 40

[65] B. Jin, M. V. Ortiz Segovia, and S. Susstrunk. Webly supervised semantic segmentation. In *CVPR*, 2017. 5

[66] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999. 2

[67] A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with Frank-Wolfe algorithm. In *ECCV*, 2014. 47, 48

[68] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, 2015. 11, 63, 82, 92, 93, 94, 118

[69] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. *CVPR*, 2017. 2, 5, 23, 24

[70] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017. 119

[71] A. Khoreva, F. Galasso, M. Hein, and B. Schiele. Classifier based graph construction for video segmentation. In *CVPR*, 2015. 61, 62

[72] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. In *CVPR*, 2017. 52, 60, 95

[73] K. Koffka. *Principles of Gestalt psychology*, volume 44. Hartcourt Brace Jovanovich, 1935. 62

[74] Y. J. Koh and C.-S. Kim. Primary object segmentation in videos based on region augmentation and reduction. In *CVPR*, 2017. 92, 93

[75] A. Kolesnikov and C. H. Lampert. Seed, expand and constraint: Three principles for weakly-supervised image segmentation. In *ECCV*, 2016. 3, 5, 8, 29, 30, 102, 104, 108, 109, 110

[76] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *NIPS*, 2011. 23, 84

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 18

[78] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, 2016. 119

[79] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised object discovery and tracking in video collections. In *ICCV*, 2015. 32, 47, 48

[80] T.-N. Le, K.-T. Nguyen, M.-H. Nguyen-Phan, T.-V. Ton, T.-A. N. (2), X.-S. Trinh, Q.-H. Dinh, V.-T. Nguyen, A.-D. Duong, A. Sugimoto, T. V. Nguyen, and M.-T. Tran. Instance re-identification flow for video object segmentation. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017. 119

[81] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995. 65

[82] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 18

[83] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011. 64

[84] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. 1, 18

[85] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011. 61

[86] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 70

[87] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. 28

[88] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, C. C. Loy, and X. Tang. Video object segmentation with re-identification. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017. 119

[89] X. Li, L. Zhao, L. Wei, M.-H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. In *CVPR*, 2016. 28

[90] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 116, 118

[91] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan. Towards computational baby learning: A weakly-supervised approach for object detection. In *ICCV*, 2015. 30

[92] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *CVPR*, 2017. 2

[93] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2, 71

[94] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016. 116

[95] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 18, 19, 31, 46, 69

[96] J. Mairal, F. Bach, J. Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014. 117

[97] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 53, 54, 69, 73, 81, 90

[98] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010. 67

[99] A. Monroy and B. Ommer. Beyond bounding-boxes: Learning object shape by model-driven grouping. In *ECCV*, 2012. 2, 16, 19

[100] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015. 46

[101] M. Narayana, A. R. Hanson, and E. G. Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. In *ICCV*, 2013. 6, 7, 10, 55, 74, 85, 116

[102] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 67

[103] P. Ochs and T. Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, 2011. 63

[104] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, 2012. 63

[105] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(6):1187–1200, 2014. 6, 70, 82, 118

[106] S. J. Oh, R. Benson, A. Khoreva, Z. Akata, M. Fritz, and B. Schiele. Exploiting saliency for object segmentation from image level labels. In *CVPR*, 2017. 3, 8, 30, 102, 104, 108

[107] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a DCNN for semantic image segmentation. In *ICCV*, 2015. 3, 5, 8, 9, 16, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34, 39, 40, 42, 43, 44, 45, 46, 48, 109, 110

[108] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 6, 10, 11, 32, 33, 36, 37, 38, 39, 40, 47, 48, 49, 52, 56, 58, 63, 64, 74, 93

[109] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, 2015. 3, 8, 9, 26, 27, 32, 39, 45, 46, 48

[110] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. In *ICLR*, 2015. 8, 9, 25, 27, 32, 45, 46

[111] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *ICLR Workshop track*, 2016. 67

[112] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 6, 70, 79, 80, 81, 82, 92

[113] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, 2015. 59, 105

[114] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *CVPR*, 2015. 5, 8, 16, 25, 32, 46

[115] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 52, 75

[116] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 30, 32, 38, 47, 105

[117] M. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *ICML*, 2008. 2

[118] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 52

[119] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *CVPR*, 2007. 59

[120] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 74, 86

[121] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*. Springer, 2016. 118

[122] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 18, 69

[123] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graphics*, 23(3):309–314, 2004. 36, 37

[124] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 82

[125] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *ECCV*, 2012. 25

[126] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015. 67

[127] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 1

[128] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 71, 84

[129] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 34

[130] E. S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990. 3, 116

[131] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 67

[132] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *NIPS*, 2015. 119

[133] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. In *ECCV*, 2010. 62, 74, 85, 86

[134] B. Taylor, V. Karasev, and S. Soatto. Causal video object segmentation from persistence of occlusions. In *CVPR*, 2015. 10, 11, 57, 58, 64, 80

[135] T. Tieleman and G. Hinton. RMSProp. COURSERA: Lecture 6.5 - Neural Networks for Machine Learning, 2012. 83

[136] P. Tokmakov, K. Alahari, and C. Schmid. Weakly-supervised semantic segmentation using motion cues. In *ECCV*, 2016. 9

[137] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *CVPR*, 2017. 12, 56, 75, 90, 92, 93, 94

[138] P. Tokmakov, K. Alahari, and C. Schmid. Learning video object segmentation with visual memory. In *ICCV*, 2017. 12

[139] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 356(1740):1321–1340, 1998. 6, 7, 10, 54, 55

[140] Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *CVPR*, 2016. 59, 60

[141] S. Valipour, M. Siam, M. Jagersand, and N. Ray. Recurrent fully convolutional networks for video segmentation. In *WACV*. IEEE, 2017. 6

[142] G. Van Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017. 117

[143] A. Vezhnevets, V. Ferrari, and J. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *CVPR*, 2012. 16

[144] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 60

[145] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *ACM Transactions on Graphics (ToG)*, volume 24, pages 585–594. ACM, 2005. 6

[146] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015. 63, 64

[147] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7032–7042, 2017. 117

[148] L. Wen, D. Du, Z. Lei, S. Z. Li, and M.-H. Yang. Jots: Joint online tracking and segmentation. In *CVPR*, 2015. 59

[149] P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990. 77, 79

[150] J. Wu, Y. Zhao, J. Zhu, S. Luo, and Z. Tu. MILCut: A sweeping line multiple instance learning paradigm for interactive image segmentation. In *CVPR*, 2014. 16

[151] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 1

[152] W. Xia, C. Domokos, J. Dong, L.-F. Cheong, and S. Yan. Semantic segmentation without annotating segments. In *ICCV*, 2013. 5, 23

[153] C. Xu and J. J. Corso. LIBSVX: A supervoxel library and benchmark for early video processing. *International Journal of Computer Vision*, 2016. 61

[154] J. Xu, A. G. Schwing, and R. Urtasun. Tell me what you see and I will show you where it is. In *CVPR*, 2014. 19, 25

[155] D. L. Yamins and J. J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016. 1

[156] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013. 63

[157] Y. Zhang, X. Chen, J. Li, C. Wang, and C. Xia. Semantic object segmentation via detection in weakly labeled video. In *CVPR*, 2015. 59

[158] H. Zhao, X. Puig, B. Zhou, S. Fidler, and A. Torralba. Open vocabulary scene parsing. In *ICCV*, 2017. 117

[159] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2

[160] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 31, 46

[161] B. Zhou, D. Bau, A. Oliva, and A. Torralba. Interpreting deep visual representations via network dissection. In *CVPR*, 2017. 117

[162] B. Zhou, A. Khosla, A. Lapedriza, O. Aude, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 8, 29, 48, 102, 115

[163] J. Zhu, J. Mao, and A. L. Yuille. Learning from weakly supervised data by the expectation loss svm (e-svm) algorithm. In *NIPS*, 2014. 5, 19, 23

[164] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014. 117