2019

Embodied Visual Perception Models For Human Behavior Understanding

Gediminas Bertasius
University of Pennsylvania, gedas90@gmail.com

Embodied Visual Perception Models For Human Behavior Understanding

Abstract
Many modern applications require extracting the core attributes of human behavior such as a person's attention, intent, or skill level from the visual data. There are two main challenges related to this problem. First, we need models that can represent visual data in terms of object-level cues. Second, we need models that can infer the core behavioral attributes from the visual data. We refer to these two challenges as ``learning to see'', and ``seeing to learn'' respectively. In this PhD thesis, we have made progress towards addressing both challenges.

We tackle the problem of ``learning to see'' by developing methods that extract object-level information directly from raw visual data. This includes, two top-down contour detectors, DeepEdge and HfL, which can be used to aid high-level vision tasks such as object detection. Furthermore, we also present two semantic object segmentation methods, Boundary Neural Fields (BNFs), and Convolutional Random Walk Networks (RWNs), which integrate low-level affinity cues into an object segmentation process. We then shift our focus to video-level understanding, and present a Spatiotemporal Sampling Network (STSN), which can be used for video object detection, and discriminative motion feature learning.

Afterwards, we transition into the second subproblem of ``seeing to learn'', for which we leverage first-person GoPro cameras that record what people see during a particular activity. We aim to infer the core behavior attributes such as a person's attention, intention, and his skill level from such first-person data. To do so, we first propose a concept of action-objects--the objects that capture person's conscious visual (watching a TV) or tactile (taking a cup) interactions. We then introduce two models, EgoNet and Visual-Spatial Network (VSN), which detect action-objects in supervised and unsupervised settings respectively. Afterwards, we focus on a behavior understanding task in a complex basketball activity. We present a method for evaluating players' skill level from their first-person basketball videos, and also a model that predicts a player's future motion trajectory from a single first-person image.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Jianbo Shi

Keywords
Computer Vision, Deep Learning, Edge Detection, Egocentric Vision, Machine Learning, Semantic Segmentation

EMBODIED VISUAL PERCEPTION MODELS FOR HUMAN BEHAVIOR UNDERSTANDING

Gediminas Bertasius

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2019

To my family

ACKNOWLEDGEMENT

For many students, PhD can be a frustrating and a lonely experience. Not only does it require extraordinary amounts of persistence, but also a substantial amount of luck. Over my five years, I observed that students who make it through their PhD typically have a very strong support system around them. I am no exception in this regard, and I attribute my overwhelmingly positive PhD experience to all the people who helped me reach this point.

First, I would like to thank my advisor Jianbo Shi for giving me full intellectual freedom to pursue any research interests that I was drawn to during the last five years. Jianbo's faith in me allowed me to work on many new and interesting problems, most of which later resulted in solid publications. Jianbo also taught me how to approach and formulate new research problems, which I think is one of the most invaluable skills that I learned throughout my PhD. Finally, I am thankful to Jianbo for always being supportive, especially after the negative paper reviews. His positivity during those times pushed me to work harder and eventually get those papers accepted to subsequent conferences. Overall, my PhD journey has been very fulfilling, and I am grateful to Jianbo for this experience.

I would also like to give a big thanks to Lorenzo Torresani, who is one of the main reasons why I pursued a PhD in the first place. Lorenzo sparked my interest in computer vision research during my undergraduate years at Dartmouth, and I've stayed in the field ever since. I was lucky to continue collaborating with Lorenzo during my PhD years, and I learned a great deal from Lorenzo about various research methodologies, experimental design, and technical writing. His guidance over all these years significantly contributed to my success, and I will always be grateful for his mentorship.

I am also grateful to Hyun Soo Park, and Stella Yu. I learned a lot from both of them. Hyun Soo taught me how to effectively sell my research ideas to other people both in writing and in presentation. I think these were truly invaluable lessons, that allowed me to publish as many papers as I did throughout my PhD. From Stella, I learned how to think about my research from a different perspective, and how to present it more effectively to broader audiences who have different backgrounds than my own.

I would also like to express my gratitude for everybody involved in the IGERT program. Being part of IGERT weekly meetings, allowed me to broaden my research horizons and think about my research in the context of other fields such as psychology, robotics, and human vision. During IGERT meetings, I had a great pleasure to observe Kostas Daniilidis', and David Brainard's discussions about research, and learn from both of them. Kostas and David consistently gave me useful feedback, and helped me to think critically about the broader impact of my work. A special thanks to Kostas for the basketball related discussions, which always brightened my mood.

I am also thankful to my other dissertation committee member CJ Taylor for his insightful comments and suggestions in regards to my research. I appreciated CJ's attention to detail, his thought provoking questions, and his kind personality. It was truly a pleasure having him on my dissertation and my WPE II committees.

Another big thanks goes to the Facebook's AI Applied Research team, who made my summer internship experience fruitful and enjoyable. In particular, I want to thank Lorenzo, Du Tran, and Christoph Feichtenhofer who were my mentors during the internship. Each of them has taught me something unique about video understanding research, and helped me to become a better researcher. I'd also like to give a special thanks to a fellow intern Antoine Miech who kept both of our spirits high throughout the internship.

Finally, I wanted to thank my parents Rima, and Gintaras, and my wife Alex for always supporting me. From the early age when my main priority was basketball, my parents always stressed the importance of education and encouraged me to excel in both of these areas, which led me to where I am now. Additionally, my wife Alex was the most ardent believer of my success, which inspired me to accomplish any goals that I set out for myself. I have been very lucky to share the highs and lows of my PhD with her.

ABSTRACT

EMBODIED VISUAL PERCEPTION MODELS FOR
HUMAN BEHAVIOR UNDERSTANDING

Gediminas Bertasius

Jianbo Shi

Many modern applications require extracting the core attributes of human behavior such as a person's attention, intent, or skill level from the visual data. There are two main challenges related to this problem. First, we need models that can represent visual data in terms of object-level cues. Second, we need models that can infer the core behavioral attributes from the visual data. We refer to these two challenges as "learning to see", and "seeing to learn" respectively. In this PhD thesis, we have made progress towards addressing both challenges.

We tackle the problem of "learning to see" by developing methods that extract object-level information directly from raw visual data. This includes, two top-down contour detectors, DeepEdge [1] and HfL [2], which can be used to aid high-level vision tasks such as object detection. Furthermore, we also present two semantic object segmentation methods, Boundary Neural Fields (BNFs) [3], and Convolutional Random Walk Networks (RWNs) [4], which integrate low-level affinity cues into an object segmentation process. We then shift our focus to video-level understanding, and present a Spatiotemporal Sampling Network (STSN), which can be used for video object detection [5], and discriminative motion feature learning [6].

Afterwards, we transition into the second subproblem of "seeing to learn", for which we leverage first-person GoPro cameras that record what people see during a particular activity. We aim to infer the core behavior attributes such as a person's attention, intention, and his skill level from such first-person data. To do so, we first propose a concept of action-objects–the objects that capture person's conscious visual (watching a TV) or tactile (taking a cup) interactions. We then introduce two models, EgoNet [7] and Visual-Spatial Network (VSN) [8], which detect action-objects in supervised and unsupervised settings respectively. Afterwards, we focus on a behavior understanding task in a complex basketball activity. We present a method for evaluating players' skill level from their first-person basketball videos [9], and also a model that predicts a player's future motion trajectory from a single first-person image [10].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Part I

Introduction

CHAPTER 1 : Problem Overview

Understanding the core attributes of human behavior such as attention, intent, or skill level is crucial for many applications. Nevertheless, directly measuring them is difficult because these attributes require understanding what that person is thinking in a particular situation.

Now, imagine a scenario where we could “tap” into a particular person’s visual system and see exactly what that person sees as he progresses through his day (see Figure 1). Could we then use such visual data to answer questions such as “what is that person paying attention to?”, “what is he doing now?”, “what will he do next?”, “how good is he at a particular activity?”.

For instance, take a look at the first image in row one of Figure 1. This image illustrates almost exactly what a person sees during a particular activity. Just by looking at this image, we can immediately tell a few things about the person behind the camera. First, we know that he is in the kitchen, and thus, he is probably cooking. Furthermore, it also seems that his attention is currently focused on the kitchen stove, and thus we could predict that he might approach the kitchen stove next. Finally, from some of the objects in the scene (i.e. a steak on the cutting board), we can even guess what kind of dish he is preparing.

Similarly consider the sequence of images in row two of Figure 1. Just by looking at these images, we can easily tell that the person is currently playing basketball. However, there are also many other details in these images that reveal more subtle cues about this person. For instance, from his current position on the court, we can infer that he is the point guard in his team. Furthermore, throughout the sequence, we observe that by calling a certain play he outmaneuvers both of the defenders, which allows him to create an open three point shot for himself. Not only does such an action signifies that this person has a good decision making ability, but it also reveals that he is a skilled shooter as we observe his shot mechanics from up close.

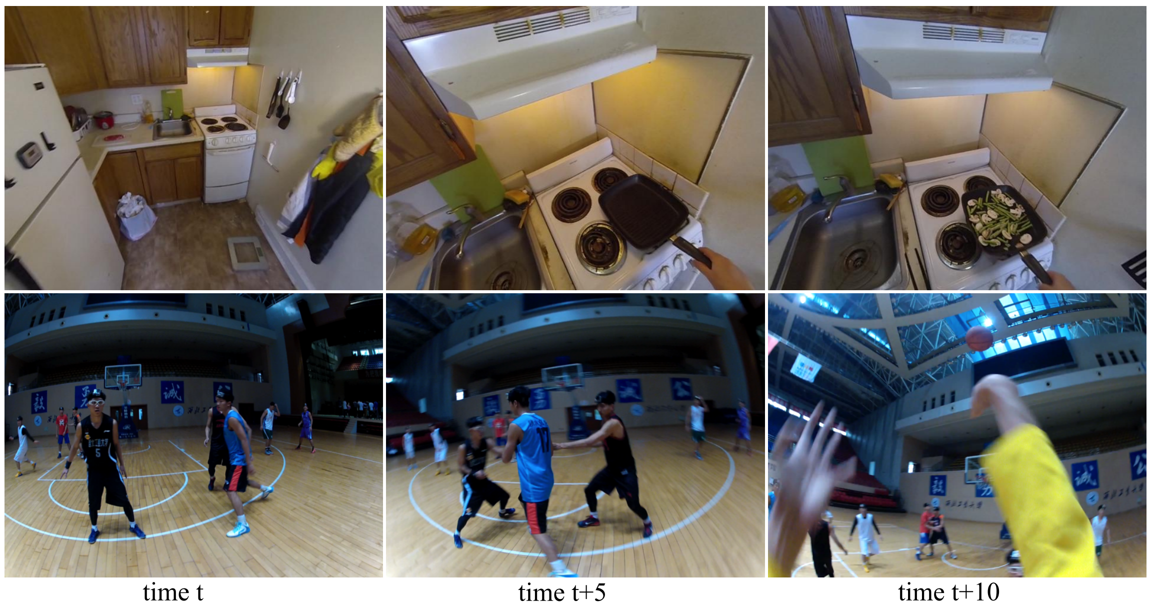


Figure 1: Egocentric images recorded using a GoPro camera attached to a person's head. The recorded first-person images capture almost exactly what that person sees as he performs activities such as cooking, or playing basketball. Such visual data allows us to infer a person's attention, his intentions, his skill level at a particular activity, and even his decision making process in different situations.

Why would knowing any of these things be useful? Consider an application of building a personal AI assistant inside one's home. Suppose such a system had access to the visual data of a person cooking a meal (row one of Figure 1). With the ability to infer that person's attention and his intentions, such a system could automatically control gadgets such as the kitchen stove, order missing materials when needed, and assist this person in a number of different ways. Similarly, having access to a particular person's visual feed during activities such as basketball (row two of Figure 1), would allow us to build novel systems for player's decision making analysis and their skill assessment. This would be highly beneficial as nowadays these tasks are still performed by humans, making it a costly and a time consuming process.

Inspired by these ideas, we aim to build visual perception models for inferring the core attributes of human behavior such as attention, intention, and skill. We identify two key challenges related to this problem. First, we need models that can represent visual data in terms of object-level cues– a representation that would allow us to understand the observed environment better. Second, we need models that can infer the core behavioral attributes from the visual data. We refer to these two challenges as “learning to see”, and “seeing to learn” respectively.

We tackle the problem of “learning to see” by developing techniques that extract object-level information directly from raw visual data. To do so, we first propose two top-down edge detectors: DeepEdge [1] and HFL [2], which utilize high-level object features for a low-level boundary detection process. We show that due to the semantic nature of our boundaries we can also use these boundaries to aid a number of high-level vision tasks such as semantic object segmentation or object detection. In addition, we also present two effective object segmentation methods, Boundary Neural Fields (BNF) [3] and Convolutional Random Walk Networks (RWN) [4], which successfully integrate low-level affinity cues for improved semantic object segmentation performance. Furthermore, we introduce a novel Spatiotemporal Sampling Network (STSN) architecture, which we use for video object detection [5], and discriminative motion feature learning [6].

Afterwards, we transition to the problem of “seeing to learn”. To tackle this problem, we leverage first-person GoPro cameras, which allow us to study the interplay between a person’s visual attention and his actions. First, we propose a concept of action-objects– the objects that capture person’s conscious visual (watching a TV) or tactile (taking a cup) interactions. We then present EgoNet [7], a joint two-stream network that holistically integrates visual appearance (RGB) and 3D spatial layout (depth and height) cues to predict per-pixel likelihood of action-objects using a supervised learning objective. Next, to enable unsupervised learning of action-objects, we introduce a Visual-Spatial Network (VSN) [8] consisting of spatial (“where”) and visual (“what”) pathways, which are optimized using our proposed alternating cross-pathway supervision scheme.

Finally, we describe several approaches for behavior understanding in a complex basketball activity. First, we present a method that assesses basketball players’ performance from first-person videos in an unscripted five-on-five basketball game [9]. Next, we propose a model that takes as input a single first-person image capturing exactly what a player sees, and then predicts his future motion trajectory [10].

In the subsequent chapters, we will discuss each of these systems in more detail. Furthermore, we will present experiments validating the effectiveness of each method at its respective task.

Part II

Learning to See

CHAPTER 1 : DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection

1.1. Introduction

Contour detection is typically considered a low-level problem, and used to aid higher-level tasks such as object detection [30, 31, 32, 33]. However, it can be argued that the tasks of detecting objects and predicting contours are closely related. For instance, given the contours we can easily infer which objects are present in the image. Conversely, if we are given exact locations of the objects we could predict contours just as easily. A commonly established pipeline in computer vision starts with with low-level contour prediction and then moves up to higher-level object detection. However, since we claim that these two tasks are mutually related, we propose to invert this process. Instead of using contours as low-level cues for object detection, we want to use object-specific information as high-level cues for contour detection. Thus, in a sense our scheme can be viewed as a top-down approach where object-level cues inform the low-level contour detection process.

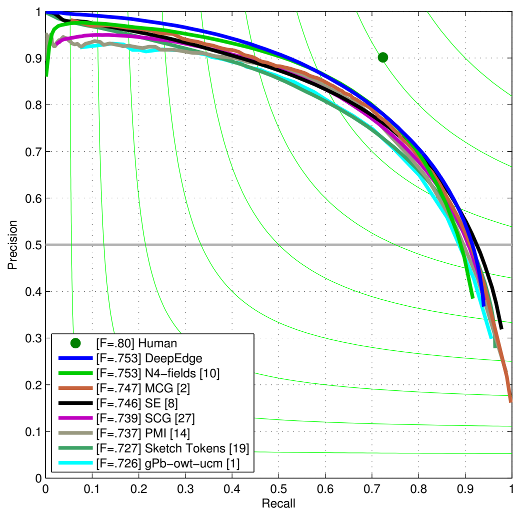


Figure 2: Contour detection accuracy on the BSDS500 dataset. Our method attains higher average precision compared to prior methods and state-of-the-art F-score. At low recall, DeepEdge achieves nearly 100% precision.

In this work, we present a unified multi-scale deep learning approach that uses higher-level object information to predict contours. Specifically, we present a front-to-end convolutional architecture where contours are learned directly from raw pixels. Our proposed deep learning architecture reuses features computed by the first five convolutional layers of the

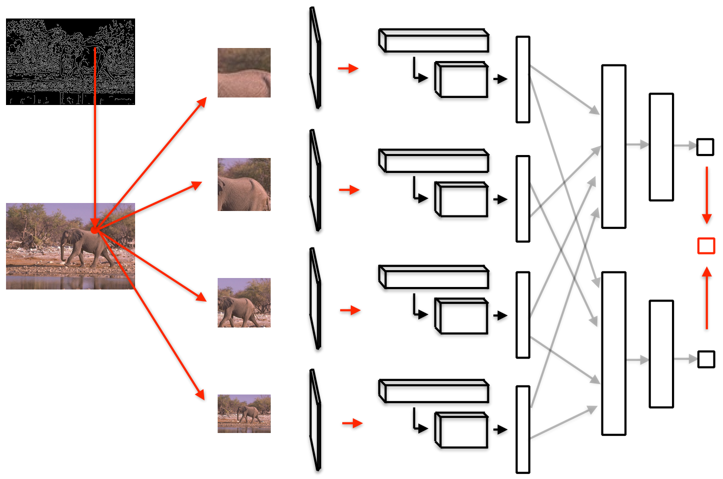


Figure 3: Visualization of multi-scale DeepEdge network architecture. To extract candidate contour points, we run the Canny edge detector. Then, around each candidate point, we extract patches at four different scales and simultaneously run them through the five convolutional layers of the *KNet* [26]. We connect these convolutional layers to two separately-trained network branches. The first branch is trained for classification, while the second branch is trained as a regressor. At testing time, the scalar outputs from these two sub-networks are averaged to produce the final score.

network of Krizhevsky et al. [26]. We refer to this network as the *KNet*. Because the KNet has been trained for object classification, reusing its features enable our method to incorporate object-level information for contour prediction. In the experimental section, we will show that this high-level object information greatly improves contour detection results.

Furthermore, our defined architecture operates on multiple scales simultaneously and combines local and global information from the image, which leads to significantly improved contour detection accuracy rates.

We connect the features computed by the convolutional layers of the KNet at four different scales of the input with a *learned* subnetwork that bifurcates into two branches (the architecture of our model is illustrated in Fig. 3).

What should the learning objective be? When a human observer decides if a pixel is a boundary edge, a number of supporting evidence is used with object level reasoning.

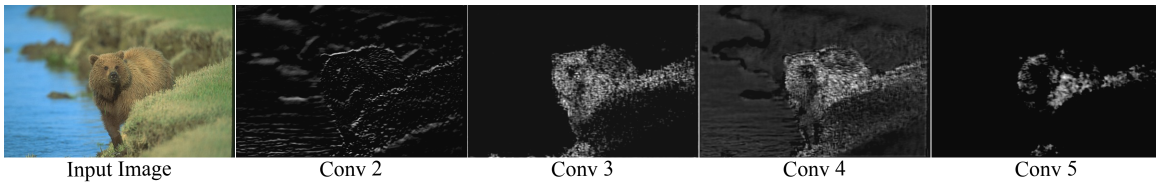


Figure 4: Visualization of the activation values at the selected convolutional filters of the *KNet* (filters are resized to the original image dimensions). The filters in the second layer fire on oriented edges inside the image. The third and fourth convolutional layers produce an outline of the shape of the object. The fifth layer fires on the specific parts of the object.

While it is impossible to record such information, we do have the fraction of observers in agreement for each pixel. We argue that a learning objective that predicts the fraction of human labelers in agreement can mimic human reasoning better.

Thus, in the bifurcated sub-network we optimize the two branches with different learning objectives. The weights in one branch are optimized with an edge classification objective, while the other branch is trained to predict the fraction of human labelers in agreement, i.e., using a regression criterion. We show that predictions from the classification branch yield high edge recall, while the outputs of the regression branch have high precision. Thus, fusing these two outputs allows us to obtain excellent results with respect to both metrics and produce state-of-the-art F-score and average precision.

In summary, the use of higher-level object features, independent optimization of edge classification and regression objectives, as well as a unified multi-scale architecture are the key characteristics that allow our method to achieve the state-of-the-art in contour detection (see Fig. 2).

1.2. Related Work

Deep Learning. In the recent years, deep convolutional networks have achieved remarkable results in a wide array of computer vision tasks [34, 35, 36, 26]. However, thus far, applications of convolutional networks focused on high-level vision tasks such as face recognition, image classification, pose estimation or scene labeling [34, 35, 36, 26]. Excellent results in these tasks beg the question whether convolutional networks could perform equally well in lower-level vision tasks such as contour detection. In this paper, we present a convolutional architecture that achieves state-of-the-art results in a contour detection task, thus demonstrating that convolutional networks can be applied successfully for lower-level vision tasks as well.

Edge Detection. Most of the contour detection methods can be divided into two branches: local and global methods. Local methods perform contour detection by reasoning about small patches inside the image. Some recent local methods include sketch tokens [37] and structured edges [15], Both of these methods are trained in a supervised fashion using a random forest classifier. Sketch tokens [37] pose contour detection as a multi-class classification task and predicts a label for each of the pixels individually. Structured edges [15], on the other hand, attempt to predict the labels of multiple pixels simultaneously.

Global methods predict contours based on the information from the full image. Some of the most successful approaches in this genre are the MCG detector [38], gPb detector [30] and sparse code gradients [39]. While sparse code gradients use supervised SVM learning [40], both gPb and MCG rely on some form of spectral methods. Other spectral-based methods include Normalized Cuts [41] and PMI [42].

Recently, there have also been attempts to apply deep learning methods to the task of contour detection. While SCT [43] is a sparse coding approach, both N^4 fields [12] and DeepNet [13] use Convolutional Neural Networks (CNNs) to predict contours. N^4 fields rely on dictionary learning and the use of the Nearest Neighbor algorithm within a CNN framework while DeepNet uses a traditional CNN architecture to predict contours.

In comparison to these prior approaches, our work offers several contributions. First, we define a novel multi-scale bifurcated CNN architecture that enables our network to achieve state-of-the-art contour detection results. Second, we avoid manual feature engineering by learning contours directly from raw data. Finally, we believe that we are the first to propose

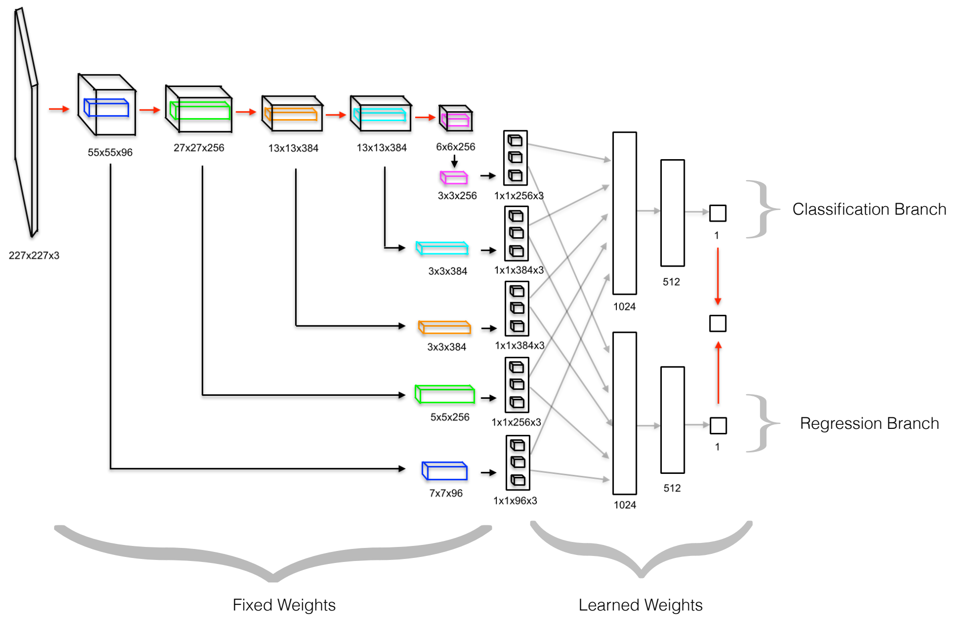


Figure 5: Detailed illustration of our proposed architecture in a single-scale setting. First, an input patch, centered around the candidate point, goes through five convolutional layers of the *KNet*. To extract high-level features, at each convolutional layer we extract a small sub-volume of the feature map around the center point, and perform *max*, *average*, and *center* pooling on this sub-volume. The pooled values feed a bifurcated sub-network. At testing time, the scalar outputs computed from the branches of a bifurcated sub-networks are averaged to produce a final contour prediction.

the use of high-level object features for contour detection, thus inverting the traditional pipeline relying on low-level cues for mid-level feature extraction. Our experiments show that this top-down approach for contour detection yields state-of-the-art results.

1.3. The DeepEdge Network

In this section, we describe our proposed deep learning approach for contour detection. For simplicity, we first present our architecture in the single-scale scenario (subsection 1.3.1) and then discuss how to take advantage of multiple scales (subsection 1.3.2).

1.3.1. Single-Scale Architecture

Selection of Candidate Edge Points. To extract a set of candidate contours with high recall we apply the Canny edge detector [44] to the input image. For each of these points we then extract a patch of fixed size such that our candidate point is the center of the patch. Patches that do not fit into the image boundaries are padded with the mirror reflections of itself.

Extraction of High-Level Features. We then resize the patch of fixed size to match the input dimensions of the KNet [26] and use this network to extract object-level features. The KNet is an appropriate model for our setting as it has been trained over a large number of object classes (the 1000 categories of the ImageNet dataset [45]) and thus captures features that are generic and useful for many object categories. While such features have been optimized for the task of object class recognition, they have been shown to be highly effective for other image-analysis tasks, including object detection [46], attribute prediction [47], and image style recognition [48]. The network was trained on 1.2 million images and it includes more than 60 million parameters. Its architecture consists of 5 convolutional layers and 3 fully connected layers. As we intend to use the KNet as a feature extractor for boundary detection, we utilize only the first 5 convolutional layers, which preserve explicit location information before the "spatial scrambling" of the fully connection layers (note that a spatially variant representation is crucially necessary to predict the presence of contours at individual pixels). The first two KNet convolutional layers learn low-level information. As we move into the deeper layers, however, we observe that the network learns higher-level object information. The second convolutional layer seems to encode coherent edge structures. The third convolutional layer fires at locations corresponding to prototypical object shapes. The fourth layer appears to generate high responses for full shapes of the object, whereas the fifth layer fires on the specific object parts (See Fig. 4).

In order to obtain a representation that captures this hierarchical information, we perform feature extraction at each of the five convolutional layers, as shown in Fig. 5. Specifically, we consider a small sub-volume of the feature map stack produced at each layer. The sub-volume is centered at the center of the patch in order to assess the presence of a contour in a small area around the candidate point. We then perform *max*, *average*, and *center* pooling

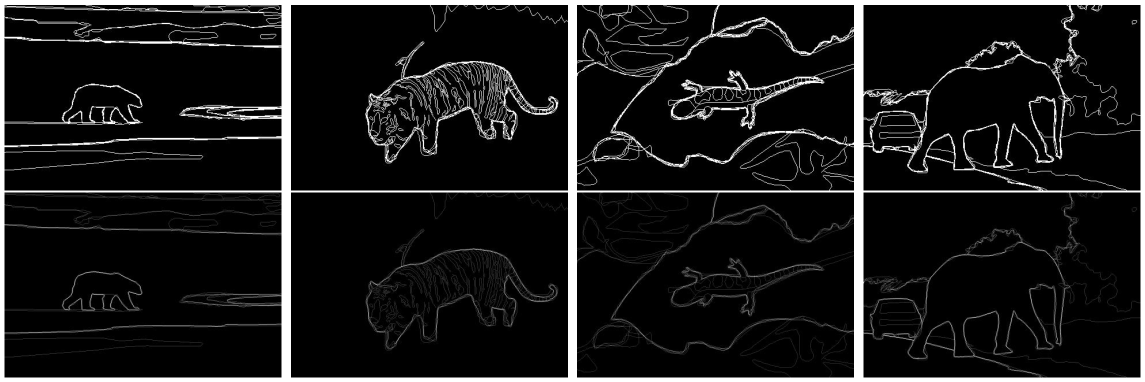


Figure 6: A few samples of ground truth data illustrating the difference between the classification (first row) and the regression (second row) objectives. The classification branch is trained to detect contours that are marked by at least one of the human annotators. Conversely, the regression branch is optimized to the contour values that depict the fraction of human annotators agreeing on the contour.

on this sub-volume. This yields a feature descriptor of size $3 \times F$ where F is the number of feature maps computed by the convolutional layer. While max and average pooling are well established operations in deep learning, we define center pooling as selecting the center-value from each of the feature maps. The motivation for center pooling is that for each candidate point we want to predict the contour presence at that particular point. Because the candidate point is located at the center of the input patch, center pooling extracts the activation value from the location that corresponds to our candidate point location.

A Bifurcated Sub-Network. We connect the feature maps computed via pooling from the five convolutional layers to two separately-trained network branches. Each branch consists of two fully-connected layers. The first branch is trained using binary labels, i.e., to perform contour classification. This branch is making less selective predictions by classifying whether a given point is a contour or not. In a sense, this classification branch abstracts details related to the edge structure (orientation, strength, etc) and simply tries to predict the presence/absence of an edge at a given point. Due to such abstractions, the classification branch produces contour predictions with high recall.

The second branch is optimized as a regressor to predict the fraction of human labelers agreeing about the contour presence at a particular point. Due to a regression objective, this branch is much more selective than the first branch. Intuitively, the second branch is trained to learn the structural differences between the contours that are marked by a different fraction of human labelers. For instance, the area that was labeled as a contour by 80% of human labelers must be significantly different than the area that was labeled as a contour by 20% human labelers. The regression branch is trying to learn such differences by predicting the fraction of human labelers who would mark a particular point as a contour. Thus, in a sense, we are training the regression branch to implicitly mimic how human labelers reason about the contour presence at a given point. In the experimental section, we demonstrate that due to its selectivity, the regression branch produces contour predictions with very high precision. In Fig. 6, we present several samples of ground truth data that illustrate the different properties of our two end-objectives.

The number of hidden layers in the first and second fully connected layers of both branches are 1024 and 512, respectively. Both branches optimize the sum of squared difference loss over the (binary or continuous) labels. At testing time, the scalar outputs computed from these two sub-networks are averaged to produce a final score indicative of the probability that the candidate point is a contour. Visualization of this architecture is presented in Fig. 5.

In order to train our sub-network, we generate patch examples and labels using training images with ground truth annotations from multiple human labelers. To generate the binary labels, we first sample $40,000$ positive examples that were marked as contours by at least one of the labelers. To generate negative examples we consider the points that were selected as candidate contour points by the Canny edge detector but that have not been marked as contours by any of the human labelers. These are essentially false positives. For training, we use a random subset of $40,000$ of such points in order to have equal proportion of negative and positive examples. These $80,000$ examples are then used to train our classification sub-network.

In addition to the binary labels, we also generate continuous labels that are used to train the regression network. For this purpose, we define the regression label of a point to be the fraction of human labelers that marked the point as a contour. These $80,000$ examples with continuous labels are then also used to train our regression sub-network.

1.3.2. Multi-Scale Architecture

In the previous section we presented a convolutional architecture for contour prediction utilizing a single scale. However, in practice, we found that a multi-scale approach works much better. In this section, we show how to modify the single-scale architecture so that it can exploit multiple scales simultaneously.

Rather than extracting a patch at a single scale as we did in the previous section, in a multi-scale setting we extract patches around the candidate point for different patch sizes so that they cover different spatial extents of the image. We then resize the patches to fit the KNet input and pump them in parallel through the five convolutional layers. Our high-level features are then built by performing max, average and center pooling in a small sub-volume of the feature map at each convolutional layer and at each scale. This effectively increases the dimensionality of the feature vector by a factor equal to the number of scales compared to the single-scale setting. These pooled features are then connected as before to the two separately-trained network branches. A visualization of our multi-scale architecture is shown in Fig. 3.

1.3.3. Implementation Details

In this section, we describe additional implementation details of our model. Our deep network is implemented using the software library Caffe [16].

We use four different scales for our patches. The sizes of these patches are $64 \times 64, 128 \times 128, 196 \times 196$ and a full-sized image. All of the patches are then resized to the *KNet* input dimensions of 227×227.

When extracting high-level features from the convolutional layers of *KNet*, we use subvolumes of convolutional feature maps having spatial sizes $7 \times 7, 5 \times 5, 3 \times 3, 3 \times 3$, and 3×3 for the convolutional layers $1, 2, 3, 4, 5$, respectively. Note that we shrink the size of the subvolume as we go deeper in the network since the feature maps get smaller due to pooling. Our choice of subvolume sizes is made to ensure we are roughly considering the same spatial extent of the original image at each layer.

As illustrated in Fig. 5, during the training the weights in the convolutional layers are fixed and only the weights in the fully connected layers of the two branches are learned.

To train our model we use the learning rate of 0.1, the dropout fraction of 0.5, 50 number of epochs, and the size of the batch equal to 100.

As described earlier, to train classification and regression branches we sample $80,000$ examples with binary labels. We also generate continuous labels for these $80,000$ examples. In addition, we sample a hold-out dataset of size $40,000$. This hold-out dataset is used for the hard-positive mining step [49].

For the first 25 epochs we train classification and regression branches independently on the original $80,000$ sample training dataset. After the first 25 epochs, we test both branches on the hold-out dataset and detect false negative predictions made by each branch. We then use these false negative examples along with the same number of randomly selected true negative examples to augment our original $80,000$ training dataset. For the remaining 25 epochs, we train both branches on this augmented dataset.

The motivation for the hard-positive mining step is to reduce the number of false negative predictions produced by both branches. By augmenting the original $80,000$ sized training data with false negative examples, we are forcing both branches to focus on *hard positive* examples, and thus, effectively reducing the number of false negative predictions.

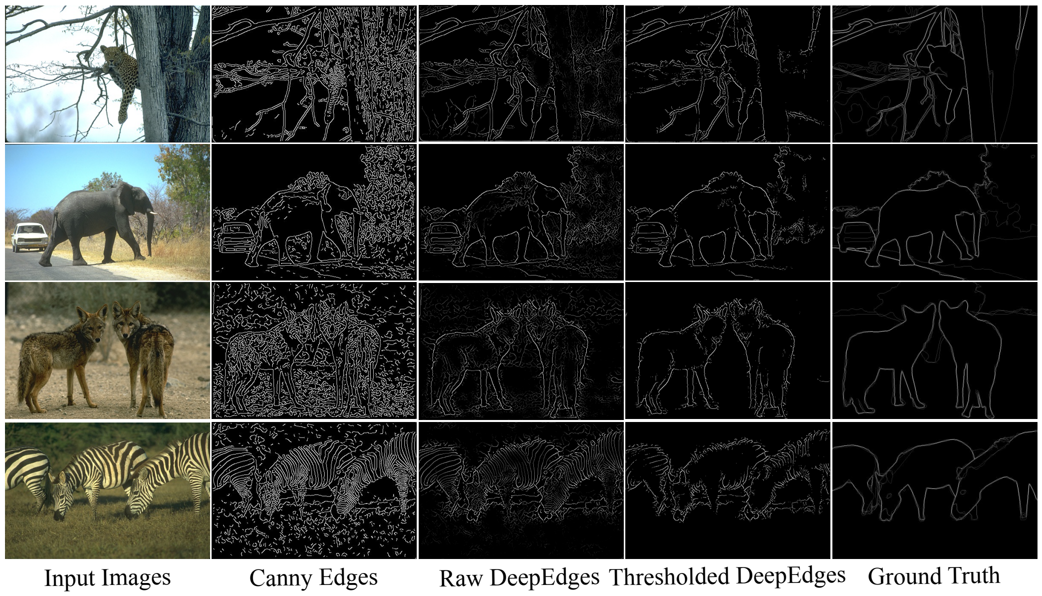


Figure 7: Qualitative results produced by our method. Notice how our method learns to distinguish between strong and weak contours. For instance, in the last row of predictions, contours corresponding to zebra stripes are assigned much lower probabilities than contours that correspond to the actual object boundaries separating the zebras from the background.

1.4. Experiments

In this section, we present our results on the BSDS500 dataset [50], which is arguably the most established benchmark for contour detection. This dataset contains 200 training images, 100 validation images, and 200 test images. Contour detection accuracy is evaluated using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP).

In section 1.4.1 we quantitatively compare our approach to the state-of-the-art. In section 1.4.2 we study how the performance of our system changes as we modify some of the architecture choices (number of scales, feature maps, pooling scheme, training objective). This will cast additional insight into the factors that critically contribute to the high accuracy of our system.

Method	ODS	OIS	AP
gPb-owt-ucm [30]	0.726	0.757	0.696
Sketch Tokens [37]	0.727	0.746	0.780
PMI [42]	0.737	0.771	0.783
DeepNet [13]	0.738	0.759	0.758
SCG [39]	0.739	0.758	0.773
SE [15]	0.746	0.767	0.803
MCG [38]	0.747	**0.779**	0.759
N^4-fields [12]	**0.753**	0.769	0.784
DeepEdge	**0.753**	0.772	**0.807**

Table 1: Edge detection results on the BSDS500 benchmark. Our DeepEdge method achieves state-of-the-art contour detections results according to both F-score and AP metrics.

1.4.1. Comparison with Other Methods

We compare the results produced by our approach and previously proposed contour detection methods. Table 1 summarizes the results. We note that our algorithm achieves contour detection accuracy that is higher or equal to state-of-the-art results according to two of the three metrics.

Fig. 2 shows the precision and recall curve for the methods considered in our comparison. It also lists the F-score for each method (in the legend). We observe that there is the accuracy margin separating our approach from prior techniques. In particular, for low-recall our method achieves almost perfect precision rate. It also produces state-of-the-art F-score.

1.4.2. Ablation Studies

Single Scale versus Multiple Scales. In this section we study the benefits of a multi-scale architecture. Results in Table 2 report accuracy for different numbers and choices of scales. The first four rows in the table illustrate the results achieved using a single-scale approach. Specifically, these four cases show performance obtained when training and testing our system with an input patch of size $64 \times 64, 128 \times 128, 196 \times 196$ or a full-sized image, respectively. Note that adding information from multiple scales leads to significantly

Scale	ODS	OIS	AP
64	0.71	0.73	0.76
128	0.72	0.74	0.78
196	0.71	0.73	0.76
Full Image	0.67	0.69	0.57
64, 128	0.72	0.75	0.78
64, 128,196	0.72	0.75	0.78
64,128,196,Full Image	**0.75**	**0.77**	**0.81**

Table 2: Results illustrating the effect of using a multi-scale architecture. Considering multiple scales for contour detection yields significantly higher accuracy relative to a single scale approach.

higher F-scores and higher average precisions. Thus, these results suggest that a multi-scale approach is highly advantageous in comparison to a single scale setting.

Advantages of Higher-Level Features. In this section, we examine the validity of our earlier claim that higher-level object-features enhance contour detection accuracy. In Table 3, we present individual contour detection results using features from the different convolutional layers of *KNet*. Note that the 4^{th} convolutional layer produces the most effective features when considering one layer at a time. From our earlier discussion we know that the 4^{th} convolutional layer encodes higher-level object information related to shape and specific object parts. This indicates that object specific cues are particularly beneficial for contour detection accuracy.

We also observe that by incorporating features from all the convolutional layers, our method achieves state-of-the-art contour detection results. This suggests that the features computed by different layers are complementary and that considering information from the entire hierarchy is advantageous.

Pooling Scheme. When presenting the architecture of our model, we discussed three different types of pooling: *max*, *average*, and *center* pooling. These three techniques were used to pool the values from the sub-volumes extracted around the center point in each convolutional filter as illustrated in Fig. 5.

Conv. Layers	ODS	OIS	AP
1^{st}	0.66	0.68	0.69
2^{nd}	0.71	0.74	0.76
3^{rd}	0.74	0.75	0.79
4^{th}	0.74	0.76	0.79
5^{th}	0.73	0.74	0.77
All	**0.75**	**0.77**	**0.81**

Table 3: This table shows the advantage of using higher-level features from the *KNet* convolutional layers. Individually, the 4^{th} convolutional layer produces the best contour prediction results, which implies that higher-level object information is indeed beneficial for contour detection. Combining the features from all convolutional layers leads to state-of-the-art results.

Pooling Type	ODS	OIS	AP
Average	0.73	0.75	0.78
Max	0.69	0.72	0.73
Center	0.74	0.76	0.8
Avg+Max+Cen	**0.75**	**0.77**	**0.81**

Table 4: Effect of different pooling schemes on contour detection results. Center pooling produces better results than max or average pooling. Combining all three types of pooling further improves the results.

We now show how each type of pooling affects contour detection results. Table 4 illustrates that, individually, center pooling yields the best contour detection results. This is expected because the candidate point for which we are trying to predict a contour probability is located at the center of the input patch.

However, we note that combining all three types of pooling, achieves better contour detection results than any single pooling technique alone.

Bifurcation and Training Objective. Next, we want to show that that the two independently-trained classification and regression branches in the bifurcated sub-network provide complementary information that yields improved contour detection accuracy. In Table 5, we present contour detection results achieved by using predictions from the individual branches of the bifurcated sub-network.

From these results, we observe that using predictions just from the classification branch

Branch	ODS	OIS	AP
Classification	**0.75**	0.76	0.78
Regression	0.74	0.76	0.80
Classification+Regression	**0.75**	**0.77**	**0.81**

Table 5: Contour detection accuracy of the two branches in our bifurcated sub-network. The classification branch yields solid F-score results whereas the regression branch achieves high average precision. Averaging the outputs from these two branches further improve the results.

produces high F-score whereas using predictions only from the regression branch yields high average precision. Combining the predictions from both branches improves the results according to both metrics thus, supporting our claim that separately optimizing edge classification and regression objectives is beneficial to contour detection.

1.4.3. Qualitative Results

Finally, we present qualitative results produced by our method. In Figure 7 we show for each input image example, the set of candidate points produced by the Canny edge detector, the un-thresholded predictions of our method, the thresholded predictions, and the ground truth contour map computed as an average of the multiple manual annotations. To generate the thresholded predictions, we use a probability threshold of 0.5.

Note that our method successfully distinguishes between strong and weak contours. Specifically, observe that in the last row of Figure 7, our method assigns lower probability to contours corresponding to zebra stripes compared to the contours of the actual object boundary separating the zebras from the background. Thus, in the thresholded version of the prediction, the weak contours inside the zebra bodies are removed and we obtain contour predictions that look very similar to the ground truth.

Due to locality of our method, it may be beneficial to apply spectral methods [51, 41] or conditional random fields [52] on top of our method to further improve its performance.

1.4.4. Computational Cost

In its current form, our method requires about $60K$ KNet evaluations ($15K$ per scale) to extract the features. Based on the runtimes reported in [16], if executed on a GPU our method would take about 5 minutes and could be made faster using the approach described in [53].

An alternative way to dramatically reduce the runtime of DeepEdge is to interpolate the entries of the feature maps produced by applying the KNet to the full image rather than to individual patches. Such an approach would reduce the number of CNN evaluations needed from $60K$ to 4 (one for each scale), which would allow our method to run in real time even on CPUs. We note that interpolation of features in deep layers has been used successfully in several recent vision papers [54, 55, 56]. Thus, we believe that such an approach could yield nearly equivalent contour detection accuracy, up to a small possible degradation caused by interpolation.

Since in this work we were primarily interested in studying the effective advantage enabled by object-level features in contour detection, we have not invested any effort in optimizing the implementation of our method. This will be one of our immediate goals in the future.

1.5. Conclusions

In this work, we presented a multi-scale bifurcated deep network for top-down contour detection. In the past, contour detection has been approached as a bottom-up task where low-level features are engineered first, then contour detection is performed, and finally contours may be used as cues for object detection. However, due to a close relationship between object and contour detection tasks, we proposed to invert this pipeline and perform contour detection in a top-down fashion. We demonstrated how to use higher-level object cues to predict contours and showed that considering higher-level object-features leads to a substantial gain in contour detection accuracy.

Additionally, we demonstrated that our multi-scale architecture is beneficial to contour prediction as well. By considering multiple scales, our method incorporates local and global information around the candidate contour points, which leads to significantly better contour detection results. Furthermore, we showed that independent optimization of contour classification and regression objectives improves contour prediction accuracy as well. As our experiments indicate, DeepEdge achieves higher average precision results compared to any prior or concurrent work.

In conclusion, our results suggest that pure CNN systems can be applied successfully to contour detection and possibly to many other low-level vision tasks.

CHAPTER 2 : High-for-Low and Low-for-High: Efficient Boundary Detection from Deep Object Features and its Applications to High-Level Vision

2.1. Introduction

In the vision community, boundary detection has always been considered a low-level problem. However, psychological studies suggest that when a human observer perceives boundaries, object level reasoning is used [57, 58, 59]. Despite these findings, most of the boundary detection methods rely exclusively on low-level color and gradient features. In this work, we present a method that uses object-level features to detect boundaries. We argue that using object-level information to predict boundaries is more similar to how humans reason. Our boundary detection scheme can be viewed as a *High-for-Low* approach where we use high-level object features as cues for a low-level boundary detection process. Throughout this paper, we refer to our proposed boundaries as *High-for-Low* boundaries (HfL).

We present an efficient deep network that uses object-level information to predict the boundaries. Our proposed architecture reuses features from the sixteen convolutional layers of the network of Simonyan et al. [60], which we refer to as VGG net. The VGG net has been trained for object classification, and therefore, reusing its features allows our method to utilize high-level object information to predict HfL boundaries. In the experimental section, we demonstrate that using object-level features produces semantically meaningful boundaries and also achieves above state-of-the-art boundary detection accuracy.

Additionally, we demonstrate that we can successfully apply our HfL boundaries to a number of high-level vision tasks. We show that by using HfL boundaries we improve the results of three existing state-of-the-art methods on the tasks of semantic boundary labeling, semantic segmentation and object proposal generation. Therefore, using HfL boundaries to boost the results in high level vision tasks can be viewed as a *Low-for-High* scheme, where boundaries serve as low-level cues to aid high-level vision tasks.

	Low-Level Task	High-Level Tasks			
	BD	SBL		SS	OP
	ODS	MF	AP	PI-IOU	MR
SotA	0.76 [11]	28.0 [14]	19.9 [14]	45.8 [29]	0.88 [61]
HfL	**0.77**	**62.5**	**54.6**	**48.8**	**0.90**

Table 6: Summary of results achieved by our proposed method (HfL) and state-of-the-art methods (SotA). We provide results on four vision tasks: Boundary Detection (BD), Semantic Boundary Labeling (SBL), Semantic Segmentation (SS), and Object Proposal (OP). The evaluation metrics include ODS F-score for BD task, max F-score (MF) and average precision (AP) for SBL task, per image intersection over union (PI-IOU) for SS task, and max recall (MR) for OP task. Our method produces better results in each of these tasks according to these metrics.

We present the summarized results for the boundary detection and the three mentioned high-level vision tasks in Table 6. Specifically, we compare our proposed method and an appropriate state-of-the-art method for that task. As the results indicate, we achieve better results in each of the tasks for each presented evaluation metric. We present more detailed results for each of these tasks in the later sections.

In summary, our contributions are as follows. First, we show that using object-level features for boundary detection produces perceptually informative boundaries that outperform prior state-of-the-art boundary detection methods. Second, we demonstrate that we can use HfL boundaries to enhance the performance on the high-level vision tasks of semantic boundary labeling, semantic segmentation and object proposal. Finally, our method can detect boundaries in near-real time. Thus, we present a boundary detection system that is accurate, efficient, and is also applicable to high level vision tasks.

2.2. Related Work

Most of the contour detection methods predict boundaries based purely on color, text, or other low-level features. We can divide these methods into three broad categories: spectral methods, supervised discriminative methods and deep learning based methods.

Spectral methods formulate contour detection problem as an eigenvalue problem. The solution to this problem is then used to reason about the boundaries. The most successful

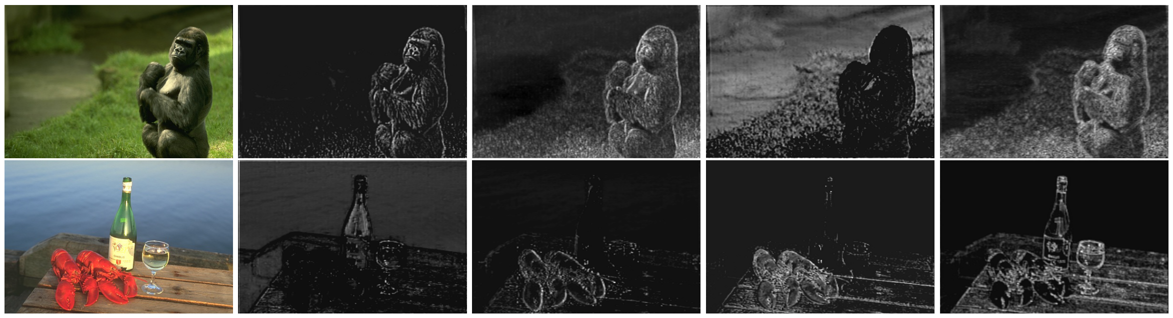


Figure 8: A visualization of selected convolutional feature maps from VGG network (resized to the input image dimension). Because VGG was optimized for an object classification task, it produces high activation values on objects and their parts.

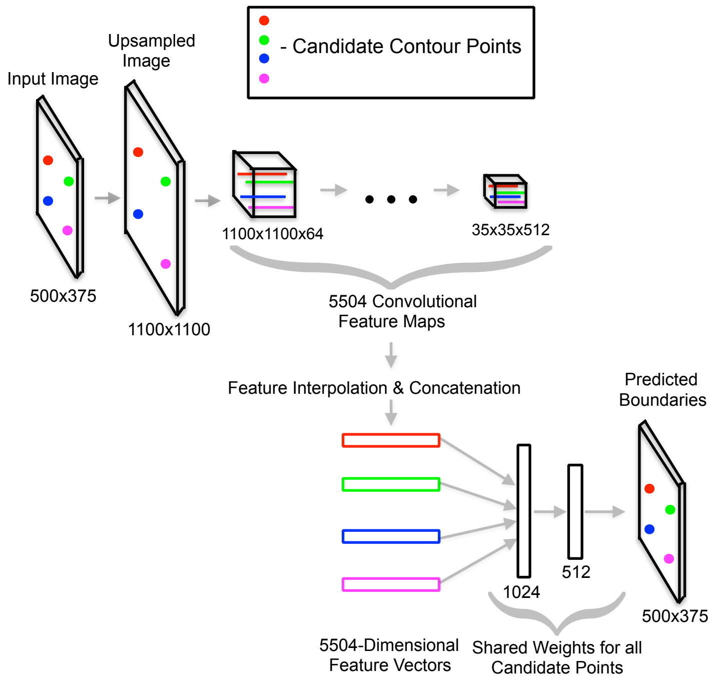
approaches in this genre are the MCG detector [38], gPb detector [30], PMI detector [42], and Normalized Cuts [41].

Some of the notable discriminative boundary detection methods include sketch tokens (ST) [37], structured edges (SE) [15] and sparse code gradients (SCG) [39]. While SCG use supervised SVM learning [40], the latter two methods rely on a random forest classifier and models boundary detection as a classification task.

Recently there have been attempts to apply deep learning to the task of boundary detection. SCT [43] is a sparse coding approach that reconstructs an image using a learned dictionary and then detect boundaries. Both N^4 fields [12] and DeepNet [13] approaches use Convolutional Neural Networks (CNNs) to predict edges. N^4 fields rely on dictionary learning and the use of the Nearest Neighbor algorithm within a CNN framework while DeepNet uses a traditional CNN architecture to predict contours.

The most similar to our approach is DeepEdge [1], which uses a multi-scale bifurcated network to perform contour detection using object-level features. However, we show that our method achieves better results even without the complicated multi-scale and bifurcated architecture of DeepEdge. Additionally, unlike DeepEdge, our system can run in near-real time.

In comparison to prior approaches, we offer several contributions. First, we propose to



- Candidate Contour Points
Input Image
Upsampled Image
500x375
1100x1100
1100x1100x64
35x35x512
5504 Convolutional Feature Maps
Feature Interpolation & Concatenation
Predicted Boundaries
1024
512
500x375
5504-Dimensional Feature Vectors
Shared Weights for all Candidate Points

Figure 9: An illustration of our architecture (best viewed in color). First we extract a set of candidate contour points. Then we upsample the image and feed it through 16 convolutional layers pretrained for object classification. For each candidate point, we find its correspondence in each of the feature maps and perform feature interpolation. This yields a 5504-dimensional feature vector for each candidate point. We feed each of these vectors to two fully connected layers and store the predictions to produce a final boundary map.

use object-level information to predict boundaries. We argue that such an approach leads to semantic boundaries, which are more consistent with humans reasoning. Second, we avoid feature engineering by learning boundaries from human-annotated data. Finally, we demonstrate excellent results for both low-level and high-level vision tasks. For the boundary detection task, our proposed HfL boundaries outperform all of the prior methods according to both F-score metrics. Additionally, we show that because HfL boundaries are based on object-level features, they can be used to improve performance in the high-level vision tasks of semantic boundary labeling, semantic segmentation, and object proposal generation.

2.3. Boundary Detection

In this section, we describe our proposed architecture and the specific details on how we predict HfL boundaries using our method. The detailed illustration of our architecture is presented in Figure 9.

2.3.1. Technical Approach

Selection of Candidate Contour Points. We first extract a set of candidate contour points with a high recall. Due to its efficiency and high recall performance, we use the SE edge detector [15]. In practice, we could eliminate this step and simply try to predict boundaries at every pixel. However, selecting a set of initial candidate contour points, greatly reduces the computational cost. Since our goal is to build a boundary detector that is both accurate and efficient, we use these candidate points to speed up the computation of our method.

Object-Level Features. After selecting candidate contour points, we up-sample the original input image to a larger dimension (for example 1100×1100). The up-sampling is done to minimize the loss of information due to the input shrinkage caused by pooling at the different layers. Afterwards, we feed the up-sampled image through 16 convolutional layers of the VGG net.

We use the VGG net as our model because it has been trained to recognize a large number of object classes (the 1000 categories of the ImageNet dataset [45]) and thus encodes object-level features that apply to many classes. To preserve specific location information we utilize only the 16 convolutional layers of the VGG net. We don't use fully connected layers because they don't preserve spatial information, which is crucial for accurate boundary detection.

We visualize some of the selected convolutional maps in Figure 8. Note the high activation values around the various objects in the images, which confirms our hypothesis that the

VGG net encodes object specific information in its convolutional feature maps.

Feature Interpolation. Similarly to [54, 55, 27], we perform feature interpolation in deep layers. After the up-sampled image passes through all 16 convolutional layers, for each selected candidate contour point we find its corresponding point in the feature maps. Due to the dimension differences in convolutional maps these correspondences are not exact. Thus we perform feature interpolation by finding the four nearest points and averaging their activation values. This is done in each of the 5504 feature maps. Thus, this results in a 5504-dimensional vector for each candidate point.

We note that the interpolation of convolutional feature maps is the crucial component that enables our system to predict the boundaries efficiently. Without feature interpolation, our method would need to independently process the candidate edge points by analyzing a small image patch around each point, as for example done in DeepEdge [1] which feeds one patch at a time through a deep network. However, when the number of candidate points is large (e.g., DeepEdge considers about 15K points at each of 4 different scales), their patches overlap significantly and thus a large amount of computation is wasted by recalculating filter response values over the same pixels. Instead, we can compute the features for all candidate points with a single pass through the network by performing deep convolution over the *entire* image (i.e., feeding the entire image rather than one patch at a time) and then by interpolating the convolutional feature maps at the location of each candidate edge point so as to produce its feature descriptor. Thanks to this speedup, our method has a runtime of 1.2 seconds (using a K40 GPU), which is better than the runtimes of prior deep-learning based edge detection methods [11, 12, 13, 1].

Learning to Predict Boundaries. After performing feature interpolation, we feed the 5504-dimensional feature vectors corresponding to each of the candidate contour points to two fully connected layers that are optimized to the human agreement criterion. To be more precise, we define our prediction objective as a fraction of human annotators agreeing on the presence of the boundary at a particular pixel. Therefore, a learning objective aims at mimicking the judgement of the human labelers.

Finally, to detect HfL boundaries, we accumulate the predictions from the fully connected layers for each of the candidate points and produce a boundary probability map as illustrated in Figure 9.

2.3.2. Implementation Details

In this section, we describe the details behind the training procedure of our model. We use the Caffe library [16] to implement our network architecture.

In the training stage, we freeze the weights in all of the convolutional layers. To learn the weights in the two fully connected layers we train our model to optimize the least squares error of the regression criterion that we described in the previous subsection. To enforce regularization we set a dropout rate of 0.5 in the fully connected layers.

Our training dataset includes $80K$ points from the BSDS500 dataset [50]. As described in the previous subsection, the labels represent the fraction of human annotators agreeing on the boundary presence. We divide the label space into four quartiles, and select an equal number of samples for each quartile to balance the training dataset. In addition to the training dataset, we also sample a hold-out dataset of size $40,000$. We use this for the hard-positive mining [49] in order to reduce the number of false-negative predictions.

For the first 25 epochs we train the network on the original $80,000$ training samples. After the first 25 epochs, we test the network on the hold-out dataset and detect false negative predictions made by our network. We then augment the original $80,000$ training samples with the false negatives and the same number of randomly selected true negatives. For the remaining 25 epochs, we train the network on this augmented dataset.

2.3.3. Boundary Detection Results

In this section, we present our results on the BSDS500 dataset [50], which is the most established benchmark for boundary detection. The quality of the predicted boundaries is

Consensus GT	*ODS*	*OIS*	*AP*	*FPS*
SCG [39]	0.6	0.64	0.56	1/280
DeepNet [13]	0.61	0.64	0.61	1/5‡
PMI [42]	0.61	**0.68**	0.56	1/900
DeepEdge [1]	0.62	0.64	0.64	1/1000‡
N^4-fields [12]	0.64	0.67	0.64	1/6‡
HfL	**0.65**	**0.68**	**0.67**	5/6‡

Any GT	*ODS*	*OIS*	*AP*	*FPS*
SE [15]	0.75	0.77	0.80	**2.5**
MCG [38]	0.75	0.78	0.76	1/24
N^4-fields [12]	0.75	0.77	0.78	1/6‡
DeepEdge [1]	0.75	0.77	**0.81**	1/1000‡
MSC [62]	0.76	0.78	0.79	-
DeepContour [11]	0.76	0.77	0.8	1/30‡
HfL	**0.77**	**0.79**	0.8	5/6‡

Table 7: Boundary detection results on BSDS500 benchmark. Upper half of the table illustrates the results for "consensus" ground-truth criterion while the lower half of the table depicts the results for "any" ground-truth criterion. In both cases, our method outperforms all prior methods according to both ODS (optimal dataset scale) and OIS (optimal image scale) metrics. We also report the run-time of our method (‡ GPU time) in the FPS column (frames per second), which shows that our algorithm is faster than prior approaches based on deep learning [11, 12, 13, 1].

evaluated using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP).

We compare our approach to the state-of-the-art based on two different sets of BSDS500 ground truth boundaries. First, we evaluate the accuracy by matching each of the predicted boundary pixels with the ground truth boundaries that were annotated by *any* of the human annotators. This set of ground truth boundaries is referred to as "any". We present the results for "any" ground truth boundaries in the lower half of Table 7. As indicated by the results, HfL boundaries outperform all the prior methods according to both F-score measures.

Recently, there has been some criticism raised about the procedure for boundary detection evaluation on the BSDS500 dataset. One issue with the BSDS500 dataset involves the so called "orphan" boundaries: the boundaries that are marked by only one or two human

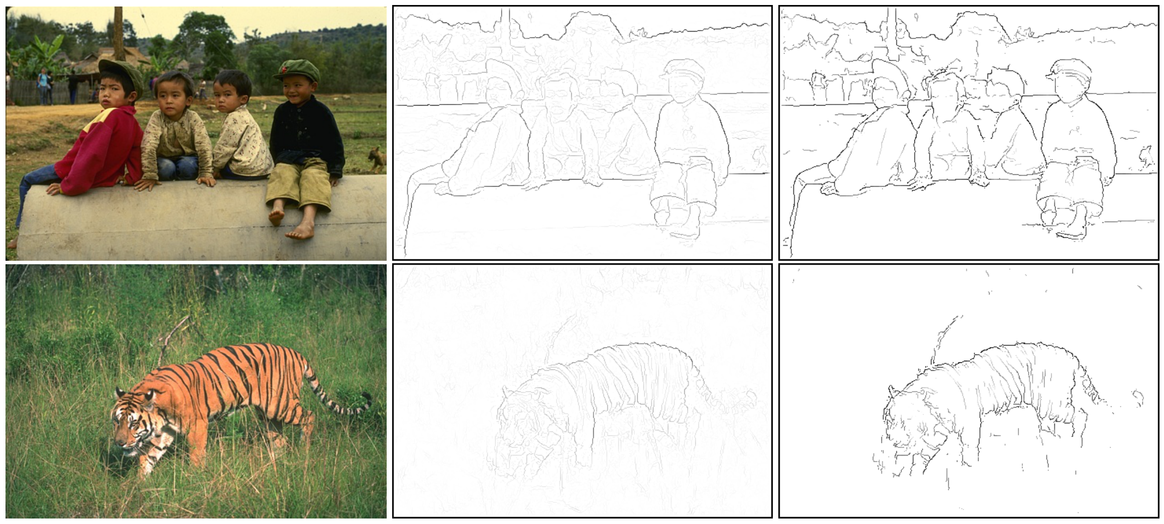


Figure 10: Qualitative results on the BSDS benchmark. The first column of images represent input images. The second column illustrates SE [15], while the third column depicts HfL boundaries. Notice that SE boundaries are predicted with low confidence if there is no significant change in color between the object and the background. Instead, because our model is defined in terms of object-level features, it can predict object boundaries with high confidence even if there is no significant color variation in the scene.

annotators. These "orphan" boundaries comprise around 30% of BSDS500 dataset but most of them are considered uninformative. However, the standard evaluation benchmark rewards the methods that predict these boundaries. To resolve this issue we also evaluate our HfL boundaries on the so called "consensus" set of ground truth boundaries. These "consensus" boundaries involve only boundaries that are marked by *all* of the human annotators and hence, are considered perceptually meaningful. In the upper half of Table 7, we present the results achieved by our method on the "consensus" set of the ground truth boundaries. Our HfL boundaries outperform or tie all the prior methods in each of the three evaluation metrics, thus suggesting that HfL boundaries are similar to the boundaries that humans annotated. We also report the runtimes in Table 7 and note that our method runs faster than previous deep-learning based edge detection systems [11, 12, 13, 1].

Our proposed model computes a highly nonlinear function of the 5504-dimensional feature vector of each candidate point. Thus, it is difficult to assess which features are used most heavily by our edge predictor. However, we can gain a better insight by replacing the

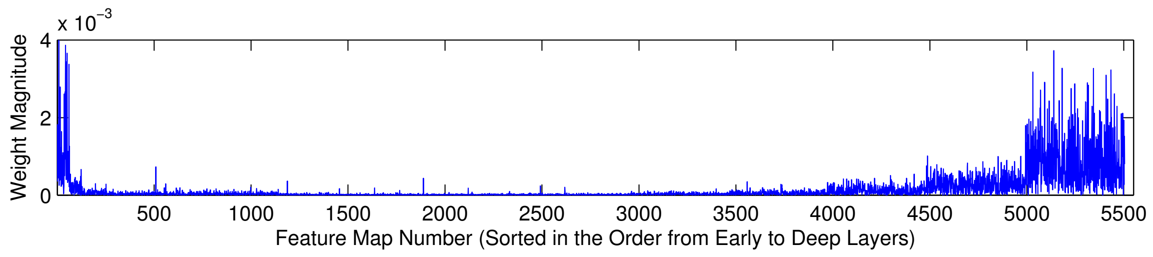


Figure 11: We train a linear regression model and visualize its weight magnitudes in order to understand which features are used most heavily in the boundary prediction (this linear regression is used only for the visualization purposes and not for the accuracy analysis). Note how heavily weighted features lie in the deepest layers of the network, i.e., the layers that are most closely associated with object information.

nonlinear function with a simple linear model. In Fig. 11 we show the weight magnitudes of a simple linear regression model (we stress that this linear model is used only for feature visualization purposes). From this Figure, we observe that many important features are located in the deepest layers of the VGG network. As shown in [63], these layers encode high-level object information, which confirms our hypothesis that high-level information is useful for boundary detection.

Finally, we present some qualitative results achieved by our method in Figure 10. These examples illustrate the effective advantage that HfL boundaries provide over another state-of-the-art edge detection system, the SE system [15]. Specifically, observe the parts of the image where there is a boundary that separates an object from the background but where the color change is pretty small. Notice that because the SE boundary detection is based on low-level color and texture features, it captures these boundaries with very low confidence. In comparison, because HfL boundaries rely on object-level features, it detects these boundaries with high confidence.

2.4. High-Level Vision Applications

In this section, we describe our proposed *Low-for-High* pipeline: using low-level boundaries to aid a number of high-level vision tasks. We focus on the tasks of semantic boundary labeling, semantic segmentation and object proposal generation. We show that using HfL

Method (Metric)	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
InvDet (MF)	42.6	49.5	15.7	16.8	36.7	43.0	40.8	22.6	18.1	26.6
HfL-FC8 (MF)	71.6	59.6	68.0	54.1	57.2	68.0	58.8	69.3	43.3	65.8
HfL-CRF (MF)	**73.9**	**61.4**	**74.6**	**57.2**	**58.8**	**70.4**	**61.6**	**71.9**	**46.5**	**72.3**
InvDet (AP)	38.4	29.6	9.6	9.9	24.2	33.6	31.3	17.3	10.7	16.4
HfL-FC8 (AP)	66.0	50.7	58.9	40.6	47.1	62.9	51.0	59.0	25.6	54.6
HfL-CRF (AP)	**71.2**	**55.2**	**69.3**	**45.7**	**48.9**	**71.1**	**56.8**	**65.7**	**29.1**	**65.9**

Method (Metric)	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
InvDet (MF)	10.2	18.0	35.2	29.4	48.2	14.3	26.8	11.2	22.2	32.0
HfL-FC8 (MF)	33.3	67.9	67.5	62.2	69.0	43.8	68.5	33.9	57.7	54.8
HfL-CRF (MF)	**36.2**	**71.1**	**73.0**	**68.1**	**70.3**	**44.4**	**73.2**	**42.6**	**62.4**	**60.1**
InvDet (AP)	3.7	12.1	28.5	20.4	45.7	7.6	16.1	5.7	14.6	22.7
HfL-FC8 (AP)	15.3	57.8	57.3	55.9	62.2	27.5	55.6	18.0	50.1	40.6
HfL-CRF (AP)	**17.7**	**64.5**	**68.3**	**64.7**	**65.9**	**29.1**	**66.5**	**25.7**	**60.0**	**49.8**

Table 8: Results of semantic boundary labeling on the SBD benchmark using the Max F-score (MF) and Average Precision (AP) metrics. Our method (HfL) outperforms Inverse Detectors [14] for all 20 categories according to both metrics. Note that using the CRF output to label the boundaries produces better results than using the outputs from the FC8 layer of FCN.

boundaries improves the performance of state-of-the-art methods in each of these high-level vision tasks.

2.4.1. Semantic Boundary Labeling

The task of semantic boundary labeling requires not only to predict the boundaries but also to associate a specific object class to each of the boundaries. This implies that given our predicted boundaries we also need to label them with object class information. We approach this problem by adopting the ideas from the recent work on Fully Convolutional Networks (FCN) [27]. Given an input image, we concurrently feed it to our boundary-predicting network (described in Section 2.3), and also through the FCN that was pretrained for 20 Pascal VOC classes and the background class. While our proposed network produces HfL boundaries, the FCN model predicts class probabilities for each of the pixels. We can then merge the two output maps as follows. For a given boundary point we consider a 9×9 grid around that point from each of the 21 FCN object-class probability maps. We calculate the maximum value inside each grid, and then label the boundary at a given pixel with the object-class that corresponds to the maximum probability across these 21 maps. We apply this procedure for each of the boundary points, in order to associate object-class labels to the boundaries. Note that we consider the grids around the boundary pixel because the output of the FCN has a poor localization, and considering the grids rather than individual pixels leads to higher accuracy.

We can also merge HfL boundaries with the state-of-the-art DeepLab-CRF segmentation [29] to obtain higher accuracy. We do this in a similar fashion as just described. First, around a given boundary point we extract a 9×9 grid from the DeepLab-CRF segmentation. We then compute the mode value in the grid (excluding the background class), and use the object-class corresponding to the mode value as a label for the given boundary point. We do this for each of the boundary points. By merging HfL boundaries and the output of FCN or DeepLab-CRF, we get semantic boundaries that are highly localized and also contain object-specific information.

Semantic Boundary Labeling Results

In this section, we present semantic boundary labeling results on the SBD dataset [14], which includes ground truth boundaries that are also labeled with one of 20 Pascal VOC classes. The boundary detection accuracy for each class is evaluated using the maximum F-score (MF), and average precision (AP) measures.

Labeling boundaries with the semantic object information is a novel and still relatively unexplored problem. Therefore, we found only one other approach (Inverse Detectors) that tried to tackle this problem [14]. The basic idea behind Inverse Detectors consists of several steps. First, generic boundaries in the image are detected. Then, a number of object proposal boxes are generated. These two sources of information are then used to construct the features. Finally, a separate classifier is used to label the boundaries with the object-specific information.

Table 8 shows that our approach significantly outperforms Inverse Detectors according to both the maximum F-score and the average precision metrics for all twenty categories. As

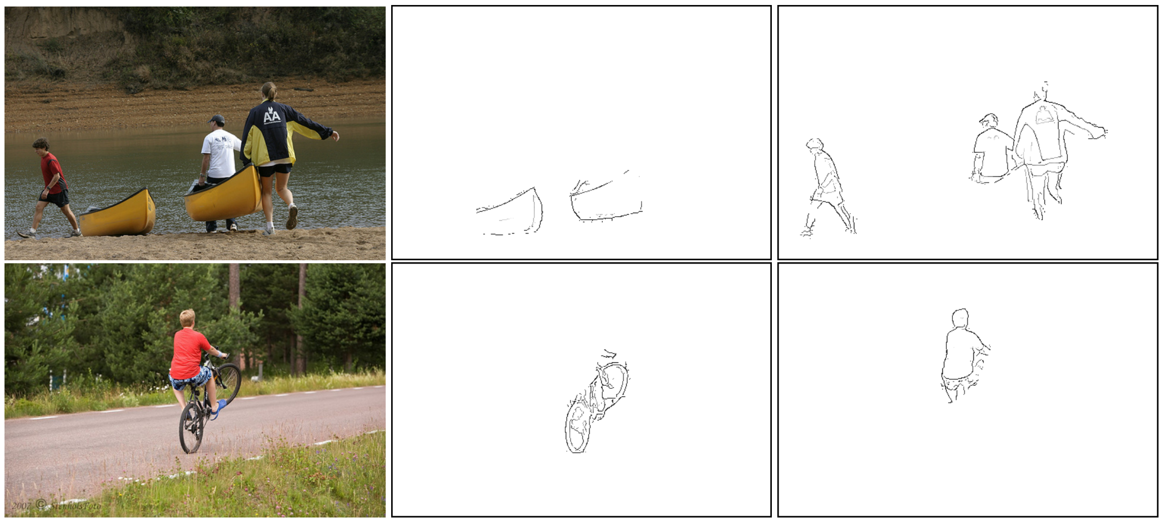


Figure 12: A visualization of the predicted semantic boundary labels. Images in the first column are input examples. Columns two and three show semantic HfL boundaries of different object classes. Note that even with multiple objects appearing simultaneously, our method outputs precise semantic boundaries.

described in Section 2.4.1 we evaluate the two variants of our method. Denoted by HfL-FC8 is the variant for which we label HfL boundaries with the outputs from the last layer (FC8) of the pretrained FCN. We denote with HfL-CRF the result of labeling our boundaries with the output from the DeepLab-CRF [29]. Among these two variants, we show that the latter one produces better results. This is expected since the CRF framework enforces spatial coherence in the semantic segments.

In Figure 12, we present some of the qualitative results produced by our method. We note that even with multiple objects in the image, our method successfully recognizes and localizes boundaries of each of the classes.

2.4.2. Semantic Segmentation

For the semantic segmentation task, we propose to enhance the DeepLab-CRF [29] with our predicted HfL boundaries. DeepLab-CRF is a system comprised of a Fully Convolutional Network (described in Section 2.4.1) and a dense CRF applied on top of FCN predictions.

Specifically, in the CRF, the authors propose to use a Gaussian kernel and a bilateral term

Metric	Method (Dataset)	mean
PP-IOU	DL-CRF (VOC)	**68.9**
	DL-CRF+HfL (VOC)	68.5
	DL-CRF (SBD)	**71.4**
	DL-CRF+HfL (SBD)	**71.4**

Metric	Method (Dataset)	mean
PI-IOU	DL-CRF (VOC)	44.6
	DL-CRF+HfL (VOC)	**46.5**
	DL-CRF (SBD)	45.8
	DL-CRF+HfL (SBD)	**48.8**

Table 9: Semantic segmentation results on the SBD and VOC 2007 datasets. We measure the results according to PP-IOU (per pixel) and PI-IOU (per image) evaluation metrics. We denote the original DeepLab-CRF system and our proposed modification as DL-CRF and DL-CRF+HfL, respectively. According to the PP-IOU metric, our proposed features (DL-CRF+HfL) yield almost equivalent results as the original DeepLab-CRF system. However, based on PI-IOU metric, our proposed features improve the mean accuracy by 3% and 1.9% on SBD and VOC 2007 datasets respectively.

including position and color terms as the CRF features (see [29]). While in most cases the proposed scheme works well, DeepLab-CRF sometimes produces segmentations that are not spatially coherent, particularly for images containing small object regions.

We propose to address this issue by adding features based on our predicted HfL boundaries in the CRF framework. Note that we use predicted boundaries from Section 2.3 and not the boundaries labeled with the object information that we obtained in Section 2.4.1. We use the Normalized Cut [41] framework to generate our features.

First, we construct a pixel-wise affinity matrix $\mathbf{W}$ using our HfL boundaries. We measure the similarity between two pixels as:

$$W_{ij} = \exp\left(-\max_{p \in \overline{ij}} \{\frac{M(p)^2}{\sigma^2}\}\right)$$

where W_{ij} represents the similarity between pixels i and j, p denotes the boundary point along the line segment $\overline{ij}$ connecting pixels i and j, M depicts the magnitude of the boundary at pixel p, and σ denotes the smoothness parameter, which is usually set to 14% of the maximum boundary value in the image.

The intuitive idea is that two pixels are similar (i.e. $W_{ij} = 1$) if there is no boundary crossing the line connecting these two pixels (i.e. $M(p) = 0 \quad \forall p \in \overline{ij}$) or if the boundary strength is low. We note that it is not necessary to build a full affinity matrix W. We build a sparse affinity matrix connecting every pair of pixels i and j that have distance 5 or less from each other.

After building a boundary-based affinity matrix $\mathbf{W}$ we set $D_{ii} = \sum_{i \neq j} W_{ij}$ and compute eigenvectors $\mathbf{v}$ of the generalized eigenvalue system:

$$(\mathbf{D} - \mathbf{W})\mathbf{v} = \lambda \mathbf{D}\mathbf{v}$$

We then resize the eigenvectors v to the original image dimensions, and use them as additional features to the CRF part of DeepLab-CRF system. In our experiments, we use the 16 eigenvectors corresponding to the smallest eigenvalues, which results in 16 extra feature channels.

Note that the eigenvectors contain soft segmentation information. Because HfL boundaries predict object-level contours with high confidence, the eigenvectors often capture regions corresponding to objects. We visualize a few selected eigenvectors in Figure 13. In the experimental section, we demonstrate that our proposed features make the output produced by DeepLab-CRF more spatially coherent and improve the segmentation accuracy according to one of the metrics.

We also note that our proposed features are applicable to any generic method that incorporates CRF. For instance, even if DeepLab-CRF used an improved DeepLab network architecture, our features would still be beneficial because they contribute directly to the CRF part and not the DeepLab network part of the system.

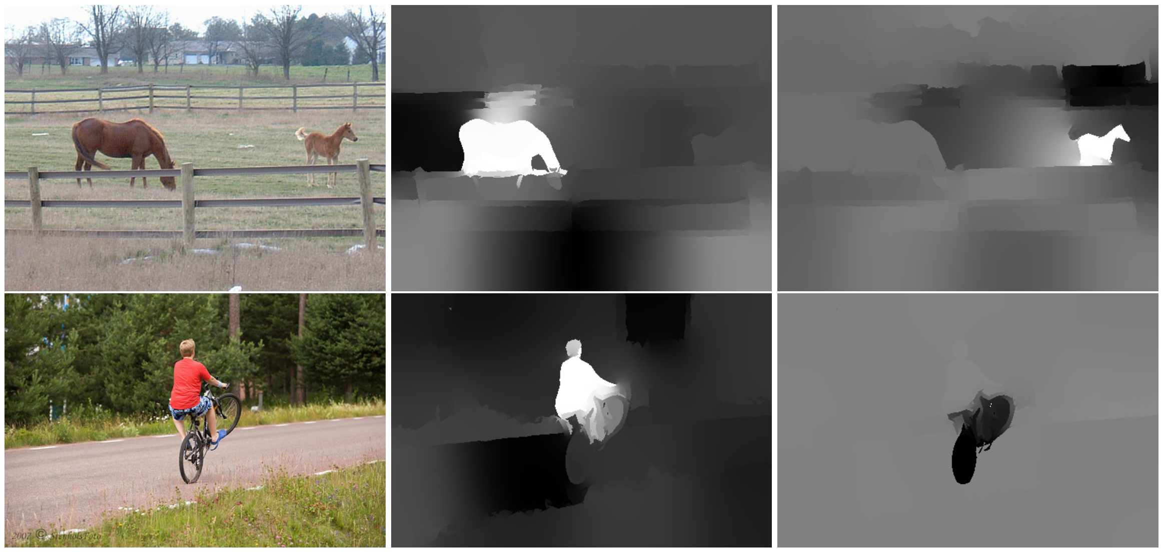


Figure 13: In this figure, the first column depicts an input image while the second and third columns illustrate two selected eigenvectors for that image. The eigenvectors contain soft segmentation information. Because HfL boundaries capture object-level boundaries, the resulting eigenvectors primarily segment regions corresponding to the objects.

Semantic Segmentation Results

In this section, we present semantic segmentation results on the SBD [14] and also Pascal VOC 2007 [64] datasets, which both provide ground truth segmentations for 20 Pascal VOC classes. We evaluate the results in terms of two metrics. The first metric measures the accuracy in terms of pixel intersection-over-union averaged per pixel (PP-IOU) across the 20 classes. According to this metric, the accuracy is computed on a per pixel basis. As a result, the images that contain large object regions are given more importance.

We observe that while DeepLab-CRF works well on the images containing large object regions, it produces spatially disjoint outputs for the images with smaller and object regions (see Figure 14). This issue is often being overlooked, because according to the PP-IOU metric, the images with large object regions are given more importance and thus contribute more to the accuracy. However, certain applications may require accurate segmentation of small objects. Therefore, in addition to PP-IOU, we also consider the PI-IOU metric (pixel intersection-over-union averaged per image across the 20 classes), which gives equal weight to each of the images.

For both of the metrics we compare the semantic segmentation results of a pure DeepLab-CRF [29] and also a modification of DeepLab-CRF with our proposed features added to the CRF framework. We present the results for both of the metrics in Table 9.

Based on these results, we observe that according to the first metric (PP-IOU), our proposed features yield almost equivalent results as the original DeepLab-CRF system. However, according to the second metric (PI-IOU) our features yield an average improvement of 3% and 1.9% in SBD and VOC 2007 datasets respectively.

We also visualize the qualitative results produced by both approaches in Figure 14. Notice how our proposed features make the segmentations look smoother relative to the segmentations produced by the original DeepLab-CRF system.

Once again, we want to stress that our HfL features are applicable to any method that uses the CRF. Therefore, based on the results presented in this section, we believe that our proposed features could be beneficial in a wide array of problems that involve the use of the CRF framework.

2.4.3. Object Proposals

Finally, we show that our method produces object-level boundaries that can be successfully exploited in an object proposal scheme. Specifically we adopt the EdgeBoxes approach [61], which can be applied to any generic boundaries to produce a list of object proposal boxes. The original EdgeBoxes method uses SE boundaries to generate the boxes. However, SE boundaries are predicted using low-level color and texture features, rather than object-level features. Thus, here we validate the hypothesis that the EdgeBoxes proposals can be improved by replacing the SE boundaries with our HfL boundaries.

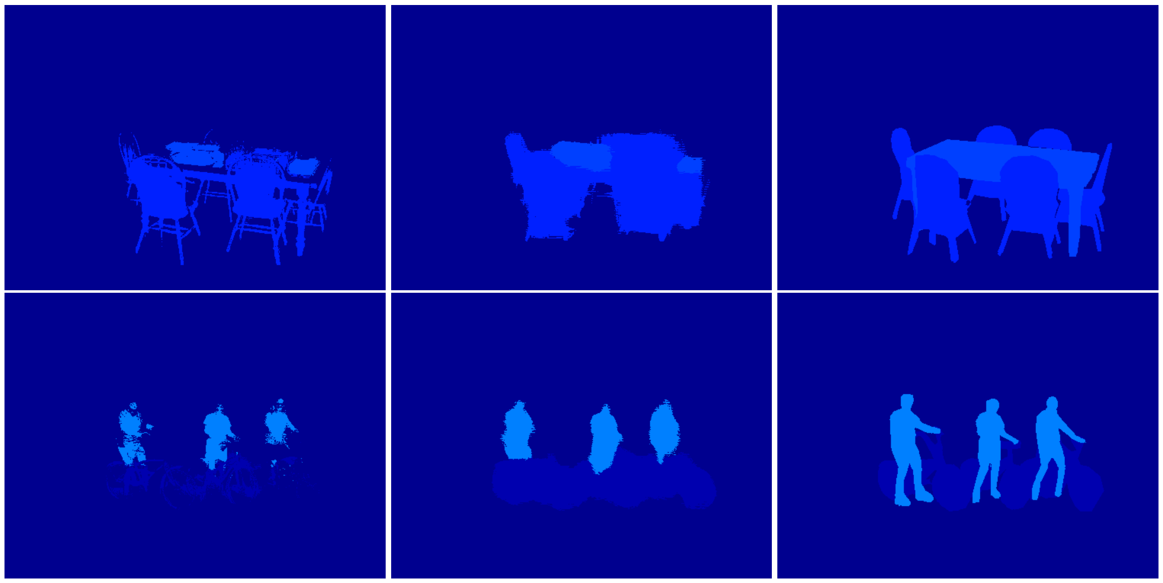


Figure 14: An illustration of the more challenging semantic segmentation examples. The first column depicts the predictions achieved by DeepLab-CRF, while the second column illustrates the results after adding our proposed features to the CRF framework. The last column represents ground truth segmentations. Notice how our proposed features render the predicted semantic segments more spatially coherent and overall more accurate.

Object Proposal Results

In this section, we present object proposal results on the Pascal VOC 2012 dataset [65]. We evaluate the quality of bounding-box proposals according to three metrics: area under the curve (AUC), the number of proposals needed to reach recall of 75%, and the maximum recall over 5000 object bounding-boxes. Additionally, we compute the accuracy for each of the metrics for three different intersection over union (IOU) values: $0.65, 0.7$, and 0.75. We present these results in Table 10. As described in Section 2.4.3, we use EdgeBoxes [61], a package that uses generic boundaries, to generate object proposals. We compare the quality of the generated object proposals when using SE boundaries and HfL boundaries. We demonstrate that for each IOU value and for each of the three evaluation metrics, HfL boundaries produce better or equivalent results. This confirms our hypothesis that HfL boundaries can be used effectively for high-level vision tasks such as generating object proposals.

Method	IoU 0.65			IoU 0.7			IoU 0.75		
	AUC	N@75%	Recall	AUC	N@75%	Recall	AUC	N@75%	Recall
SE	0.52	413	0.93	0.47	658	0.88	**0.41**	inf	0.75
HfL	**0.53**	**365**	**0.95**	**0.48**	**583**	**0.9**	**0.41**	**2685**	**0.77**

Table 10: Comparison of object proposal results. We compare the quality of object proposals using Structured Edges [15] and HfL boundaries. We evaluate the performance for three different IOU values and demonstrate that using HfL boundaries produces better results for each evaluation metric and for each IOU value.

2.5. Conclusions

In this work, we presented an efficient architecture that uses object-level information to predict semantically meaningful boundaries. Most prior edge detection methods rely exclusively on low-level features, such as color or texture, to detect the boundaries. However, perception studies suggest that humans employ object-level reasoning when deciding whether a given pixel is a boundary [57, 58, 59]. Thus, we propose a system that focuses on the semantic object-level cues rather than low level image information to detect the boundaries. For this reason we refer to our boundary detection scheme as a *High-for-Low* approach, where high-level object features inform the low-level boundary detection process. In this paper we demonstrated that our proposed method produces boundaries that accurately separate objects and the background in the image and also achieve higher F-score compared to any prior work.

Additionally, we showed that, because HfL boundaries are based on object-level features, they can be employed to aid a number of high level vision tasks in a *Low-for-High* fashion. We use our boundaries to boost the accuracy of state-of-the-art methods on the high-level vision tasks of semantic boundary labeling, semantic segmentation, and object proposals generation. Using HfL boundaries leads to better results in each of these tasks.

To conclude, our boundary detection method is accurate, efficient, applicable to a variety of datasets, and also useful for multiple high-level vision tasks. We plan to release the source code for HfL upon the publication of the paper .

CHAPTER 3 : Semantic Segmentation with Boundary Neural Fields

3.1. Introduction

The recent introduction of fully convolutional networks (FCNs) [27] has led to significant quantitative improvements on the task of semantic segmentation. However, despite their empirical success, FCNs suffer from some limitations. Large receptive fields in the convolutional layers and the presence of pooling layers lead to blurring and segmentation predictions at a significantly lower resolution than the original image. As a result, their predicted segments tend to be blobby and lack fine object boundary details. We report in Fig. 15 some examples illustrating typical poor localization of objects in the outputs of FCNs.

Recently, Chen at al. [29] addressed this issue by applying a Dense-CRF post-processing step [28] on top of coarse FCN segmentations. However, such an approach introduces several problems of its own. First, the Dense-CRF adds new parameters that are difficult to tune and integrate into the original network architecture. Additionally, most methods based on CRFs or MRFs use low-level pixel affinity functions, such as those based on color. These low-level affinities often fail to capture semantic relationships between objects and lead to poor segmentation results (see last column in Fig. 15).

We propose to address these shortcomings by means of a Boundary Neural Field (BNF), an architecture that employs a single semantic segmentation FCN to predict semantic boundaries and then use them to produce semantic segmentation maps via a global optimization. We demonstrate that even though the semantic segmentation FCN has not been optimized to detect boundaries, it provides good features for boundary detection. Specifically, the contributions of our work are as follows:

- We show that semantic boundaries can be expressed as a linear combination of interpolated convolutional feature maps inside an FCN. We introduce a boundary detection method that exploits this intuition to predict object boundaries with accuracy supe-

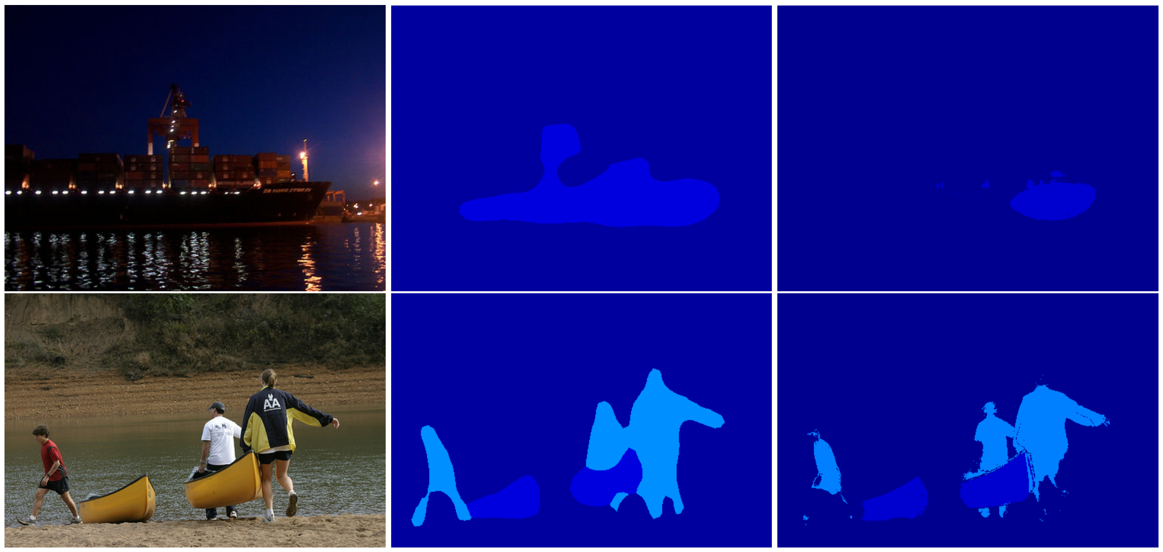


Figure 15: Examples illustrating shortcomings of prior semantic segmentation methods: the second column shows results obtained with a FCN [27], while the third column shows the output of a Dense-CRF applied to FCN predictions [28, 29]. Segments produced by FCN are blob-like and are poorly localized around object boundaries. Dense-CRF produces spatially disjoint object segments due to the use of a color-based pixel affinity function that is unable to measure semantic similarity between pixels.

rior to the state-the-of-art.

- We demonstrate that boundary-based pixel affinities are better suited for semantic segmentation than the commonly used color affinity functions.

- Finally, we introduce a new global energy that decomposes semantic segmentation into multiple binary problems and relaxes the integrality constraint. We show that minimizing our proposed energy yields better qualitative and quantitative results relative to traditional globalization models such as MRFs or CRFs.

3.2. Related Work

Boundary Detection. Spectral methods comprise one of the most prominent categories for boundary detection. In a typical spectral framework, one formulates a generalized eigenvalue system to solve a low-level pixel grouping problem. The resulting eigenvectors are then used to predict the boundaries. Some of the most notable approaches in this genre are MCG [38], gPb [30], PMI [42], and Normalized Cuts [41]. A weakness of spectral approaches is that they are slow as they perform a global inference over the entire image.

To address this issue, recent approaches cast boundary detection as a classification problem and predict the boundaries in a local manner with high efficiency. The most notable examples in this genre include sketch tokens (ST) [37] and structured edges (SE) [15], which employ fast random forests. However, many of these methods are based on hand-constructed features, which are difficult to tune.

The issue of hand-constructed features have been recently addressed by several approaches based on deep learning, such as N^4 fields [12], DeepNet [13], DeepContour [11], DeepEdge [1], HfL [2] and HED [66]. All of these methods use CNNs in some way to predict the boundaries. Whereas DeepNet and DeepContour optimize ordinary CNNs to a boundary based optimization criterion from scratch, DeepEdge and HfL employ pretrained models to compute boundaries. The most recent of these methods is HED [66], which shows the benefit of deeply supervised learning for boundary detection.

In comparison to prior deep learning approaches, our method offers several contributions. First, we exploit the inherent relationship between boundary detection and semantic segmentation to predict semantic boundaries. Specifically, we show that even though the semantic FCN has not been explicitly trained to predict boundaries, the convolutional filters inside the FCN provide good features for boundary detection. Additionally, unlike DeepEdge [1] and HfL [2], our method does not require a pre-processing step to select candidate contour points, as we predict boundaries on all pixels in the image. We demonstrate that our approach allows us to achieve state-of-the-art boundary detection results according to both F-score and Average Precision metrics. Additionally, due to the semantic nature of our boundaries, we can successfully use them as pairwise potentials for semantic segmentation in order to improve object localization and recover fine structural details, typically lost by pure FCN-based approaches.

Semantic Segmentation. We can group most semantic segmentation methods into three broad categories. The first category can be described as “two-stage” approaches, where an image is first segmented and then each segment is classified as belonging to a certain object class. Some of the most notable methods that belong to this genre include [67, 68, 69, 70].

The primary weakness of the above methods is that they are unable to recover from errors made by the segmentation algorithm. Several recent papers [55, 71] address this issue by proposing to use deep per-pixel CNN features and then classify each pixel as belonging to a certain class. While these approaches partially address the incorrect segmentation problem, they perform predictions independently on each pixel. This leads to extremely local predictions, where the relationships between pixels are not exploited in any way, and thus the resulting segmentations may be spatially disjoint.

The third and final group of semantic segmentation methods can be viewed as front-to-end schemes where segmentation maps are predicted directly from raw pixels without any intermediate steps. One of the earliest examples of such methods is the FCN introduced in [27]. This approach gave rise to a number of subsequent related approaches which have improved various aspects of the original semantic segmentation [29, 72, 73, 74, 75]. There have also been attempts at integrating the CRF mechanism into the network architecture [29, 72]. Finally, it has been shown that semantic segmentation can also be improved using additional training data in the form of bounding boxes [73].

Our BNF offers several contributions over prior work. To the best of our knowledge, we are the first to present a model that exploits the relationship between boundary detection and semantic segmentation **within a FCN framework**. We introduce pairwise pixel affinities computed from semantic boundaries inside an FCN, and use these boundaries to predict the segmentations in a global fashion. Unlike [75], which requires a large number of additional parameters to learn for the pairwise potentials, our global model only needs $\approx 5K$ extra parameters, which is about 3 orders of magnitudes less than the number of parameters in a typical deep convolutional network (e.g. VGG [60]). We empirically show that our

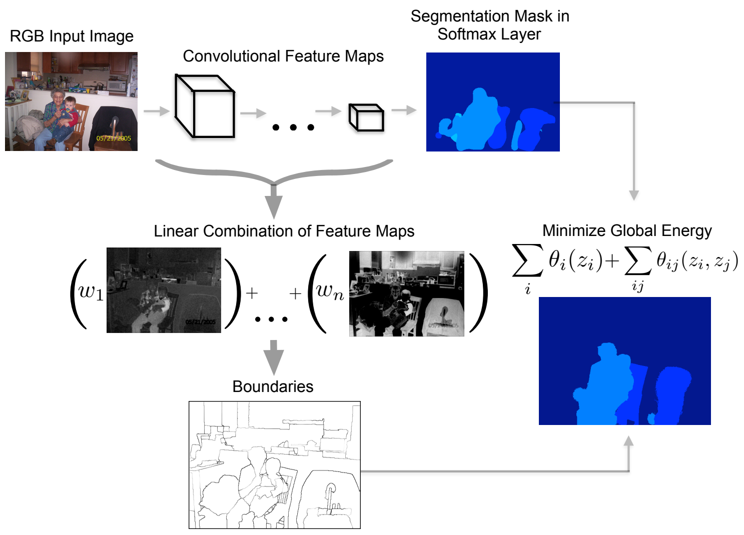

RGB Input Image
Convolutional Feature Maps
Segmentation Mask in Softmax Layer
Linear Combination of Feature Maps
w_1
w_n
Minimize Global Energy
$\sum_i \theta_i(z_i) + \sum_{ij} \theta_{ij}(z_i, z_j)$
Boundaries

Figure 16: We employ a semantic segmentation FCN [29] for two purposes: 1) to obtain semantic segmentation unaries for our global energy; 2) to compute object boundaries. Specifically, we define semantic boundaries as a linear combination of these feature maps (with a sigmoid function applied on top of the sum) and learn individual weights corresponding to each convolutional feature map. We integrate this boundary information in the form of pairwise potentials (pixel affinities) for our energy model.

proposed boundary-based affinities are better suited for semantic segmentation than color-based affinities. Additionally, unlike in [29, 72, 75], the solution to our proposed global energy can be obtained in closed-form, which makes global inference easier. Finally we demonstrate that our method produces better results than traditional globalization models such as CRFs or MRFs.

3.3. Boundary Neural Fields

In this section, we describe Boundary Neural Fields. Similarly to traditional globalization methods, Boundary Neural Fields are defined by an energy including unary and pairwise potentials. Minimization of the global energy yields the semantic segmentation. BNFs build both unary and pairwise potentials from the input RGB image and then combine them in a global manner. More precisely, the coarse segmentations predicted by a semantic FCN are used to define the unary potentials of our BNF. Next, we show that the convolutional feature maps of the FCN can be used to accurately predict semantic boundaries. These boundaries are then used to build pairwise pixel affinities, which are used as pairwise potentials by the BNF. Finally, we introduce a global energy function, which minimizes the energy corresponding to the unary and pairwise terms and improves the initial FCN segmentation. The detailed illustration of our architecture is presented in Figure 16. We now explain each of these steps in more detail.

3.3.1. FCN Unary Potentials

To predict semantic unary potentials we employ the DeepLab model [29], which is a fully convolutional adaptation of the VGG network [60]. The FCN consists of 16 convolutional layers and 3 fully convolutional layers. There are more recent FCN-based methods that have demonstrated even better semantic segmentation results [73, 72, 74, 75]. Although these more advanced architectures could be integrated into our framework to improve our unary potentials, in this work we focus on two aspects orthogonal to this prior work: 1) demonstrating that our boundary-based affinity function is better suited for semantic segmentation than the common color-based affinities and 2) showing that our proposed global energy achieves better qualitative and quantitative semantic segmentation results in comparison to prior globalization models.

3.3.2. Boundary Pairwise Potentials

In this section, we describe our approach for building pairwise pixel affinities using semantic boundaries. The basic idea behind our boundary detection approach is to express semantic boundaries as a function of convolutional feature maps inside the FCN. Due to the close relationship between the tasks of semantic segmentation and boundary detection, we hypothesize that convolutional feature maps from the semantic segmentation FCN can be employed as features for boundary detection.

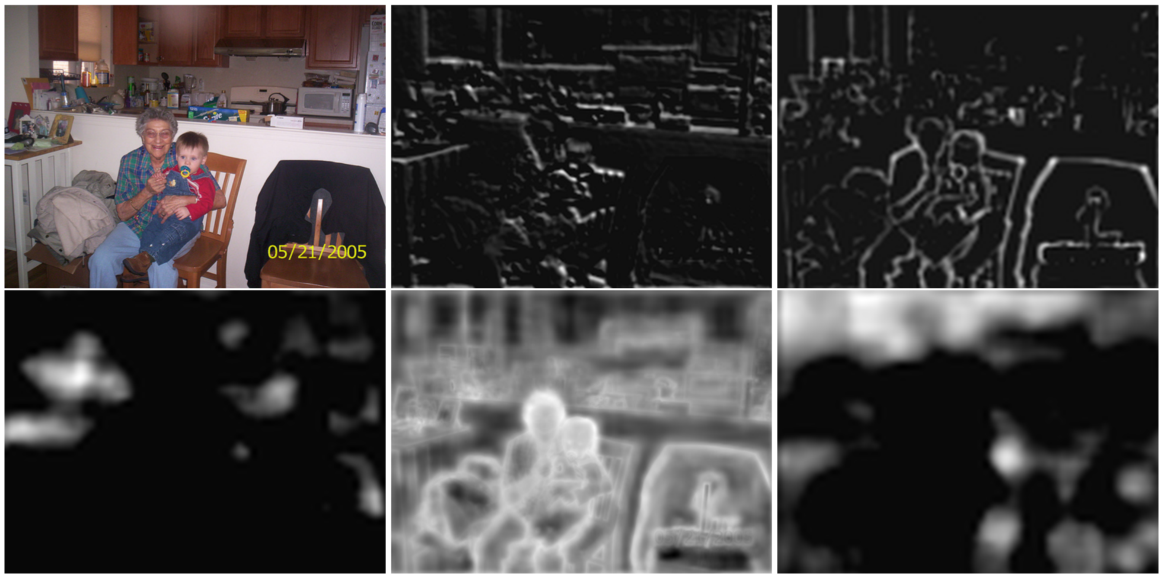


Figure 17: An input image and convolutional feature maps corresponding to the largest weight magnitude values. Intuitively these are the feature maps that contribute most heavily to the task of boundary detection.

Learning to Predict Semantic Boundaries.

We propose to express semantic boundaries as a linear combination of interpolated FCN feature maps with a non-linear function applied on top of this sum. We note that interpolation of feature maps has been successfully used in prior work (see e.g. [55]) in order to obtain dense pixel-level features from the low-resolution outputs of deep convolutional layers. Here we adopt interpolation to produce pixel-level boundary predictions. There are several advantages to our proposed formulation. First, because we express boundaries as a linear combination of feature maps, we only need to learn a small number of parameters, corresponding to the individual weight values of each feature map in the FCN. This amounts to $\approx 5K$ learning parameters, which is much smaller than the number of parameters in the entire network ($\approx 15M$). In comparison, DeepEdge [1] and HFL [2] need 17M and 6M *additional* parameters to predict boundaries.

Additionally, expressing semantic boundaries as a linear combination of FCN feature maps allows us to efficiently predict boundary probabilities for all pixels in the image (we resize the FCN feature maps to the original image dimensions). This eliminates the need to select candidate boundary points in a pre-processing stage, which was instead required in prior boundary detection work [1, 2].

Our boundary prediction pipeline can be described as follows. First we use use SBD segmentations [14] to optimize our FCN for semantic segmentation task. We then treat FCN convolutional maps as features for the boundary detection task and use the boundary annotations from BSDS 500 dataset [50] to learn the weights for each feature map. BSDS 500 dataset contains 200 training, 100 validation, 200 testing images, and ground truth annotations by 5 human labelers for each of these images.

To learn the weights corresponding to each convolutional feature map we first sample $80K$ points from the dataset. We define the target labels for each point as the fraction of human annotators agreeing on that point being a boundary. To fix the issue of label imbalance (there are many more non-boundaries than boundaries), we divide the label space into four quartiles, and select an equal number of samples for each quartile to balance the training dataset. Given these sampled points, we then define our features as the values in the interpolated convolutional feature maps corresponding to these points. To predict semantic boundaries we weigh each convolutional feature map by its weight, sum them up and apply a sigmoid function on top of it. We obtain the weights corresponding to each convolutional feature map by minimizing the cross-entropy loss using a stochastic batch gradient descent for 50 epochs. To obtain crisper boundaries at test-time we post-process the boundary probabilities using non-maximum suppression.

To give some intuition on how FCN feature maps contribute to boundary detection, in Fig. 17 we visualize the feature maps corresponding to the highest weight magnitudes. It is clear that many of these maps contain highly localized boundary information.

Boundary Detection Results Before discussing how boundary information is integrated in our energy for semantic segmentation, here we present experimental results assessing

Method	ODS	OIS	AP
SCG [39]	0.739	0.758	0.773
SE [15]	0.746	0.767	0.803
MCG [38]	0.747	0.779	0.759
N^4-fields [12]	0.753	0.769	0.784
DeepEdge [1]	0.753	0.772	0.807
DeepContour [11]	0.756	0.773	0.797
HfL [2]	0.767	0.788	0.795
HED [66]	0.782	0.804	0.833
BNF	**0.788**	**0.807**	**0.851**

Table 11: Boundary detection results on BSDS500 benchmark. Our proposed method outperforms all prior algorithms according to all three evaluation metrics.

the accuracy of our boundary detection scheme. We tested our boundary detector on the BSDS500 dataset [50], which is the standard benchmark for boundary detection. The quality of the predicted boundaries is evaluated using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP).

In Table 11 we show that our algorithm outperforms all prior methods according to both F-score measures and the Average Precision metric. In Fig. 18, we also visualize our predicted boundaries. The second column shows the pixel-level softmax output computed from the linear combination of feature maps, while the third column depicts our final boundaries after applying a non-maximum suppression post-processing step.

We note that our predicted boundaries achieve high-confidence predictions around objects. This is important as we employ these boundaries to improve semantic segmentation results, as discussed in the next subsection.

Constructing Pairwise Pixel Affinities.

We can use the predicted boundaries to build pairwise pixel affinities. Intuitively, we declare two pixels as similar (i.e., likely to belong to the same segment) if there is no boundary crossing the straight path between these two pixels. Conversely, two pixels are dissimilar if there is a boundary crossing their connecting path. The larger the boundary magnitude

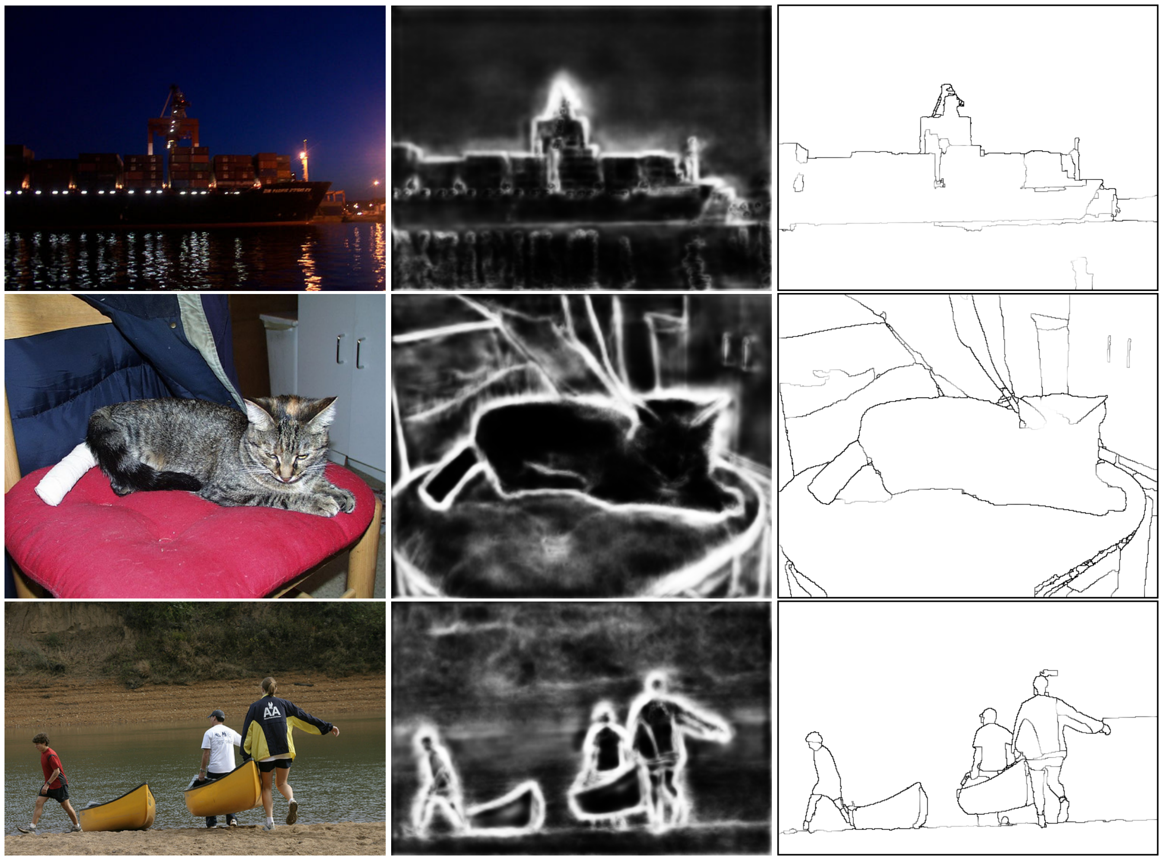


Figure 18: A figure illustrating our boundary detection results. In the second column, we visualize the raw probability output of our boundary detector. In the third column, we present the final boundary maps after non-maximum suppression. While most prior methods predict the boundaries where the sharpest change in color occurs, our method captures semantic object-level boundaries, which we subsequently use to aid semantic segmentation.

of the crossed path, the more dissimilar the two pixels should be, since a strong boundary is likely to mark the separation of two distinct segments. Similarly to [30], we encode this intuition with a following formulation:

$$w_{ij}^{sb} = \exp\left(\frac{-M_{ij}}{\sigma_{sb}}\right) \tag{3.1}$$

where M_{ij} denotes the maximum boundary value that crosses the straight line path between pixels i and j, σ_{sb} depicts the smoothing parameter and w_{ij}^{sb} denotes the semantic boundary-based affinity between pixels i and j.

Similarly, we want to exploit high-level object information in the network to define another type of pixel similarity. Specifically, we use object class probabilities from the *softmax* (SM) layer to achieve this goal. Intuitively, if pixels i and j have different hard segmentation labels from the *softmax* layer, we set their similarity (w_{ij}^{sm}) to 0. Otherwise, we compute their similarity using the following equation:

$$w_{ij}^{sm} = \exp\left(\frac{-D_{ij}}{\sigma_{sm}}\right) \tag{3.2}$$

where D_{ij} denotes the difference in *softmax* output values corresponding to the most likely object class for pixels i and j, and σ_{sm} is a smoothing parameter. Then we can write the final affinity measure as:

$$w_{ij} = \exp\left(w_{ij}^{sm}\right) w_{ij}^{sb} \tag{3.3}$$

We exponentiate the term corresponding to the object-level affinity because our boundary-based affinity may be too aggressive in declaring two pixels as dissimilar. To address this issue, we increase the importance of the object-level affinity in (3.3) using the exponential function. However, in the experimental results section, we demonstrate that most of the benefit from modeling pairwise potentials comes from w_{ij}^{sb} rather than w_{ij}^{sm}.

We then use this pairwise pixel affinity measure to build a global affinity matrix W that encodes relationships between pixels in the entire image. For a given pixel, we sample $\approx 10\%$ of points in the neighborhood of radius 20 around that pixel, and store the resulting affinities into W.

3.3.3. Global Inference

The last step in our proposed method is to combine semantic boundary information with the coarse segmentation from the FCN *softmax* layer to produce an improved segmentation. We do this by introducing a global energy function that utilizes the affinity matrix constructed in the previous section along with the segmentation from the FCN *softmax* layer. Using this energy, we perform a global inference to get segmentations that are well localized around the object boundaries and that are also spatially smooth.

Typical globalization models such as MRFs [76], CRFs [28] or Graph Cuts [77] produce a discrete label assignment for the segmentation problem by jointly modeling a multi-label distribution and solving a non-convex optimization. The common problem in doing so is that the optimization procedure may get stuck in local optima.

We introduce a new global energy function, which overcomes this issue and achieves better segmentation in comparison to prior globalization models. Similarly to prior globalization approaches, our goal is to minimize the energy corresponding to the sum of unary and pairwise potentials. However, the key difference in our approach comes from the relaxation of some of the constraints. Specifically, instead of modeling our problem as a joint multi-label distribution, we propose to decompose it into multiple binary problems, which can be solved concurrently. This decomposition can be viewed as assigning pixels to foreground and background labels for each of the different object classes. Additionally, we relax the integrality constraint. Both of these relaxations make our problem more manageable and allow us to formulate a global energy function that is differentiable, and has a closed form

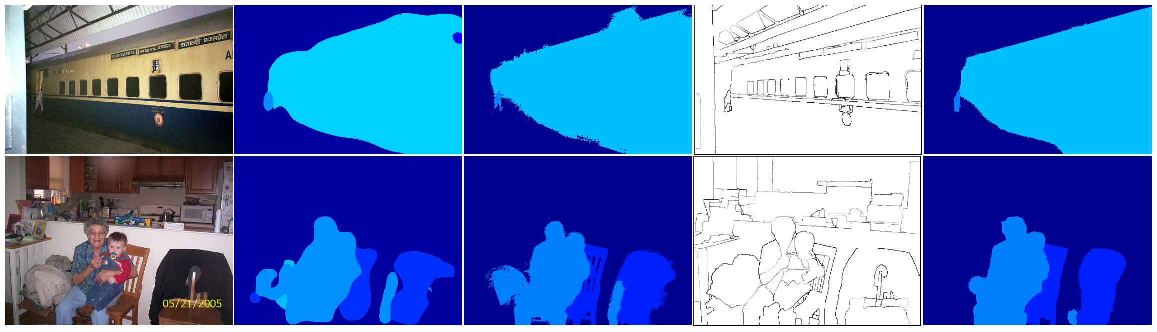


Figure 19: A figure illustrating semantic segmentation results. Images in columns two and three represent FCN*softmax* and Dense-CRF predictions, respectively. Note that all methods use the same FCN unary potentials. Additionally, observe that unlike FCN and Dense-CRF, our methods predicts segmentation that are both well localized around object boundaries and that are also spatially smooth.

solution.

In [78], the authors introduce the idea of learning with global and local consistency in the context of semi-supervised problems. Inspired by this work, we incorporate some of these ideas in the context of semantic segmentation. Before defining our proposed global energy function, we introduce some relevant notation.

For the purpose of illustration, suppose that we only have two classes: foreground and background. Then we can denote an optimal continuous solution to such a segmentation problem with variable z^*. To denote similarity between pixels i and j we use w_{ij}. Then, d_i indicates the degree of a pixel i. In graph theory, the degree of a node denotes the number of edges incident to that node. Thus, we set the degree of a pixel to $d_i = \sum_{j=1}^{n} w_{ij}$ for all j except $i \neq j$. Finally, with f_i we denote an initial segmentation probability, which in our case is obtained from the FCN *softmax* layer.

Using this notation, we can then formulate our global inference objective as:

$$z^* = \underset{z}{\operatorname{argmin}} \frac{\mu}{2} \sum_i d_i (z_i - \frac{f_i}{d_i})^2 + \frac{1}{2} \sum_{ij} w_{ij} (z_i - z_j)^2 \tag{3.4}$$

This energy consists of two different terms. Similar to the general globalization framework, our first term encodes the unary energy while the second term includes the pairwise energy. We now explain the intuition behind each of these terms. The unary term attempts to find a segmentation assignment (z_i) that deviates little from the initial candidate segmentation computed from the *softmax* layer (denoted by f_i). The z_i in the unary term is weighted by the degree d_i of the pixel in order to produce larger unary costs for pixels that have many similar pixels within the neighborhood. Instead, the pairwise term ensures that pixels that are similar should be assigned similar z values. To balance the energies of the two terms we introduce a parameter μ and set it to 0.025 throughout all our experiments.

We can also express the same global energy function in matrix notation:

$$\mathbf{z}^* = \operatorname*{argmin}_{\mathbf{z}} \frac{\mu}{\mathbf{2}}\mathbf{D}(\mathbf{z} - \mathbf{D}^{-\mathbf{1}}\mathbf{f})^{\mathbf{T}}(\mathbf{z} - \mathbf{D}^{-\mathbf{1}}\mathbf{f}) + \frac{\mathbf{1}}{\mathbf{2}}\mathbf{z}^{\mathbf{T}}(\mathbf{D} - \mathbf{W})\mathbf{z} \tag{3.5}$$

where $\mathbf{z}^*$ is a $n \times 1$ vector containing an optimal continuous assignment for all n pixels, $\mathbf{D}$ is a diagonal degree matrix, and $\mathbf{W}$ is the $n \times n$ pixel affinity matrix. Finally, $\mathbf{f}$ denotes a $n \times 1$ vector containing the probabilities from the *softmax* layer corresponding to a particular object class.

An advantage of our energy is that it is differentiable. If we denote the above energy as $E(z)$ then the derivative of this energy can be written as follows:

$$\frac{\partial E(z)}{\partial z} = \mu\mathbf{D}(\mathbf{z} - \mathbf{D}^{-\mathbf{1}}\mathbf{f}) + (\mathbf{D} - \mathbf{W})\mathbf{z} = \mathbf{0} \tag{3.6}$$

With simple algebraic manipulations we can then obtain a closed form solution to this optimization:

$$\mathbf{z}^* = (\mathbf{D} - \alpha\mathbf{W})^{-\mathbf{1}}\beta\mathbf{f} \tag{3.7}$$

where $\alpha = \frac{1}{1+\mu}$ and $\beta = \frac{\mu}{1+\mu}$. In the general case where we have k object classes we can write the solution as:

$$\mathbf{Z}^* = (\mathbf{D} - \alpha\mathbf{W})^{-\mathbf{1}}\beta\mathbf{F} \tag{3.8}$$

where $\mathbf{Z}$ now depicts a $n \times k$ matrix containing assignments for all k object classes, while $\mathbf{F}$ denotes $n \times k$ matrix with object class probabilities from *softmax* layer. Due to the large size of $\mathbf{D} - \alpha\mathbf{W}$ it is impractical to invert it. However, if we consider an image as a graph where each pixel denotes a vertex in the graph, we can observe that the term $\mathbf{D} - \mathbf{W}$ in our optimization is equivalent to a Laplacian matrix of such graph. Since we know that a Laplacian matrix is positive semi-definite, we can use the preconditioned conjugate gradient method [79] to solve the system in Eq. (3.9). Alternatively, because our defined global energy in Eq. (3.5) is differentiable, we can efficiently solve this optimization problem using stochastic gradient descent. We choose the former option and solve the following system:

$$(\mathbf{D} - \alpha\mathbf{W})\mathbf{z}^* = \beta\mathbf{f} \tag{3.9}$$

To obtain the final discrete segmentation, for each pixel we assign the object class that corresponds to the largest column value in the row of $\mathbf{Z}$ (note that each row in $\mathbf{Z}$ represents a single pixel in the image, and each column in $\mathbf{Z}$ represents one of the object classes). In the experimental section, we show that this solution produces better quantitative and qualitative results in comparison to commonly used globalization techniques.

3.4. Experimental Results

In this section we present quantitative and qualitative results for semantic segmentation on the SBD [14] dataset, which contains objects and their per-pixel annotations for 20 Pascal VOC classes. We evaluate semantic segmentation results using two evaluation metrics. The first metric measures accuracy based on pixel intersection-over-union averaged per pixels (PP-IOU) across the 20 classes. According to this metric, the accuracy is computed on a per-pixel basis. As a result, the images that contain large object regions are given more importance. However, for certain applications we may need to accurately segment small objects. Therefore, similar to [2] we also consider the PI-IOU metric (pixel intersection-over-union averaged per image across the 20 classes), which gives equal weight to each of the images.

We compare Boundary Neural Fields with other commonly used global inference methods. These methods include Belief Propagation [76], Iterated Conditional Mode (ICM), Graph Cuts [77], and Dense-CRF [28]. Note that in all of our evaluations we use the same FCN unary potentials for every model.

Our evaluations provide evidence for three conclusions:

- In Subsection 3.4.1, we show that our boundary-based pixel affinities are better suited for semantic segmentation than the traditional color-based affinities.
- In Subsection 3.4.2, we demonstrate that our global minimization leads to better results than those achieved by other inference schemes.
- In Fig. 19, we qualitatively compare the outputs of FCN and Dense-CRF to our predicted segmentations. This comparison shows that the BNF segments are better localized around the object boundaries and that they are also spatially smooth.

3.4.1. Comparing Affinity Functions for Semantic Segmentation

In Table 12, we consider two global models. Both models use the same unary potentials obtained from the FCN *softmax* layer. However, the first model uses the popular color-based pairwise affinities, while the second employs our boundary-based affinities. Each of these two models is optimized using several inference strategies. The table shows that using our boundary based-affinity function improves the results of all global inference methods

Metric	Inference Method	RGB Affinity	**BNF Affinity**
PP-IOU	Belief Propagation [76]	75.4	**75.6**
	ICM	74.2	**75.8**
	TRWS [80]	75.9	**76.7**
	QPBO [81]	76.9	**77.2**
	BNF	74.6	**77.6**
PI-IOU	Belief Propagation [76]	45.9	**46.2**
	ICM	45.7	**48.8**
	TRWS [80]	51.5	**52.0**
	QPBO [81]	55.3	**57.2**
	BNF	53.0	**58.5**

Table 12: We compare semantic segmentation results when using a color-based pixel affinity and our proposed boundary-based affinity. We note that our proposed affinity improves the performance of all globalization techniques. Note that all of the inference methods use the **same FCN unary potentials**. This suggests that for every method our semantic boundary based affinity is more beneficial for semantic segmentation than the color-based affinities.

according to both evaluation metrics. Note that we cannot include Dense-CRF [28] in this comparison because it employs an efficient message-passing technique and integrating our affinities into this technique is a non-trivial task. However, we compare our method with Dense-CRF in Subsection 3.4.2.

The results in Table 12 suggest that our semantic boundary based pixel affinity function yields better semantic segmentation results compared to the commonly-used color based affinities. We note that we also compared the results of our inference technique using other edge detectors, notably UCM [30] and HfL [2]. In comparison to UCM edges, we observed that our boundaries provide 1.0% and 6.0% according to both evaluation metrics respectively. When comparing our boundaries with HfL method, we observed similar segmentation performance, which suggests that our method works best with the high quality semantic boundaries.

3.4.2. Comparing Inference Methods for Semantic Segmentation

Additionally, we also present semantic segmentation results for both of the metrics (PP-IOU and PI-IOU) in Table 13. In this comparison, all the techniques use the same FCN unary potentials. Additionally, all inference methods except Dense-CRF use our affinity measure (since the previous analysis suggested that our affinities yield better performance).

Metric	Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	mean
PP-IOU	FCN-Softmax	80.7	71.6	80.7	71.3	72.9	88.1	81.8	86.6	47.4	82.9	74.8
	Belief Propagation [76]	81.4	72.2	82.4	72.2	74.3	88.8	82.4	87.2	48.4	83.8	75.6
	ICM	81.7	72.2	82.8	72.1	75.3	89.6	83.4	87.7	46.3	83.3	75.8
	TRWS [80]	81.6	70.9	83.8	72.0	75.1	89.5	82.5	88.0	**51.7**	86.6	76.7
	Graph Cuts [77]	82.5	72.4	84.6	73.3	**77.2**	89.7	83.3	88.8	49.3	84.0	76.9
	QPBO [81]	82.6	72.3	84.7	73.1	76.7	89.9	**83.6**	89.3	49.7	85.0	77.2
	Dense-CRF [28]	**83.4**	71.5	84.9	72.6	76.2	89.5	83.3	89.1	50.4	**86.7**	77.3
	BNF-SB	81.9	72.5	84.9	73.3	76.0	90.3	83.1	89.2	51.2	**86.7**	**77.4**
	BNF-SB-SM	82.2	**73.1**	**85.1**	**73.8**	76.7	**90.6**	83.4	**89.5**	51.3	**86.7**	**77.6**
PI-IOU	FCN-Softmax	56.9	35.1	47.8	41.1	27.4	51.1	43.4	52.7	22.2	43.1	41.8
	Belief Propagation [76]	68.0	38.6	52.9	45.8	31.9	55.9	47.2	58.2	24.6	49.9	46.2
	ICM	65.3	40.9	56.4	45.3	33.7	58.9	49.5	61.9	25.8	53.5	48.8
	TRWS [80]	67.5	40.7	60.3	46.3	35.6	63.4	49.6	69.3	29.7	58.9	52.0
	Graph Cuts [77]	72.1	47.8	64.5	50.8	36.0	70.8	51.4	71.6	31.7	65.8	55.3
	QPBO [81]	71.6	46.8	65.6	49.6	38.0	72.6	52.7	76.7	32.5	69.6	57.2
	Dense-CRF [28]	68.0	39.5	58.0	45.0	33.4	62.8	47.7	66.0	29.4	60.9	51.1
	BNF-SB	71.6	48.1	**67.2**	52.3	37.8	**79.5**	52.9	**80.8**	**33.3**	71.5	58.5
	BNF-SB-SM	**72.0**	**48.9**	66.5	**52.9**	**39.1**	79.0	**53.4**	78.6	32.9	**72.2**	58.5

Metric	Method	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
PP-IOU	FCN-Softmax	57.9	83.9	79.6	80.4	81.0	64.7	78.2	54.5	80.9	69.9	74.8
	Belief Propagation [76]	58.4	84.6	80.5	80.9	81.5	65.1	79.5	55.5	81.5	71.2	75.6
	ICM	58.4	84.6	80.6	81.4	81.5	65.8	79.5	56.0	80.7	74.1	75.8
	TRWS [80]	61.9	85.8	83.3	80.8	81.1	65.3	81.5	**58.8**	77.6	75.9	76.7
	Graph Cuts [77]	60.3	85.4	82.2	81.2	81.9	66.7	79.8	58.0	82.3	74.9	76.9
	QPBO [81]	61.1	86.2	82.9	81.3	**82.3**	67.1	80.5	**58.8**	82.2	75.1	77.2
	Dense-CRF [28]	61.0	**86.8**	**83.5**	**81.8**	**82.3**	66.9	**82.2**	58.2	81.9	75.1	77.3
	BNF-SB	**61.5**	86.6	83.2	81.3	81.9	66.2	81.7	58.6	81.6	**75.8**	**77.4**
	BNF-SB-SM	61.4	**86.8**	83.3	81.7	**82.3**	**67.7**	81.9	58.4	**82.4**	75.4	**77.6**
PI-IOU	FCN-Softmax	29.2	54.2	40.5	45.6	59.1	24.2	43.6	24.8	55.9	37.2	41.8
	Belief Propagation [76]	31.7	60.2	44.9	50.1	62.4	25.2	49.9	27.6	62.3	42.2	46.2
	ICM	33.2	62.1	48.0	53.2	63.4	24.1	54.8	34.0	63.7	47.7	48.8
	TRWS [80]	37.8	67.4	57.3	53.8	64.1	26.3	62.0	36.9	63.1	49.9	52.0
	Graph Cuts [77]	34.4	71.8	62.0	59.4	64.8	29.0	60.9	38.7	70.3	51.6	55.3
	QPBO [81]	38.9	74.4	61.4	61.0	**66.2**	30.3	**68.7**	**41.4**	72.2	52.8	57.2
	Dense-CRF [28]	36.0	68.5	54.6	51.4	63.7	28.3	57.6	37.1	65.9	48.2	51.1
	BNF-SB	**39.5**	**75.1**	65.7	63.4	65.1	31.1	67.5	39.6	**73.2**	**54.7**	**58.5**
	BNF-SB-SM	39.4	74.6	**65.9**	**64.2**	65.8	**31.7**	66.9	39.0	73.1	53.9	**58.5**

Table 13: Semantic segmentation results on the SBD dataset according to PP-IOU (per pixel) and PI-IOU (per image) evaluation metrics. We use BNF-SB to denote the variant of our method that uses only semantic boundary based affinities. Additionally, we use BNF-SB-SM to indicate our method that uses boundary and *softmax* based affinities. We observe that our proposed globalization method outperforms other globalization techniques according to both metrics by at least 0.3% and 1.3% respectively. Note that in this experiment, all of the inference methods use **the same FCN unary potentials**. Additionally, for each method except Dense-CRF (it is challenging to incorporate boundary based affinities into the Dense-CRF framework) we use our boundary based affinities, since those lead to better results.

We use BNF-SB to denote the variant of our method that uses only semantic boundary based affinities. Additionally, we use BNF-SB-SM to indicate the version of our method that uses both boundary and *softmax*-based affinities (see Eq. (3.3)).

Based on these results, we observe that our proposed technique outperforms all the other globalization methods according to both metrics, by 0.3% and 1.3% respectively. Additionally, these results indicate that most benefit comes from the semantic boundary affinity term rather than the *softmax* affinity term.

In Fig. 19, we also present qualitative semantic segmentation results. Note that, compared to the segmentation output from the *softmax* layer, our segmentation is much better localized around the object boundaries. Additionally, in comparison to Dense-CRF predictions, our method produces segmentations that are much spatially smoother.

3.4.3. Semantic Boundary Classification

We can also label our boundaries with a specific object class, using the same classification strategy as in the HfL system [2]. Since the SBD dataset provides annotations for semantic boundary classification, we can test our results against the state-of-the-art HfL [2] method for this task. Due to the space limitation, we do not include full results for each category. However, we observe that our produced results achieve mean Max F-Score of 54.5% (averaged across all 20 classes) whereas HfL method obtains 51.7%.

3.5. Conclusions

In this work we introduced a Boundary Neural Field (BNF), an architecture that employs a semantic segmentation FCN to predict semantic boundaries and then uses the predicted boundaries and the FCN output to produce an improved semantic segmentation maps a global optimization. We showed that our predicted boundaries are better suited for semantic segmentation than the commonly used low-level color based affinities. Additionally, we introduced a global energy function that decomposes semantic segmentation into multiple binary problems and relaxes an integrality constraint. We demonstrated that the minimization of this global energy allows us to predict segmentations that are better localized around the object boundaries and that are spatially smoother compared to the segmentations achieved by prior methods. We made the code of our globalization technique available at `http://www.seas.upenn.edu/~gberta/publications.html`.

The main goal of this work was to show the effectiveness of boundary-based affinities for semantic segmentation. However, due to differentiability of our global energy, it may be possible to add more parameters inside the BNFs and learn them in a front-to-end fashion. We believe that optimizing the entire architecture jointly could capture the inherent relationship between semantic segmentation and boundary detection even better and further improve the performance of BNFs. We will investigate this possibility in our future work.

CHAPTER 4 : Convolutional Random Walk Networks for Semantic Image Segmentation

4.1. Introduction

Fully convolutional networks (FCNs) were first introduced in [27] where they were shown to yield significant improvements in semantic image segmentation. Adopting the FCN approach, many subsequent methods have achieved even better performance [29, 72, 73, 74, 75, 29, 72, 73, 82, 83]. However, traditional FCN-based methods tend to suffer from several limitations. Large receptive fields in the convolutional layers and the presence of pooling layers lead to low spatial resolution in the deepest FCN layers. As a result, their predicted segments tend to be blobby and lack fine object boundary details. We report in Fig. 27 some examples illustrating typical poor localization of objects in the outputs of FCNs. Recently, Chen at al. [29] addressed this issue by applying a Dense-CRF post-processing step [28] on top of coarse FCN segmentations. However, such approaches often fail to accurately capture semantic relationships between objects and lead to spatially fragmented segmentations (see an example in the last column of Fig. 27).

To address these problems several recent methods integrated CRFs or MRFs directly into the FCN framework [72, 82, 75, 84]. However, these new models typically involve (1) a large number of parameters, (2) complex loss functions requiring specialized model training

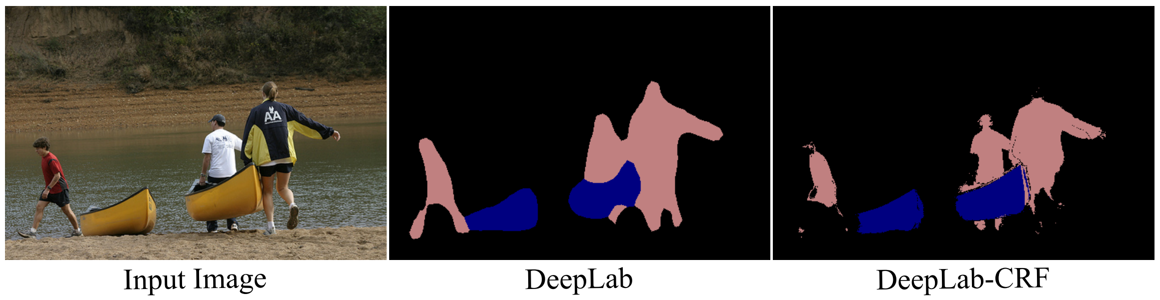


Figure 20: Examples illustrating shortcomings of prior semantic segmentation methods. Segments produced by FCNs are poorly localized around object boundaries, while Dense-CRF produce spatially-disjoint object segments.

	[29]	[84]	[83]	[72]	[82]	**RWN**
requires post-processing?	✓	✗	✗	✗	✗	✗
uses complex loss?	✗	✗	✗	✓	✓	✗
requires recurrent layers?	✗	✓	✗	✓	✗	✗
model size (in MB)	**79**	**79**	961	514	1000	**79**

Table 14: Summary of recent semantic segmentation models. For each model, we report whether it requires: (1) CRF post-processing, (2) complex loss functions, or (3) recurrent layers. We also list the size of the model (using the size of Caffe [16] models in MB). We note that unlike prior methods, our RWN does not need post-processing, it is implemented using standard layers and loss functions, and it also has a compact model.

or (3) recurrent layers, which make training and testing more complicated. We summarize the most prominent of these approaches and their model complexities in Table 14.

We note that we do not claim that using complex loss functions always makes the model overly complex and too difficult to use. If a complex loss is integrated into an FCN framework such that the FCN can still be trained in a standard fashion, and produce better results than using standard losses, then such a model is beneficial. However, in the context of prior segmentation methods [72, 82, 75, 84], such complex losses often require: 1) modifying the network structure (casting CNN into an RNN) [72, 84], or 2) using a complicated multi-stage learning scheme, where different layers are optimized during a different training stage [82, 75]. Due to such complex training procedures, which are adapted for specific tasks and datasets, these models can be quite difficult to adapt for new tasks and datasets, which is disadvantageous.

Inspired by random walk methods [85, 86, 87], in this work we introduce a simple, yet effective alternative to traditional FCNs: a Convolutional Random Walk Network (RWN) that combines the strengths of FCNs and random walk methods. Our model addresses the issues of (1) poor localization around the boundaries suffered by FCNs and (2) spatially disjoint segments produced by dense CRFs. Additionally, unlike recent semantic segmentation approaches [83, 72, 82, 75], our RWN does so without significantly increasing the complexity of the model.

Our proposed RWN jointly optimizes (1) pixelwise affinity and (2) semantic segmentation

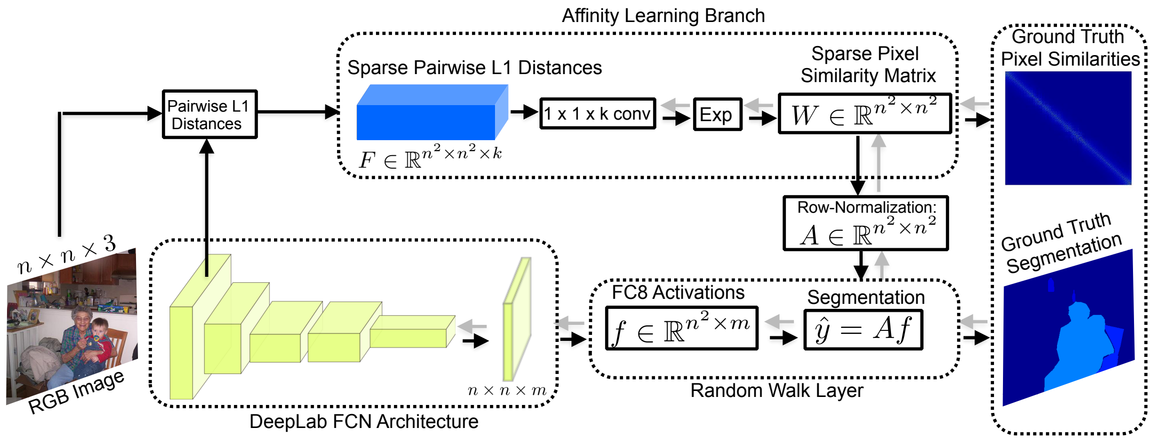

Affinity Learning Branch
Sparse Pairwise L1 Distances
$F \in \mathbb{R}^{n^2 \times n^2 \times k}$
1 x 1 x k conv
Exp
Sparse Pixel Similarity Matrix
$W \in \mathbb{R}^{n^2 \times n^2}$
Ground Truth Pixel Similarities
Pairwise L1 Distances
$n \times n \times 3$
RGB Image
Row-Normalization: $A \in \mathbb{R}^{n^2 \times n^2}$
Ground Truth Segmentation
FC8 Activations
$f \in \mathbb{R}^{n^2 \times m}$
Segmentation
$\hat{y} = Af$
Random Walk Layer
$n \times n \times m$
DeepLab FCN Architecture

Figure 21: The architecture of our proposed Random Walk Network (RWN) (best viewed in color). Our RWN consists of two branches: (1) one branch devoted to the segmentation predictions , and (2) another branch predicting pixel-level affinities. These two branches are then merged via a novel random walk layer that encourages spatially smooth segmentation predictions. The entire RWN is jointly optimized end-to-end via a standard back-propagation algorithm.

learning objectives that are linked via a novel random walk layer, which enforces spatial consistency in the deepest layers of the network. The random walk layer is implemented via matrix multiplication. As a result, RWN seamlessly integrates both affinity and segmentation branches, and can be jointly trained end-to-end via standard back-propagation with minimal modifications to the existing FCN framework. Additionally, our implementation of RWN requires only 131 additional parameters. Thus, the effective complexity of our model is the same as the complexity of traditional FCNs (see Table 14). We compare our approach to several variants of the DeepLab semantic segmentation system [29, 88], and show that our proposed RWN consistently produces better performance over these baselines for the tasks of semantic segmentation and scene labeling.

4.2. Related Work

The recent introduction of fully convolutional networks (FCNs) [27] has led to remarkable advances in semantic segmentation. However, due to the large receptive fields and many pooling layers, segments predicted by FCNs tend to be blobby and lack fine object boundary details. Recently there have been several attempts to address these problems. These approaches can be divided into several groups.

The work in [3, 29, 89, 90, 3] used FCN predictions as unary potentials in a separate globalization model that refines the segments using similarity cues based on regions or boundaries. One disadvantage of these methods is that the learning of the unary potentials and the training of the globalization model are completely disjoint. As a result, these methods often fail to capture semantic relationships between objects, which produces segmentation results that are spatially disjoint (see the right side of Fig. 27).

To address these issues, several recent methods [72, 82, 75] have proposed to integrate a CRF or a MRF into the network, thus enabling end-to-end training of the joint model. However, the merging of these two models leads to a dramatic increase in complexity and number of parameters. For instance, the method in [72], requires to cast the original FCN into a Recurrent Neural Network (RNN), which renders the model much bigger in size (see Table 14). A recent method [84] jointly predicts boundaries and segmentations, and then combines them using a recurrent layer, which also requires complex modifications to the existing FCN framework.

The work in [82] proposes to use **local** convolutional layers, which leads to a significantly larger number of parameters. Similarly, the method in [75] proposes to model unary and pairwise potentials by separate multi-scale branches. This leads to a network with at least twice as many parameters as the traditional FCN and a much more complex multi-stage training procedure.

In addition to the above methods, it is worth mentioning deconvolutional networks [83, 74], which use deconvolution and unpooling layers to recover fine object details from the coarse FCN predictions. However, in order to effectively recover fine details one must employ nearly as many deconvolutional layers as the number of convolutional layers, which yields a large growth in number of parameters (see Table 14).

Unlike these prior methods, our implementation of RWN needs only 131 additional parameters over the base FCN. These additional parameters represent only 0.0008% of the total number of parameters in the network. In addition, our RWN uses standard convolution and matrix multiplication. Thus, it does not need to incorporate complicated loss functions or new complex layers [72, 84, 82, 75]. Finally, unlike the methods in [3, 29, 89, 90] that predict and refine the segmentations disjointly, our RWN model jointly optimizes pixel affinity and semantic segmentation in an end-to-end fashion. Our experiments show that this leads to spatially smoother segmentations.

4.3. Background

Random Graph Walks. Random walks are one of the most widely known and used methods in graph theory [85]. Most notably, the concept of random walks led to the development of PageRank [87] and Personalized PageRank [86], which are widely used for many applications. Let $G = (V, E)$ denote an undirected graph with a set of vertices V and a set of edges E. Then a random walk in such graph can be characterized by the transition probabilities between its vertices. Let W be a symmetric $n \times n$ affinity matrix, where n denotes the number of nodes in the graph and where $W_{ij} \in [0, 1]$ denotes how similar the nodes i and j are. In the context of a semantic segmentation problem, each pixel in the image can be viewed as a separate node in the graph, where the similarity between two nodes can be evaluated according to some metric (e.g. color or texture similarity etc). Then let D indicate a diagonal $n \times n$ matrix, which stores the degree values for each node: $D_{ii} = \sum_{j=1}^{n} W_{ij}$ for all j except $i = j$. Then, we can express our random walk transition matrix as $A = D^{-1}W$.

Given this setup, we want to model how the information in the graph spreads if we start at a particular node, and perform a random walk in this graph. Let y_t be a $n \times 1$ vector denoting a node distribution at time t. In the context of the PageRank algorithm, y_t may indicate the rank estimates associated with each of the n Web pages at time t. Then, according to the random walk theory, we can spread the rank information in the graph by performing a one-step random walk. This process can be expressed as $y_{t+1} = Ay_t$, where y_{t+1} denotes a newly obtained rank distribution after one random walk step, the matrix A contains the random walk transition probabilities, and y_t is the rank distribution at time step t. Thus, we can observe that the information among the nodes can be diffused, by simply multiplying the random walk transition probability matrix A, with the rank distribution y_t at a particular time t. This process can be repeated multiple times, until convergence is reached. For a more detailed survey please see [85, 87].

Difference from MRF/CRF Approaches. CRFs and MRFs have been widely used in structured prediction problems [91]. Recently, CRFs and MRFs have also been integrated into the fully convolutional network framework for semantic segmentation [72, 82, 75]. We want to stress that while the goals of CRF/MRF and random walk methods are the same (i.e. to globally propagate information in the graph structures), the mechanism to achieve this objective is very different in these two approaches. While MRFs and CRFs typically employ graphs with a fixed grid structure (e.g., one where each node is connected to its four closest neighbors), random walk methods are much more flexible and can implement any arbitrary graph structure via the affinity matrix specification. Thus, since our proposed RWN is based on random walks, it can employ any arbitrary graph structure, which can be beneficial as different problems may require different graph structures.

Additionally, to globally propagate information among the nodes, MRFs and CRFs need to employ approximate inference techniques, because exact inference tends to be intractable in graphs with a grid structure. Integrating such approximate inference techniques into the steps of FCN training and prediction can be challenging and may require lots of domain-specific modifications. In comparison, random walk methods globally propagate information among the nodes via a simple matrix multiplication. Not only is the matrix multiplication efficient and exact, but it is also easy to integrate into the traditional FCN framework for both training and prediction schemes. Additionally, due to the use of standard convolution and matrix multiplication operations, our RWN can be trivially trained via standard back-propagation in an end-to-end fashion.

4.4. Convolutional Random Walk Networks

In this work, our goal is to integrate a random walk process into the FCN architecture to encourage coherent semantic segmentation among pixels that are similar to each other. Such a process introduces an explicit grouping mechanism, which should be beneficial in addressing the issues of (1) poor localization around the boundaries, and (2) spatially fragmented segmentations.

A schematic illustration of our proposed RWN architecture is presented in Fig. 21. Our RWN is a network composed of two branches: (1) one branch that predicts semantic segmentation potentials, and (2) another branch devoted to predicting pixel-level affinities. These two branches are merged via a novel random walk layer that encourages spatially coherent semantic segmentation. The entire RWN can be jointly optimized end-to-end. We now describe each of the components of the RWN architecture in more detail.

4.4.1. Semantic Segmentation Branch

For the semantic segmentation branch, we present results for several variants of the DeepLab segmentation systems, including DeepLab-LargeFOV [29], DeepLab-attention [88], and DeepLab-v2, which is one of the top performing segmentation systems. DeepLab-largeFOV is a fully convolutional adaptation of the VGG [60] architecture, which contains 16 convolutional layers. DeepLab-attention [88], is a multi-scale VGG based network, for which each multi-scale branch focuses on a specific part of the image. Finally, DeepLab-v2 is a multi-scale network based on the residual network [92] implementation. We note that even though we use a DeepLab architecture in our experiments, other architectures such as [**?**] and many others could be integrated into our framework.

4.4.2. Pixel-Level Affinity Branch

To learn the pairwise pixel-level affinities, we employ a separate affinity learning branch with its own learning objective (See Fig. 21). The affinity branch is connected with the input $n \times n \times 3$ RGB image, and low-level $conv1\_1$ and $conv1\_2$ layers. The feature maps corresponding to these layers are $n \times n$ in width and height but they have a different number of channels $(3, 64,$ and 64 respectively). Let k be the total number of affinity learning parameters (in our case $k = 3 + 64 + 64 = 131$). Then, let F be a **sparse** $n^2 \times n^2 \times k$ matrix that stores $L1$ distances between each pixel and all of its neighbors within a radius R, according to each channel. Note that the distances are **not** summed up across the k channels, but instead they are computed and stored separately for each channel. The resulting matrix F is then used as an input to the affinity branch, as shown in Figure 21.

The affinity branch consists of a $1 \times 1 \times k$ convolutional layer and an exponential layer. The output of the exponential layer is then attached to the Euclidean loss layer and is optimized to predict the ground truth pixel affinities, which are obtained from the original semantic segmentation annotations. Specifically, we set the ground truth affinity between two pixels to 1 if the pixels share the same semantic label and have distance less than R from each other. Note that F, which is used as an input to the affinity branch, is a **sparse** matrix, as only a small fraction of all the entries in F are populated with non-zero values. The rest of the entries are ignored during the computation.

Also note that we only use features from RGB, $conv1\_1$ and $conv1\_2$ layers, because they are not affected by pooling, and thus, preserve the original spatial resolution. We also experimented with using features from deeper FCN layers such as *fc6*, and *fc7*. However, we observed that features from deeper layers are highly correlated to the predicted semantic segmentation unary potentials, which causes redundancy and little improvement in the segmentation performance. We also experimented with using more than one convolutional layer in the affinity learning branch, but observed that additional layers provide negligible improvements in accuracy.

4.4.3. Random Walk Layer

To integrate the semantic segmentation potentials and our learned pixel-level affinities, we introduce a novel random walk layer, which propagates the semantic segmentation information based on the learned affinities. The random walk layer is connected to the two bottom layers: (1) the *fc8* layer containing the semantic segmentation potentials, and (2) the affinity layer that outputs a sparse $n^2 \times n^2$ random walk transition matrix A. Then, let f denote the activation values from the *fc8* layer, reshaped to the dimensions of $n^2 \times m$, where n^2 refers to the number of pixels, and m is the number of object classes in the dataset. A single random walk layer implements one step of the random walk process, which can be performed as $\hat{y} = Af$, where $\hat{y}$ indicates the diffused segmentation predictions, and A denotes the random walk transition matrix.

The random walk layer is then attached to the softmax loss layer, and is optimized to predict ground truth semantic segmentations. One of the advantages of our proposed random walk layer is that it is implemented as a matrix multiplication, which makes it possible to back-propagate the gradients to both (1) the affinity branch and (2) the segmentation branch. Let the softmax-loss gradient be an $n^2 \times m$ matrix $\frac{\partial L}{\partial \hat{y}}$, where n^2 is the number of pixels in the *fc8* layer, and m is the number of predicted object classes. Then the gradients, which are back-propagated to the semantic segmentation branch are computed as $\frac{\partial L}{\partial f} = A^T \frac{\partial L}{\partial \hat{y}}$, where A^T is the transposed random walk transition matrix. Also, the gradients, that are back-propagated to the affinity branch are computed as $\frac{\partial L}{\partial A} = \frac{\partial L}{\partial \hat{y}} f^T$, where f^T is a $m \times n^2$ matrix that contains transposed activation values from the *fc8* layer of the segmentation branch. We note that $\frac{\partial L}{\partial A}$ is a **sparse** $n^2 \times n^2$ matrix, which means that the above matrix multiplication only considers the pixel pairs that correspond to the non-zero entries in the random walk transition matrix A.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	mean	overall
DeepLab-largeFOV	79.8	71.5	78.9	70.9	72.1	87.9	81.2	85.7	46.9	80.9	73.8	76.0
RWN-largeFOV	**81.6**	**72.1**	**82.3**	**72.0**	**75.4**	**89.1**	**82.5**	**87.4**	**49.1**	**83.6**	**75.8**	**77.9**
DeepLab-attention	83.4	76.0	83.0	**74.2**	77.6	91.6	85.2	89.1	54.4	86.1	79.0	80.5
RWN-attention	**84.7**	**76.6**	**85.5**	74.0	**79.0**	**92.4**	**85.6**	**90.0**	**55.6**	**87.4**	**79.9**	**81.5**
DeepLab-v2	85.5	**50.6**	86.9	**74.4**	82.7	93.1	88.4	91.9	62.1	89.7	81.4	83.4
RWN-v2	**86.0**	50.0	**88.4**	73.5	**83.9**	**93.4**	**88.6**	**92.5**	**63.9**	**90.9**	**82.1**	**84.3**

Method	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean	overall
DeepLab-largeFOV	56.5	82.6	77.9	79.3	80.1	64.4	77.6	52.7	80.3	70.0	73.8	76.0
RWN-largeFOV	**57.9**	**84.8**	**80.7**	**80.2**	**81.2**	**65.7**	**79.7**	**55.5**	**81.5**	**74.0**	**75.8**	**77.9**
DeepLab-attention	62.9	86.7	83.8	84.2	82.4	**70.2**	84.7	61.0	84.8	77.9	79.0	80.5
RWN-attention	**63.5**	**88.2**	**85.0**	**84.8**	**83.4**	70.1	**85.9**	**62.6**	**85.1**	**79.3**	**79.9**	**81.5**
DeepLab-v2	71.5	90.3	86.2	86.3	84.6	**75.1**	87.6	72.2	87.8	81.3	81.4	83.4
RWN-v2	**72.6**	**90.9**	**87.3**	**86.9**	**85.7**	75.0	**89.0**	**74.0**	**88.1**	**82.3**	**82.1**	**84.3**

Table 15: Semantic segmentation results on the SBD dataset according to the per-pixel Intersection over Union evaluation metric. From the results, we observe that our proposed RWN consistently outperforms DeepLab-LargeFOV, DeepLab-attention, and DeepLab-v2 baselines.

4.4.4. Random Walk Prediction at Testing

In the previous subsection, we mentioned that the prediction in the random walk layer can be done via a simple matrix multiplication operation $\hat{y} = Af$, where A denotes the random walk transition matrix, and f depicts the activation values from the *fc8* layer. Typically, we want to apply multiple random walk steps until convergence is reached. However, we also do not want to deviate too much from our initial segmentation predictions, in case the random walk transition matrix is not perfectly accurate, which is a reasonable expectation. Thus, our prediction scheme needs to balance two effects: (1) propagating the segmentation information across the nodes using a random walk transition matrix, and (2) not deviating too much from the initial segmentation.

This tradeoff between the two quantities is very similar to the idea behind MRF and CRF models, which try to minimize an energy formed by unary and pairwise terms. However, as discussed earlier, MRF and CRF methods tend to use 1) grid structure graphs and 2) various approximate inference techniques to propagate segmentation information globally. In comparison, our random walk approach is advantageous because it can use 1) any arbitrary graph structure and 2) an exact matrix multiplication operation to achieve the same goal.

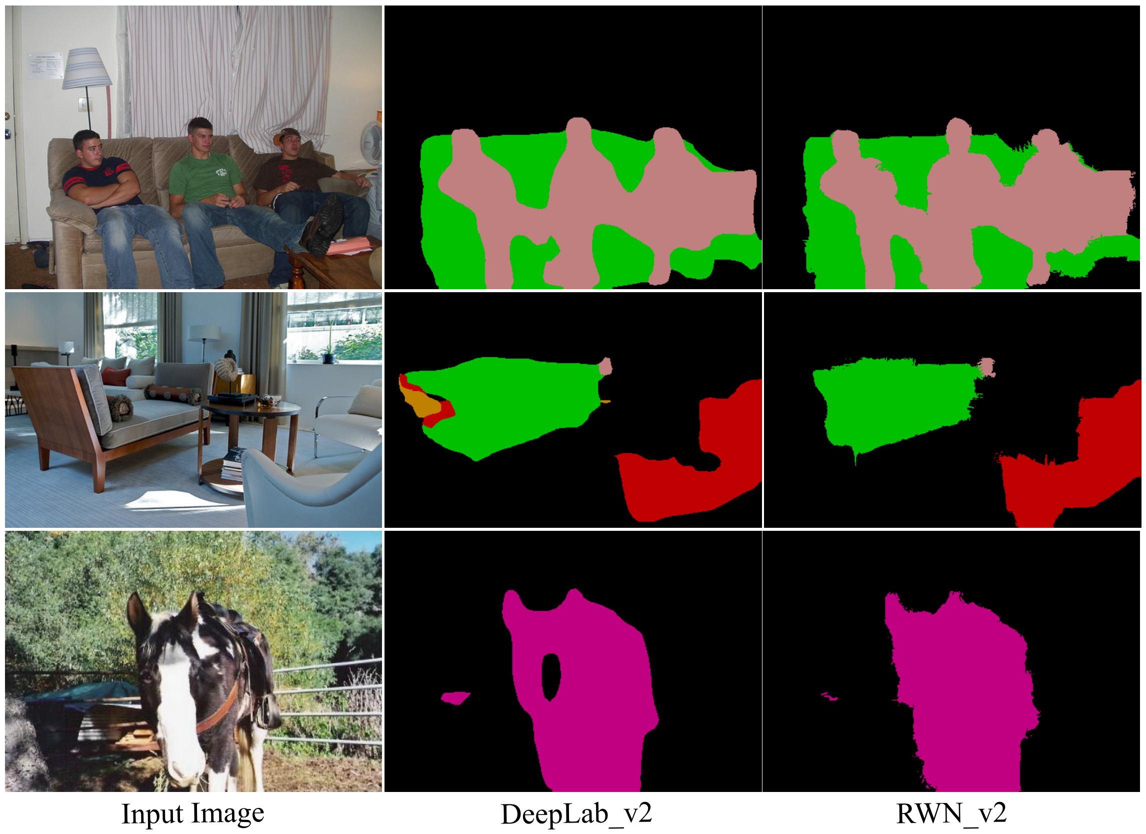


Figure 22: A figure illustrating the segmentation results of our RWN and the DeepLab-v2 network. Note that RWN produced segmentations are spatially smoother and produce less false positive predictions than the DeepLab-v2 system.

Let us first denote our segmentation prediction after $t+1$ random walk steps as $\hat{y}^{t+1}$. Then our general prediction scheme can be written as:

$$\hat{y}^{t+1} = \alpha A \hat{y}^{t} + (1-\alpha) f \qquad (4.1)$$

where α denotes a parameter $[0,1]$ that controls the tradeoff between (1) diffusing segmentation information along the connections of a random walk transition matrix and (2) not deviating too much from initial segmentation values (i.e. the outputs of the last layer of the FCN). Let us now initialize $\hat{y}^0$ to contain the output values from the *fc8* layer, which we denoted with f. Then we can write our prediction equation by substituting the recurrent expressions:

$$\hat{y}^{t+1} = (\alpha A)^{t+1} f + (1-\alpha) \sum_{i=0}^{t} (\alpha A)^i f \tag{4.2}$$

Now, because we want to apply our random walk procedure until convergence we set $t = \infty$. Then, because our random walk transition matrix is stochastic we know that $\lim_{t\to\infty} (\alpha A)^{t+1} = 0$. Furthermore, we can write $S_t = \sum_{i=0}^{t} A^i = I + A + A^2 + \ldots + A^t$, where I is an identity matrix, and where S_t denotes a partial sum of random walk transitions until iteration t. We can then write $S_t - AS_t = I - A^{t+1}$, which implies:

$$(I - A)S_t = I - A^{t+1} \tag{4.3}$$

From our previous derivation, we already know that $\lim_{t\to\infty} (A)^{t+1} = 0$, which implies that

$$S_\infty = (I - A)^{-1} \tag{4.4}$$

Thus, our final prediction equation, which corresponds to applying repeated random walk steps until convergence, can be written as

$$\hat{y}^\infty = (I - \alpha A)^{-1} f \tag{4.5}$$

In practice, the random walk transition matrix A is pretty large, and inverting it is impractical. To deal with this problem, we shrink the matrix $(I - \alpha A)$ using a simple and efficient technique presented in [38], and then invert it to compute the final segmentation. In the experimental section, we show that such a prediction scheme produces solid results and is still pretty efficient (≈ 1 second per image). We also note that we use this prediction scheme only during testing. During training we employ a scheme that uses a single random walk step (but with a much larger radius), which is faster. We explain this procedure in

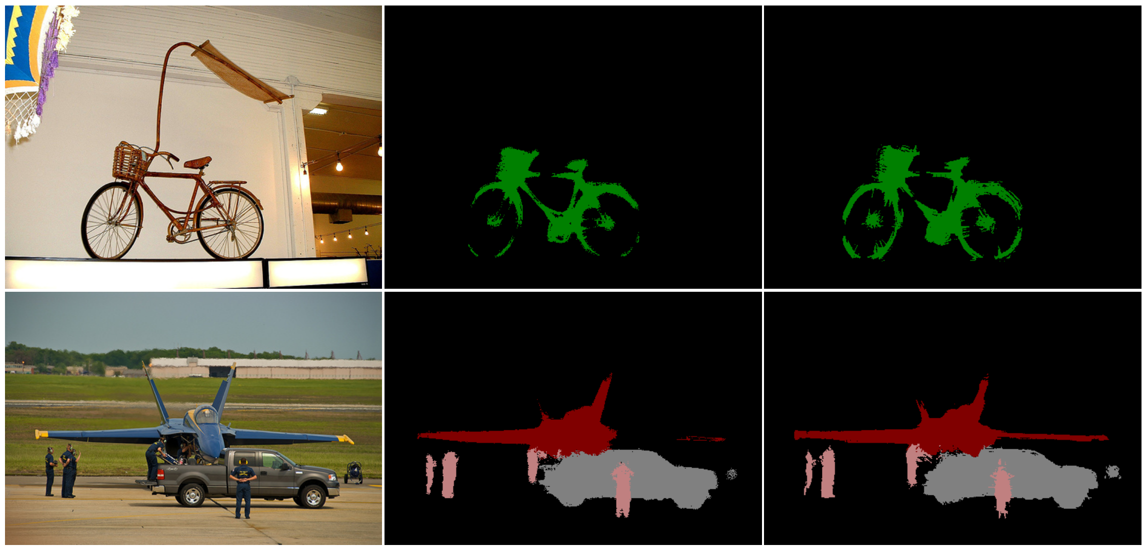


Figure 23: Comparison of segmentation results produced by our RWN versus the DeepLab-v2-CRF system. It can be noticed that, despite not using any post-processing steps, our RWN predicts fine object details (e.g., bike wheels or plane wings) more accurately than DeepLab-v2-CRF, which fails to capture some these object parts.

the next subsection.

4.4.5. Implementation Details

We jointly train our RWN in an end-to-end fashion for 2000 iterations, with a learning rate of 10^{-5}, 0.9 momentum, the weight decay of $5 \cdot 10^{-5}$, and 15 samples per batch. For the RWN model, we set the tradeoff parameter α to 0.01. During testing we set the random walk connectivity radius $R = 5$ and apply the random walk procedure until convergence. However, during training we set $R = 40$, and apply a single random walk step. This training strategy works well because increasing the radius size eliminates the need for multiple random walk steps, which speeds up the training. However, using $R = 5$ and applying an infinite number of random walk steps until convergence still yields slightly better results (see study in 2.5.3), so we use it during testing. For all of our experiments, we use a Caffe library [16]. During training, we also employ data augmentation techniques such as cropping, and mirroring.

Method	mean IOU	overall IOU
DeepLab-largeFOV-CRF	75.7	77.7
RWN-largeFOV	**75.8**	**77.9**
DeepLab-attention-CRF	**79.9**	**81.6**
RWN-attention	**79.9**	81.5
DeepLab-v2-CRF	81.9	84.2
RWN-v2	**82.1**	**84.3**
DeepLab-DT	76.6	78.7
RWN	**76.7**	**78.8**

Table 16: Quantitative comparison between our RWN model and several variants of the DeepLab system that use a dense CRF or a domain-transfer (DT) filter for post-processing. These results suggest that our RWN acts as an effective globalization scheme, since it produces results that are similar or even better than the results achieved by post-processing the DeepLab outputs with a CRF or DT.

4.5. Experimental Results

In this section, we present our results for semantic segmentation on the SBD [14] dataset, which contains objects and their per-pixel labels for 20 Pascal VOC classes (excluding the background class). We also include scene labeling results on the commonly used Stanford background [93] and Sift Flow [94] datasets. We evaluate our segmentation results on these tasks using the standard metric of the intersection-over-union (IOU) averaged per pixels across all the classes from each dataset. We also include the class-agnostic overall pixel intersection-over-union score, which measures the per-pixel IOU across all classes.

We experiment with several variants of the DeepLab system [29, 88] as our main baselines throughout our experiments: DeepLab-LargeFOV [29], DeepLab-attention [88], and DeepLab-v2.

Our evaluations provide evidence for four conclusions:

- In subsections 4.5.1, 4.5.2, we demonstrate that our proposed RWN outperforms DeepLab baselines for both semantic segmentation, and scene labeling tasks.
- In subsection 4.5.1, we demonstrate that, compared to the dense CRF approaches, RWN predicts segmentations that are spatially smoother.

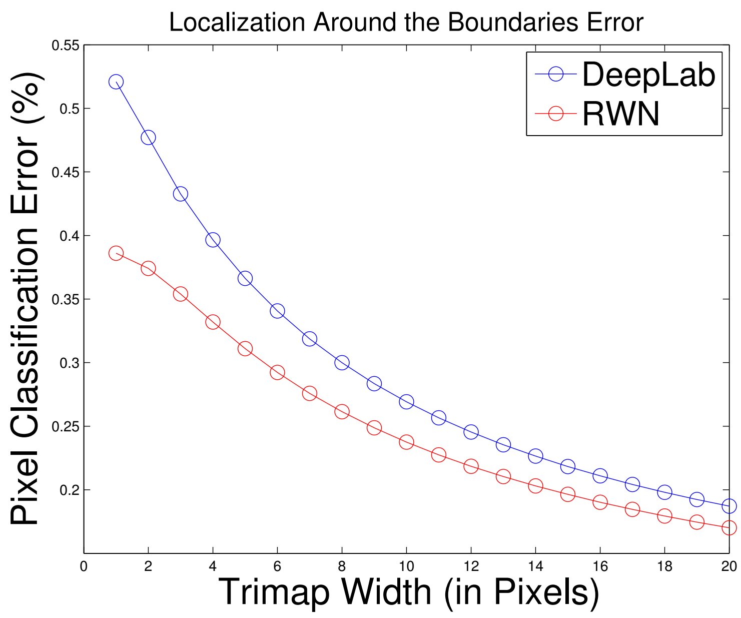


Figure 24: Localization error around the object boundaries within a trimap. Compared to the DeepLab system (blue), our RWN (red) achieves lower segmentation error around object boundaries for all trimap widths.

- In Subsection 4.5.3, we show that our approach is more efficient than the denseCRF inference.

- Finally, in Subsection, 2.5.3, we demonstrate that our random walk layer is beneficial and that our model is flexible to use different graph structures.

4.5.1. Semantic Segmentation Task

Standard Evaluation. In Table 15, we present semantic segmentation results on the Pascal SBD dataset [14], which contains 8055 training and 2857 testing images. These results indicate that RWN consistently outperforms all three of the DeepLab baselines. In Figure 22, we also compare qualitative segmentation results of a DeepLab-v2 network and our RWN model. We note that the RWN segmentations contain fewer false positive predictions and are also spatially smoother across the object regions.

Furthermore, in Table 16, we present experiments where we compare RWN with models using dense CRFs [28] to post-process the predictions of DeepLab systems. We also include

DeepLab-DT [84], which uses domain-transfer filtering to refine the segmentations inside an FCN. Based on these results, we observe that, despite not using any post-processing, our RWN produces results similar to or even better than the DeepLab models employing post-processing. These results indicate that RWN can be used as a globalization mechanism to ensure spatial coherence in semantic segmentation predictions. In Figure 23 we present qualitative results where we compare the final segmentation predictions of RWN and the DeepLab-v2-CRF system. Based on these qualitative results, we observe that RWN captures more accurately the fine details of the objects, such as the bike wheels, or plane wings. The DeepLab-v2-CRF system misses some of these object parts.

Localization Around the Boundaries. Earlier we claimed that due to the use of large receptive fields and many pooling layers, FCNs tend to produce blobby segmentations that lack fine object boundary details. We want to show that our RWN produces more accurate segmentations around object boundaries the traditional FCNs. Thus, adopting the practice from [28], we evaluate segmentation accuracy around object boundaries. We do so by counting the relative number of misclassified pixels within a narrow band ("trimap") surrounding the ground truth object boundaries. We present these results in Figure 24. The results show that RWN achieves higher segmentation accuracy than the DeepLab (DL) system for all trimap widths considered in this test.

Spatial Smoothness. We also argued that applying the dense CRF [28] as a post-processing technique often leads to spatially fragmented segmentations (see the right side of Fig. 27). How can we evaluate whether a given method produces spatially smooth or spatially fragmented segmentations? Intuitively, spatially fragmented segmentations will produce many false boundaries that do not correspond to actual object boundaries. Thus, to test the spatial smoothness of a given segmentation, we extract the boundaries from the segmentation and then compare these boundaries against ground truth object boundaries using the standard maximum F-score (MF) and average precision (AP) metrics, as done in the popular BSDS benchmark [50]. We perform this experiment on the Pascal SBD dataset

Method	MF	AP
DeepLab-largeFOV-CRF	0.676	0.457
RWN-largeFOV	**0.703**	**0.494**
DeepLab-attention-CRF	0.722	0.521
RWN-attention	**0.747**	**0.556**
DeepLab-v2-CRF	0.763	0.584
RWN-v2	**0.773**	**0.595**

Table 17: Quantitative comparison of spatial segmentation smoothness. We extract the boundaries from the predicted segmentations and evaluate them against ground truth object boundaries using max F-score (MF) and average precision (AP) metrics. These results suggest that RWN segmentations are spatially smoother than the DeepLab-CRF segmentations across all baselines.

and present these results in Table 17. We can see that the boundaries extracted from the RWN segmentations yield better MF and AP results compared to the boundaries extracted from the different variants of the DeepLab-CRF system. Thus, these results suggest that RWN produces spatially smoother segmentations than DeepLab-CRF.

4.5.2. Scene Labeling

We also tested our RWN on the task of scene labeling using two popular datasets: Stanford Background [93] and Sift Flow [94]. Stanford Background is a relatively small dataset for scene labeling. It contains 715 images, which we randomly split into 600 training images and 115 testing images. In contrast, the Sift Flow dataset contains 2489 training examples and 201 testing images. For all of our experiments, we use the DeepLab-largeFOV [29] architecture since it is smaller and more efficient to train and test. To evaluate scene labeling results, we use the overall IOU evaluation metric which is a commonly used metric for this task. In Table 18, we present our scene labeling results on both of these datasets. Our results indicate that our RWN method outperforms the DeepLab baseline by 2.57%, and 2.54% on these two datasets, respectively.

4.5.3. Runtime Comparisons

We also include the runtime comparison of our RWN approach versus the denseCRF inference. We note that using a single core of a 2.7GHz Intel Core i7 processor, the denseCRF

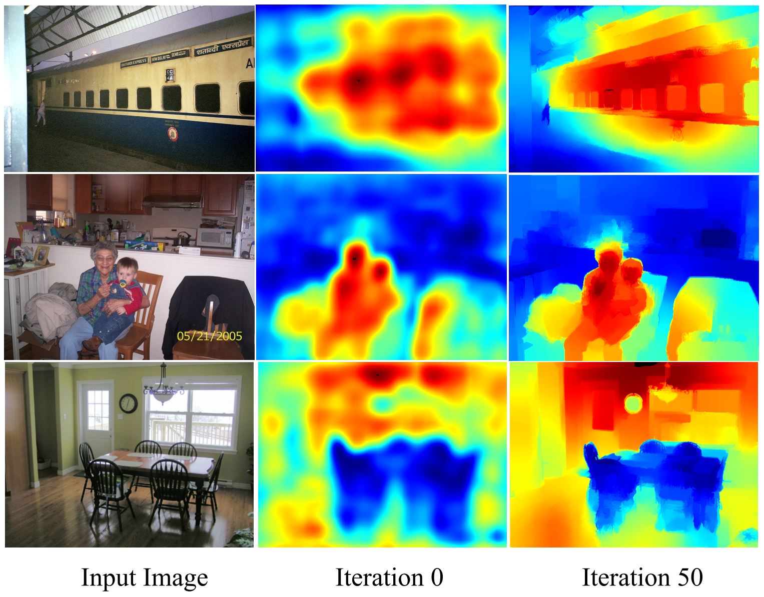


Figure 25: A figure illustrating how the probability predictions change as we apply more random walk steps. Note that the RWN predictions become more refined and better localized around the object boundaries as more random walk steps are applied.

inference requires 3.301 seconds per image on average on a Pascal SBD dataset. In comparison, a single iteration of a random walk, which is simply a sparse matrix multiplication, takes 0.032 seconds on average on the same Pascal SBD dataset. A DeepLab_v2 post-processed with denseCRF achieves 81.9% IOU score on this same Pascal SBD dataset. In comparison, RWN_v2 with a single random walk iteration and with R=40 (radius) achieves 82.2% IOU, which is both more accurate and more than 100 times more efficient than the denseCRF inference.

4.5.4. Ablation Experiments

Optimal Number of Random Walk Steps. In Figure 26, we illustrate how the IOU accuracy changes when we use a different number of random walk steps. We observe that the segmentation accuracy keeps increasing as we apply more random walk steps, and that it reaches its peak performance when the random walk process converges, which indicates the effectiveness of our random walk step procedure. In Figure 25, we also illustrate how the predicted object segmentation probabilities change as we apply more random walk steps.

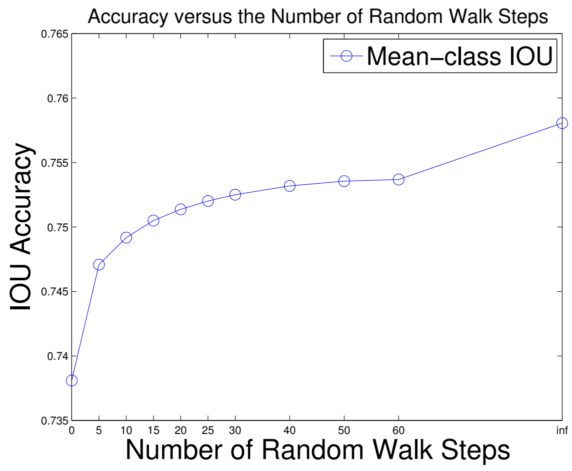


Figure 26: IOU accuracy as a function of the number of random walk steps. From this plot we observe that the segmentation accuracy keeps improving as we apply more random walk steps and that it reaches its peak when the random walk process converges.

We observe that the object boundaries become much better localized as more iterations of random walk are applied.

Radius Size. To analyze the effect of a radius size in the RWN architecture, we test alternative versions of our model with different radii sizes. Our results indicate, that the RWN model produces similar results with different radii in the interval of $R > 3$ and $R < 20$ if the random walk step process is applied until convergence. We also note that, if we select $R = 40$, and apply a random walk step only **once**, we can achieve the segmentation accuracy of 75.5% and 77.6% according to the two evaluation metrics, respectively. In comparison, choosing $R = 5$ and applying random walk until convergence yields the accuracies of 75.8% and 77.9%, which is slightly better. However, note that selecting $R = 40$, and applying multiple random walk steps does not yield any improvement in segmentation accuracy. These experiments show the flexibility of our model compared to the MRF or CRF models, which typically use graphs with a fixed grid structure. Our model has the ability to use different graph structures depending on the problem.

	DeepLab-largeFOV	RWN-largeFOV
Stanford-BG	65.74	**68.31**
Sift-Flow	67.31	**69.85**

Table 18: Scene labeling results on the Stanford Background and Sift-Flow datasets measured according to the overall IOU evaluation metric. We use a DeepLab-largeFOV network as base model, and show that our RWN yields better results on both of these scene labeling datasets.

4.6. Conclusion

In this work, we introduced Random Walk Networks (RWNs), and showed that, compared to traditional fully convolutional networks (FCNs), they produce improved accuracy for the same model complexity. Our RWN addresses the issues of 1) poor localization around the segmentation boundaries and 2) spatially disjoint segmentations. Additionally, our implementation of RWN uses only 131 additional learnable parameters (0.0008% of the original number of the parameters in the network) and it can be easily integrated into the standard FCN learning framework for a joint end-to-end training. Finally, RWN provides a more efficient alternative to the denseCRF approaches.

Our future work includes experimenting with alternative RWN architectures and applying RWN to new domains such as language processing or speech recognition.

CHAPTER 5 : Object Detection in Video with Spatiotemporal Sampling Networks

5.1. Introduction

In recent years, deep convolutional networks have achieved remarkable results in many computer vision tasks [26, 95, 60, 1, 96, 97, 36, 4], including object detection in images [98, 99, 100, 101, 102, 46, 103, 104, 105, 106, 107]. However, directly applying these image-level models to object detection in video is difficult due to motion blur, video defocus, unusual poses, or object occlusions (see Figure 27). Despite these challenges, it is natural to assume that video object detectors should be more powerful than still image detectors because video contains richer information about the same object instance (e.g., its appearance in different poses, and from different viewpoints). The key challenge then is designing a model that effectively exploits temporal information in videos.

Prior work [108, 109, 20, 110] has proposed to exploit such temporal information in videos by means of various post-processing steps aimed at making object detections coherent across time. However, since temporal coherence is enforced in a second stage, typically these methods cannot be trained end-to-end. To overcome this limitation, recent work [18] has introduced a flow-based aggregation network that is trainable end-to-end. It exploits optical flow to find correspondences across time and it then aggregates features across temporal correspondences to smooth object detections over adjacent frames. However, one of the downsides of this new model is that in addition to performing object detection, it also needs to predict motion. This is disadvantageous due to the following reasons: 1) designing an effective flow network architecture is not trivial, 2) training such a model requires large amounts of flow data, which may be difficult and costly to obtain, 3) integrating a flow network and a detection network into a single model may be challenging due to factors such as different loss functions, differing training procedures for each network, etc.

To address these shortcomings, in this work, we introduce a simple, yet effective Spatiotemporal Sampling Network (STSN) that uses deformable convolutions [111] across space and

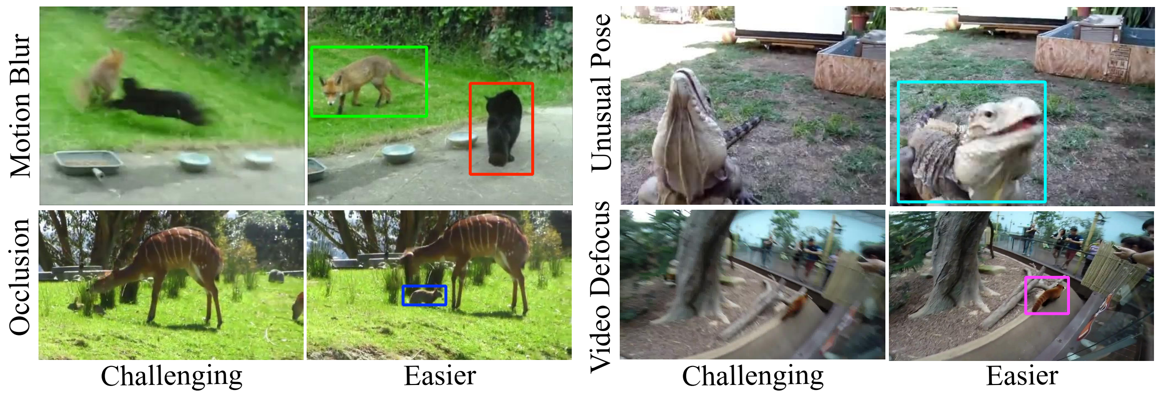


Figure 27: An illustration of the common challenges associated with object detection in video. These include video defocus, motion blur, occlusions and unusual poses. The bounding boxes denote the objects that we want to detect in these examples.

time to leverage temporal information for object detection in video. Our STSN learns to spatially sample useful feature points from nearby video frames such that object detection accuracy in a given video frame is maximized. To achieve this, we train our STSN end-to-end on a large set of video frames labeled with bounding boxes. We show that this leads to a better accuracy compared to the state-of-the-art on the ImageNet VID dataset [17], without requiring complex flow network design, or the need to train the network on large amounts of flow data.

5.2. Related Work

Object Detection in Images. Modern object detectors [98, 99, 100, 101, 102, 46, 103, 104, 105, 106, 107] are predominantly built on some form of deep CNNs [26, 60, 96]. One of the earliest deep CNN object detection systems was R-CNN [46], which involved a two-stage pipeline where object proposals were extracted in the first stage, and then each proposal was classified using a CNN. To reduce the computational burden, the methods in [98], and [102] leveraged ROI pooling, which led to more efficient learning. Furthermore, to unify the entire object detection pipeline, Faster R-CNN [101] replaced various region proposal methods by another network to make the entire system trainable end-to-end. Following this work, several methods [106, 107] extended Faster R-CNN into a system that runs in real time with small reduction in performance. Additionally, recent work [105] introduced position sensitive ROI pooling, which significantly improved the detection efficiency compared to prior object detection systems. Finally, two recent methods, Mask R-CNN [99], and Deformable CNNs [111], improved object detection results even further and they represent the current state-of-the-art in object detection. Whereas Mask-RCNNs use an additional branch that predicts a mask for each region of interest, Deformable CNNs employ deformable convolutions, which allow the network to condition discriminatively its receptive field on the input, and to also model deformations of objects more robustly.

While the aforementioned methods work well on images, they are not designed to exploit temporal relationships in video. Instead, our Spatiotemporal Sampling Network (STSN), is specifically designed for a video object detection task. Unlike standard Deformable CNNs [111], which use deformable convolution in the spatial domain, our STSN learns to sample features temporally across different video frames, which leads to improved video object detection accuracy.

Object Detection in Videos. Up until the introduction of the ImageNet VID challenge [17], there were no large-scale benchmarks for video object detection. Thus, there are only few methods that we can compare our work to. T-CNNs [108, 109] use a video object detection pipeline that involves predicting optical flow first, then propagating image-level predictions according to the flow, and finally using a tracking algorithm to select temporally consistent high confidence detections. Seq-NMS [20] constructs a temporal graph from overlapping bounding box detections across the adjacent frames, and then uses dynamic programming to select bounding box sequences with the highest overall detection score. The work of Lee et al. [110] treats a video object detection task as a multi-object tracking problem. Finally, the method of Feichtenhofer et al. [19] proposes a ConvNet architecture that solves detection and tracking problems jointly, and then applies a Viterbi algorithm based optimization to link the detections across time.

The approach most similar to our work is the method of Zhu et al. [18], who proposed an end-to-end trainable network that jointly estimates optical flow and also detects objects in video. This is accomplished by using the predicted optical flow to align the features from the adjacent frames. The aggregated features are then fed as input to the detection network.

Our method is beneficial over the methods that use optical flow CNNs such as the method of Zhu et al. [18]. First, we note that pretrained optical flow CNNs do not always generalize to new datasets, which may hinder video object detection performance. In contrast, our method has a learnable spatiotemporal sampling module that is discriminatively trained from object detection labels, and thus, it does not suffer from this issue. Furthermore, our STSN can be trained for video object detection in a single stage end-to-end. In comparison, methods that rely on optical flow require an additional stage to train an optical flow CNN, which renders the training procedure more cumbersome and lengthy. Finally, as we will show in the experimental section, our STSN also yields a gain —albeit moderate— in video object detection accuracy.

5.3. Background: Deformable Convolution

Before describing our method, we first review some background information on deformable convolution [111], which is one of the key components of our STSN. Let us first note that a standard 2D convolution is comprised of two steps: 1) sampling locations on a uniformly-spaced grid $\mathcal{R}$, and 2) performing a weighted summation of sampled values using weights w. For example, if we consider a standard 2D convolution with a 3×3 kernel, and a dilation factor of 1, the grid $\mathcal{R}$ is defined as $\mathcal{R} = \{(-1,-1),(-1,0),\ldots,(0,1),(1,1)\}$. Under a standard 2D convolution, to compute a new value at pixel location p_0 in the output feature map y, we would perform the following operation on the input feature map x:

$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n), \tag{5.1}$$

Instead, in a deformable 2D convolution, the grid $\mathcal{R}$ is augmented with data-conditioned offsets $\{\Delta p_n | n = 1, \ldots, N\}$, where $N = |\mathcal{R}|$. We can then compute a deformable convolution as:

$$y(p_0) = \sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \tag{5.2}$$

Since the offset Δp_n is typically fractional, the operation above is implemented using bilinear interpolation. Note that the offsets are obtained by applying a separate convolutional layer to the activation tensor containing the feature map x. This yields an offset map that has the same spatial resolution as the input feature map. Also, note that the offsets are shared across all feature channels of a given activation tensor. During training, the weights for the deformable convolution kernel, and the offsets kernel are learned jointly by propagating gradients through the bilinear interpolation operator. We refer the reader to the original work that introduced deformable convolutions [111] for further details.

5.4. Spatiotemporal Sampling Network

Our goal is to design a network architecture that incorporates temporal information for object detection in video.

Let us denote with I_t the frame at time t in the video. Let us consider one of the scenarios depicted in Figure 27, e.g., a setting where I_t is blurry, contains an object in an unusual pose, or perhaps an occlusion. But let us assume that a nearby frame I_{t+k} includes the same object clearly visible and in a relatively standard pose. If we only had access to I_t, accurate object detection would be very challenging. However, leveraging information from I_{t+k} may enable more robust detection in the frame I_t . Thus, the main challenge in this setting is incorporating object-level information from the *supporting* frame I_{t+k} for an improved object detection accuracy in the *reference* frame I_t. Note that in our system each frame in

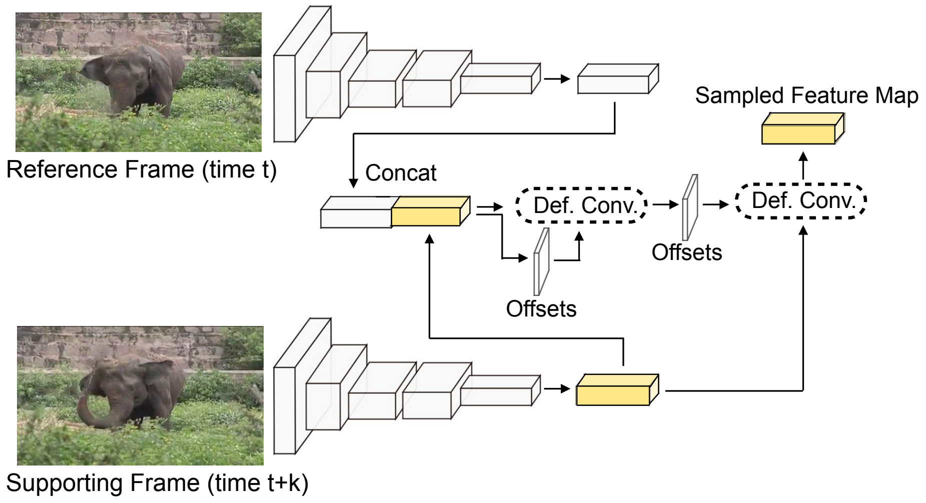


Figure 28: Our spatiotemporal sampling mechanism, which we use for video object detection. Given the task of detecting objects in a particular video frame (i.e., a reference frame), our goal is to incorporate information from a nearby frame of the same video (i.e., a supporting frame). First, we extract features from both frames via a backbone convolutional network (CNN). Next, we concatenate the features from the reference and supporting frames, and feed them through multiple deformable convolutional layers. The last of such layers produces offsets that are used to sample informative features from the supporting frame. Our spatiotemporal sampling scheme allows us to produce accurate detections even if objects in the reference frame appear blurry or occluded.

the video is treated in turn as a reference frame in order to produce object detection in every frame of the video. Furthermore, in practice we use $2K$ supporting frames for detection in the reference frame, by taking the K preceding frames and the K subsequent frames as supporting frames, i.e. $\{I_{t-K}, I_{t-(K-1)}, \dots, I_{t-1}, I_{t+1}, \dots, I_{t+(K-1)}, I_{t+K}\}$. However, for ease of explanation we introduce our STSN by considering a single supporting frame I_{t+k}.

To effectively integrate temporal information we need two things: 1) powerful object-level features from an image-level network, and 2) an ability to sample useful object-level features from the supporting frames for the reference frame. We achieve the former by employing a state-of-the-art backbone network. For the latter, we design a spatiotemporal sampling scheme, which we describe below.

Our STSN can be summarized in four steps. First, a backbone convolutional network computes object-level features for each video frame individually. Then, spatiotemporal sampling blocks are applied to the object-level feature maps in order to sample relevant features from nearby frames conditioned on the input *reference* frame. Next, the sampled features from each video frame are temporally aggregated into a single feature tensor for the reference frame using a per-pixel weighted summation. Finally, the feature tensor is provided as input to the detection network to produce final object detection results for the given reference frame. We note that our framework integrates these conceptually-distinct four steps into a single architecture, which we train end-to-end.

Backbone Architecture. Our backbone network is applied to each frame of the video. As backbone network, we use a Deformable CNN [111] based on the ResNet-101 [96] architecture, which is one of the top-performing object detection systems at the moment. Similarly to [111], our backbone network employs 6 deformable convolutional layers. We also note that even though we use a Deformable CNN architecture, our system can easily integrate other architectures and thus it can benefit from future improvements in still-image object detection.

Spatiotemporal Feature Sampling. Our main contribution is the design of a spatiotemporal sampling mechanism, which seamlessly integrates temporal information in a given video. As a first step, we feed the reference frame I_t and the supporting frame I_{t+k} through our image-level backbone network, which produces feature tensors f_t and f_{t+k}, respectively. Note that $f_t, f_{t+k} \in \mathbb{R}^{c \times h \times w}$ where $c, h,$ and w are the number of channels, the height, and the width of the activation tensor. The feature tensors f_t, and f_{t+k} are then concatenated into a new feature tensor $f_{t,t+k} \in \mathbb{R}^{2c \times h \times w}$. Note that this tensor $f_{t,t+k}$ now has twice as many channels as our initial tensors, and that it now contains object-level information from both the reference and the supporting frame.

Next, we use the tensor $f_{t,t+k}$ to predict (x, y) location offsets, which are then used to sample the supporting tensor f_{t+k}. The sampling mechanism is implemented using a deformable convolutional layer, which takes 1) the predicted offsets, and 2) the supporting tensor

f_{t+k} as its inputs, and then outputs a newly sampled feature tensor $g_{t,t+k}$, which can be used for object detection in the reference frame. We use subscript $t, t+k$ to denote the resampled tensor because, although g is obtained by resampling the *supporting* tensor, the offset computation uses both the reference as well as the supporting frame. A detailed illustration of our spatiotemporal sampling scheme is presented in Figure 28.

In practice, our spatiotemporal sampling block has 4 deformable convolution layers (only 2 are shown in Figure 28). This means that the initially predicted offsets $o^{(1)}_{t,t+k}$ and the concatenated temporal features $f_{t,t+k}$ are first used as inputs to a deformable convolution layer that outputs a new feature map $g^{(1)}_{t,t+k}$. Next, we use $g^{(1)}_{t,t+k}$ to predict offsets $o^{(2)}_{t,t+k}$, and a new feature map $g^{(2)}_{t,t+k}$. This continues for 2 more layers until we obtain offsets $o^{(4)}_{t,t+k}$, which are then used to sample the points out of the supporting feature map f_{t+k}. The final sampled feature map $g^{(4)}_{t,t+k}$ is obtained via another deformable convolutional layer that takes as inputs offsets $o^{(4)}_{t,t+k}$ and the original supporting feature map f_{t+k}.

Our proposed spatiotemporal sampling mechanism learns, which object-level features in the supporting frame are useful for object detection in the reference frame. Conceptually, it replaces the optical flow used in [18] to establish temporal correspondences with a learnable module that is discriminatively trained from object detection labels. In our experimental section, we show that such a sampling scheme allows us to improve video object detection performance over the still-image baseline and the flow-based method of Zhu et al. [18] without training our model on optical flow data.

Feature Aggregation. The spatiotemporal sampling procedure is applied for all the supporting frames in the selected range. Note, that this includes a special case, when the reference frame is treated as a supporting frame to itself to produce $g^{(4)}_{t,t}$, which is a feature tensor computed from only the reference frame.

The resulting feature tensors have the following form: $g^{(4)}_{t,t+k} \in \mathbb{R}^{c^{(4)} \times h \times w}$. These feature tensors are aggregated into an output feature tensor $g^{agg}_{t} \in \mathbb{R}^{c^{(4)} \times h \times w}$ for the reference frame. This tensor captures information from the reference frame, its K preceding frames and its K subsequent frames. The output tensor value $g_t^{agg}(p)$ for frame t at pixel p is computed as a weighted summation:

$$g_t^{agg}(p) = \sum_{k=-K}^{K} w_{t,t+k}(p)\; g_{t,t+k}^{(4)}(p) \tag{5.3}$$

Inspired by strong results presented in [18], we use their proposed feature aggregation method where the weights w indicate the importance of each supporting frame to the reference frame. To compute the weights w, we attach a 3-layer subnetwork $S(x)$ to the features $g_{t,t+k}^{(4)}$ and then compute their intermediate feature representations $S(g_{t,t+k}^{(4)})$. We then obtain the weights w by applying an exponential function on the cosine similarity between each corresponding feature point in a reference frame and a supporting frame:

$$w_{t,t+k}(p) = \exp\left(\frac{S(g_{t,t}^{(4)})(p) \cdot S(g_{t,t+k}^{(4)})(p)}{|S(g_{t,t}^{(4)})(p)||S(g_{t,t+k}^{(4)})(p)|}\right) \tag{5.4}$$

Finally, all weights w are fed into the softmax layer, to ensure that the weights sum up to 1 at each pixel location p (i.e., $\sum_{k=-K}^{K} w_{t,t+k}(p) = 1 \; \forall p$).

Object Detection. Finally, the aggregated feature tensor g_t^{agg} is used as input to the detection network, which outputs the final bounding box predictions and their object class probabilities. We describe more details related to the detection network in the next section along with other implementation details.

5.4.1. Implementation Details

For our experiments we use the MXNet [112] library. Below we provide details related to our STSN architecture, and our training and inference procedures.

Architecture. For our backbone network we adopt a state-of-the-art Deformable CNN [111] based on the ResNet-101 [96] architecture. Our spatiotemporal sampling block consists of four 3×3 deformable convolutional layers each with 1024 output channels. In addition, it also has four 3×3 convolutional layers predicting (x, y) offsets. To implement a subnetwork $S(x)$ that predicts feature aggregation weights, we use a sequence of 1×1, 3×3 and 1×1 convolutional layers with $512, 512$ and 2048 output channels respectively. Our detection network is implemented based on the deformable R-FCN design [105, 113, 111]. When feeding the aggregated feature g_t^{agg} to the detection network, we split its 1024 channels into two parts, and feed the first and the last 512 channels to the RPN and R-FCN subnetworks respectively. For the RPN, we use 9 anchors and 300 proposals for each image. Furthermore, for the R-FCN, we use deformable position-sensitive ROI pooling with 7×7 groups.

Training. Our entire STSN model is fully differentiable, and thus, trainable end-to-end. During training, we resize all input images to a shorter side of 600 pixels, and use $T = 3$ frames to train our model (i.e., $K = 1$). More specifically, we randomly sample one supporting frame before and one supporting frame after the reference frame. We observed that using more supporting frames in training does not lead to a higher accuracy.

For the rest of our training procedure, we follow the protocol outlined in [18]. Specifically, we train our model in two stages. First, we pre-train our full model on the Imagenet DET dataset using the annotations of the 30 object classes that overlap with the Imagenet VID dataset. Note that Imagenet DET dataset contains only images, and thus, we cannot sample meaningful supporting frames in this case. Therefore, in the case of images, we use the reference frames as our supporting frames. Afterwards, the entire model is trained for $120K$ iterations on 4 Tesla K40 GPUs with each GPU holding a single mini-batch. The learning rate is set to 0.001 and 0.0001 for the first $80K$ and the last $40K$ iterations respectively. Afterwards, we finetune the entire model on the Imagenet VID dataset for $60K$ iterations with a learning rate of 0.001 and 0.0001 for the first $40K$ and the last $20K$ iterations respectively. Note that in the second stage of training we sample the supporting frames randomly within a certain neighborhood of a reference frame (as described above).

Inference. During inference, we use $T = 27$, meaning that we consider $K = 13$ supporting frames before and after the reference frame. To avoid GPU memory issues, we first extract features from the backbone network for each image individually, and then cache these features in the memory. Afterwards, we feed all these features into our spatiotemporal sampling block. At the end, standard NMS with a threshold of 0.3 is applied to refine the detections. To handle the first and the last $K = 13$ frames in the video —two boundary cases that require sampling the neighboring frames beyond the video start and end, we pad the start of a video with K copies of the first frame, and the end of a video with K copies of the last frame.

5.5. Experimental Results

In this section, we evaluate our approach for video object detection on the ImageNet VID [17] dataset, which has $3,862$ and 555 training and testing video clips respectively. Each video is annotated with bounding boxes. The frames from each video are extracted at $25-30$ fps. The dataset contains 30 object categories that are a subset of the 200 categories in the ImageNet DET dataset.

Quantitative Results. To assess the effectiveness of our method we compare it to several relevant baselines, mainly two state-of-the-art methods FGFA [18] and D&T [19]. First, to verify that using temporal information from video is beneficial, we include a static image-level variant of our model (SSN) that uses only the reference frame to make its predictions. Furthermore, we also want to show that our proposed spatiotemporal sampling scheme

	Methods					
	D&T	SSN	FGFA	STSN	D&T+	STSN+
No FlowNet?	✓	-	✗	✓	✓	✓
Not Using Flow Data?	✓	-	✗	✓	✓	✓
No Temporal Post-Processing?	✓	-	✓	✓	✗	✗
mAP@0.5	75.8	76.0	78.8	78.9	79.8	**80.4**

Table 19: We use the ImageNet VID [17] dataset to compare our STSN to the state-of-the-art FGFA [18] and D&T [19] methods. Note that SSN refers to our static baseline, which is obtained by using only the reference frame for output generation (no temporal info). Also note, that D&T+ and STSN+ refer to D&T and STSN baselines with temporal post-processing applied on top of the CNN outputs. Based on these results, we first point out that unlike FGFA, our STSN does not rely on the external optical flow data, and still yields higher mAP (**78.9** vs **78.8**). Furthermore, when no temporal post-processing is used, our STSN produces superior performance in comparison to the D&T baseline (**78.9** vs **75.8**). Finally, we demonstrate that if we use a simple Seq-NMS [20] temporal post-processing scheme on top of our STSN predictions, we can further improve our results and outperform all the other baselines.

works as effectively as the optical flow network in [18], but without requiring optical flow supervision. To do so, we replace the optical flow network from [18], with our spatiotemporal sampling mechanism. The rest of the architecture and the training details are kept the same for both baselines. Such an experimental design allows us to directly compare the effectiveness of our spatiotemporal sampling scheme and the optical flow network of FGFA [18].

Finally, we demonstrate that our method performs better than the D&T [19] method in two scenarios: 1) when we only use CNN-level outputs for video object detection, and also 2) when we allow temporal post-processing techniques such as Seq-NMS to be applied on top of the CNN outputs. We note that in Table 19, D&T [19] and STSN refer to the CNN-level baselines whereas D&T+ [19] and STSN+ denote these same methods but with temporal post-processing (i.e. Seq-NMS [20], object-tube based linking [19], etc) applied on top of the CNN outputs.

We present our results in Table 19, where we assess each method according to several criteria. In the first row of Table 19, we list whether a given method requires integrating a separate flow network into its training / prediction pipeline. Ideally, we would want to eliminate

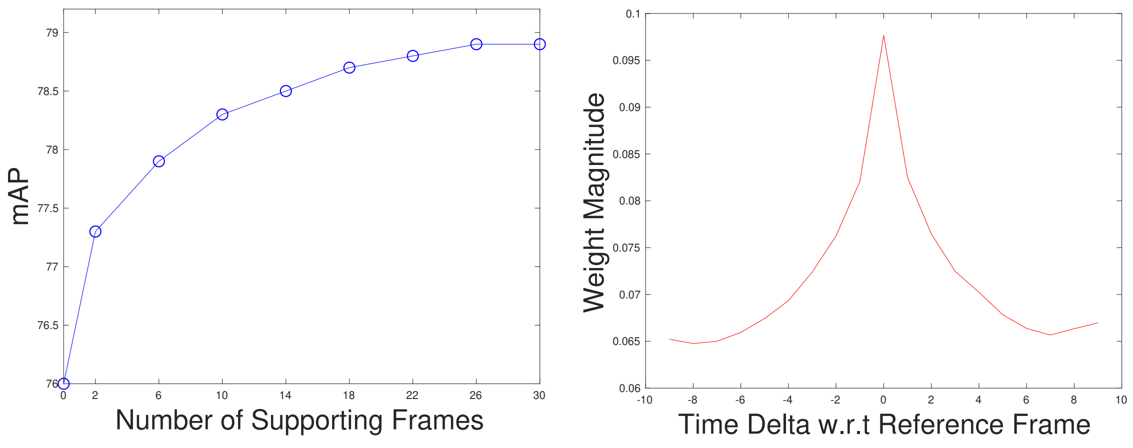


Figure 29: A figure illustrating some of our ablation experiments. **Left:** we plot mAP as a function of the number of supporting frames used by our STSN. From this plot, we notice that the video object detection accuracy improves as we use more supporting frames. **Right:** To understand the contribution of each of the supporting frames, we plot the average weight magnitudes $w_{t,t+k}(p)$ for different values of k. Here, p represents a point at the center of an object. From this plot, we observe that the largest weights are associated with the supporting frames that are near the reference frame. However, note that even supporting frames that are further away from the reference frame (e.g. $k = 9$) contribute quite substantially to the final object detection predictions.

this step because optical flow prediction requires designing a highly complex flow network architecture. We also list whether a given method requires pre-training on the external optical flow data, which we would want to avoid since it makes the whole training pipeline more costly. Additionally, we list, whether a given method uses any external temporal post-processing steps, which we would want to eliminate because they typically make the training / prediction pipeline disjoint and more complex. Finally, we assess each method according to the standard mean average precision (mAP) metric at intersection over union (IoU) threshold of 0.5.

Based on our results in Table 19, we make the following three conclusions. First, we note that our STSN produces better quantitative results than the state-of-the-art FGFA method (**78.9** vs **78.8**). We acknowledge that our accuracy improvement over FGFA is moderate. However, we point out that our STSN operates in a much more challenging setting than FGFA. Unlike FGFA, our STSN does not use any optical flow supervision.

Instead, it is trained directly for video object detection. The fact that STSN learns temporal correspondences without direct optical flow supervision, and still outperforms FGFA is quite impressive. Such results also show the benefit of discriminative end-to-end training with respect to the final video object detection task objective.

Second, our results indicate that in a setting when no temporal post-processing (i.e. Seq-NMS [20], object-tube based linking [19], etc) is used, our STSN outperforms the D&T baseline by a substantial margin (**78.9** vs **75.8**). These results indicate, that our STSN is able to learn powerful spatiotemporal features, and produce solid video object detection results even without using explicit temporal post-processing algorithms that link bounding box detections over time.

Lastly, we show that if we integrate a simple Seq-NMS [20] temporal post-processing scheme into our STSN, we can further improve our results and outperform the D&T+ baseline, which uses a similar Viterbi based temporal post-processing scheme [19] (**80.4** vs **79.8**). In general, we would like to note that our STSN aims to produce powerful spatiotemporal features, whereas the method of D&T [19] is targeted more for smoothing the final bounding box predictions across time. Thus, due to these characteristics we believe that these two methods are complementary, and it would be possible to integrate them together for even better results.

Optimal Number of Supporting Frames. In the left subplot of Figure 29, we also illustrate how the number of supporting frames affects the video object detection accuracy. We notice that the performance keeps increasing as we add more supporting frames, and then plateaus at $T = 27$.

Increasing the Temporal Stride. We also investigate how the temporal stride k, at which we sample the supporting frames, affects STSN's performance. We report that temporal strides of $k = 2$ and $k = 4$, yield mAP scores of 79.0 and 77.9, respectively. Thus, $k = 2$ yields a slight improvement over our original 78.9 mAP score. However, increasing k to

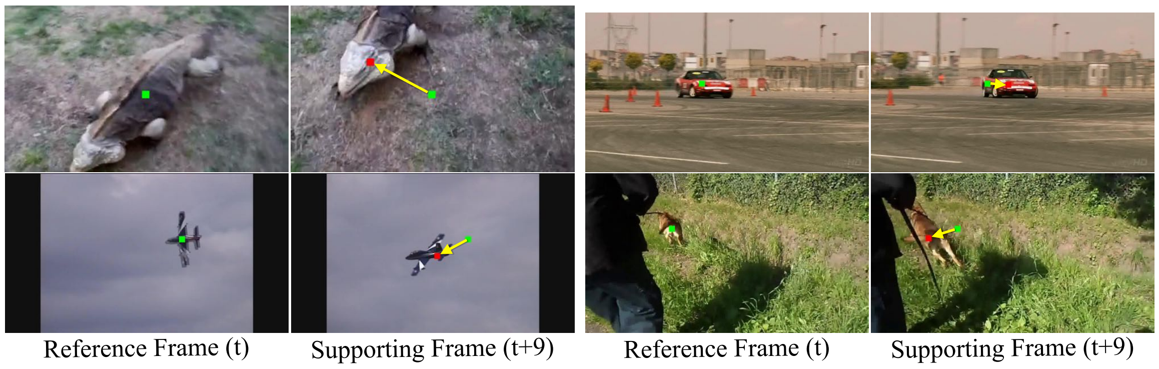


Figure 30: An illustration of our spatiotemporal sampling scheme (zoom-in for a better view). The green square indicates a point in the reference frame, for which we want to compute a new convolutional output. The red square indicates the corresponding point predicted by our STSN in a supporting frame. The yellow arrow illustrates the estimated object motion. Although our model is trained discriminatively for object detection and *not* for tracking or motion estimation, our STSN learns to sample from the supporting frame at locations that coincide almost perfectly with the same object. This allows our method to perform accurate object detection even if objects in the reference frame are blurry or occluded.

larger values reduces the accuracy.

Feature Aggregation Weight Analysis. To analyze how much each of the supporting frame contributes to the final object detections, we visualize the average weight magnitudes $w_{t,t+k}(p)$ for different values of k. This visualization is presented in the right subplot of Figure 29. We note that in this case, the weight magnitudes correspond to the point p, which is located at the center of an object. From this plot, we can conclude that the largest contribution comes from the supporting frames that are near the reference frame ($k = -1, 0, 1$). However, note that even even supporting frames that are further away from the reference frame (e.g. $k = -9, 9$), have non-zero weights, and contribute quite substantially to the final object detection predictions.

Qualitative Results. To get a better understanding how our STSN exploits temporal information from a given video, we visualize in Figure 30, the average offsets predicted by the STSN sampling block. These offsets are used by the STSN to decide, which object-level information from the supporting frame should be used to detect an object in the

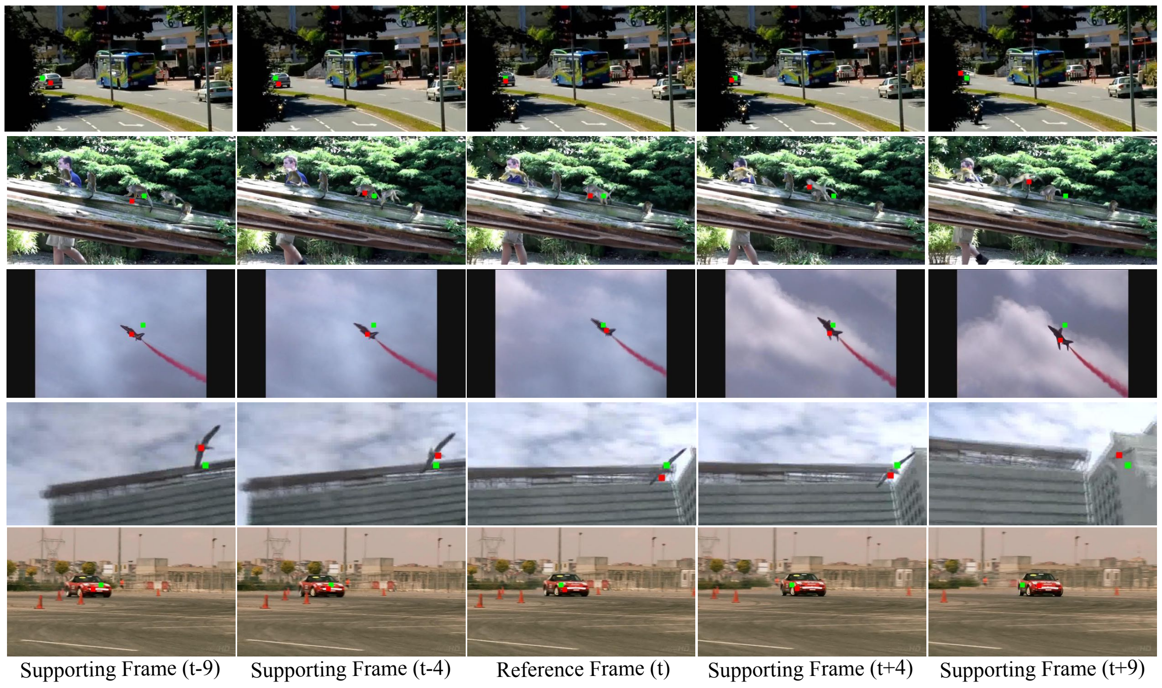


Figure 31: An illustration of using our spatiotemporal sampling scheme in action. The green square indicates a fixed object location in the reference frame. The red square depicts a location in a supporting frame, from which relevant features are sampled. Even without optical flow supervision, our STSN learns to track these objects in video. In our supplementary material, we include more of such examples in the video format.

reference frame. The green square in the reference frame depicts a pixel, for which we want to compute a convolution output. The red square in the supporting frame represents an average offset, which is used to determine which feature points from the supporting frame should be sampled. The yellow arrow indicates object's motion between the reference frame and the supporting frame. Note that despite a relatively large motion between the reference and the supporting frames, our STSN samples features from the supporting frame right around the center of the object, which is exactly what we want. Such spatiotemporal sampling allows us to detect objects even if they appear blurry or occluded in the reference frame.

In addition, based on the results in Figure 30, we observe that even without an explicit optical flow supervision, our STSN learns to accurately capture the motion of the objects, which is another appealing property of our model. In fact, in Figure 31, we illustrate several examples of using our STSN to track objects in a given video. From Figure 31, we observe that despite a relatively large motion in each sequence, our STSN accurately samples features around objects in every supporting frame. Such results indicate that we may be able to use our sampling mechanism for discriminative object tracking. In fact, we note that the commonly used dense optical flow methods are often redundant because most applications do not require flow prediction for every single pixel. In comparison, we point out that our STSN captures a more discriminative form of motion, which is learned to exclusively benefit a video object detection task. In our supplementary material, we include more of such results in the video form.

In addition to visualizing our spatiotemporal sampling results, in Figure 32, we also illustrate object detections of the static SSN baseline, and those of our full STSN model (zoom-in to see the probabilities and class predictions of each baseline). In all of these cases, we observe that incorporating temporal information helps STSN to correct the mistakes made by the static baseline. For instance, in the third row of Figure 32, a static SSN baseline incorrectly labels an object in the reference frame as a bird, which probably happens due to the occluded head of the lizard. However, STSN is able to fix this mistake by looking at the supporting frames, and by sampling around the lizard body and its head (See Row 3, Column 1 in Figure 32). Furthermore, in the last row, a static SSN baseline fails to detect one of the bicycles because it is occluded in the reference frame. Once again STSN fixes this error, by sampling around the missed bicycle in the supporting frame where the bicycle is more clearly visible. Similar behavior also occurs in other cases where STSN successfully resolves occlusion and blurriness issues.

5.6. Conclusion

In this work, we introduced the Spatiotemporal Sampling Network (STSN) which is a new architecture for object detection in video. Compared to the state-of-the-art FGFA [18] method, our model involves a simpler design, it does not require optical flow computation and it produces higher video object detection accuracy. Our model is fully differentiable, and unlike prior video object detection methods, it does not necessitate optical flow training data. This renders our model easy to train end-to-end. Our future work will include experimenting with more complex design of spatiotemporal sampling blocks.

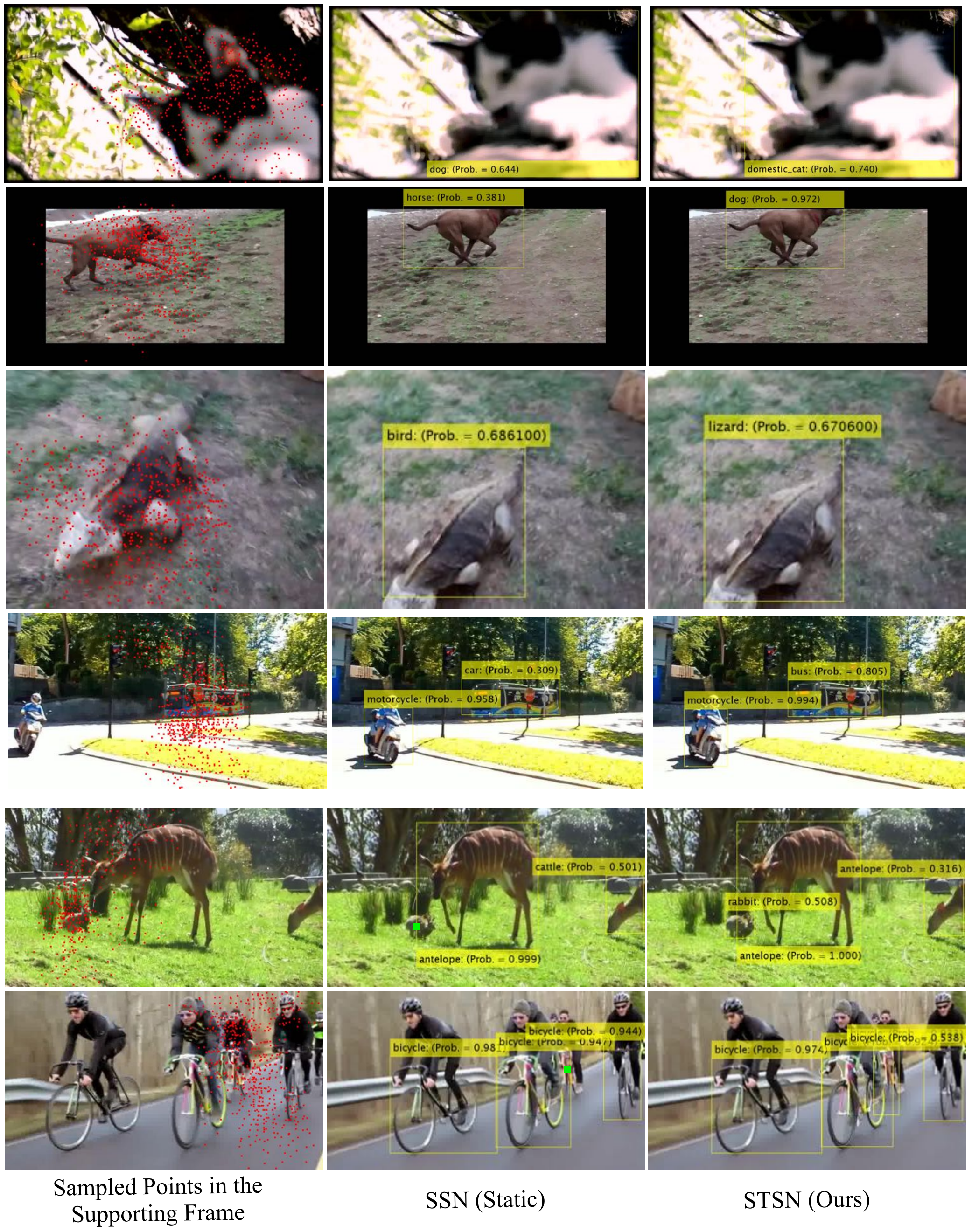


Figure 32: A figure illustrating object detection examples where our spatiotemporal sampling mechanism helps STSN to correct the mistakes made by a static SSN baseline (please zoom-in to see the class predictions and their probabilities). These mistakes typically occur due to occlusions, blurriness, etc. STSN fixes these errors by using relevant object level information from supporting frames. In Column 1 we illustrate the points in the supporting frame that STSN considers relevant when computing the output for a point denoted by the green square in Column 2.

CHAPTER 6 : Learning Discriminative Motion Features Through Detection

6.1. Introduction

Modern CNN based detection models have been highly successful on various image understanding tasks such as edge detection, object detection, semantic segmentation, and pose detection [98, 99, 100, 101, 102, 46, 103, 104, 105, 114, 107, 4, 72, 115, 116, 1, 66]. However, such models lack the means to learn motion cues from video data, and thus, they have not been used as widely for video understanding tasks where motion information plays a critical role.

Prior work [117, 118, 119] has attempted to address this issue by adopting two-stream architectures, where one stream learns appearance features, while the other stream aims to learn motion cues extracted from optical flow inputs. However, these models are expensive since they require processing optical flow with an additional CNN stream.

Recently, there have been many attempts to learn effective video representations with 3D convolutional networks [120, 121]. In principle, such models are capable of capturing salient temporal and motion features, directly optimized for the end task. However, due to their high computational cost, 3D CNNs typically operate on very small spatial resolution inputs, and as a result are not very good at detecting fine grained visual cues such as the pose of a person. Furthermore, it has been recently shown that 3D CNNs optimized from RGB data for action recognition tend to be surprisingly insensitive to temporal ordering of frames [122]. This suggests that in practice they capture little motion information. Thus, it is typically necessary to apply them to optical flow inputs in order to effectively leverage motion.

In this work, we propose a learning scheme that allows modern detection models (e.g., Faster R-CNN) to learn motion cues directly from the RGB video data while being optimized for a discriminative video pose estimation task. Given a pair of annotated frames from the same video—Frame A and Frame B—we train our model to detect pose in Frame A,

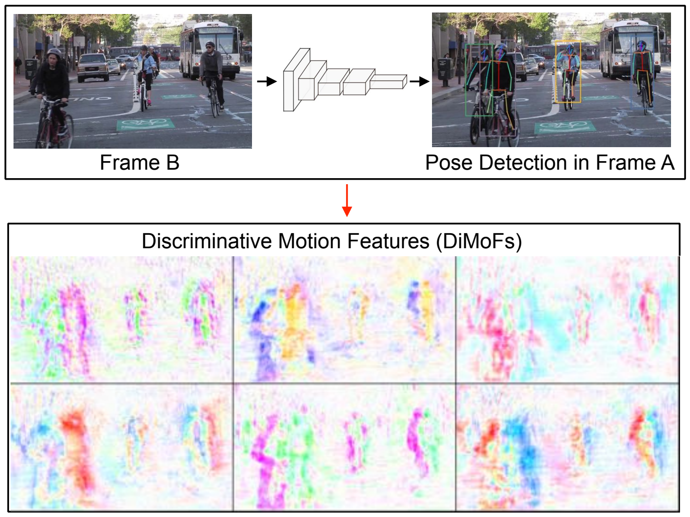

Frame B
Pose Detection in Frame A
Discriminative Motion Features (DiMoFs)

Figure 33: We extend modern detection models (e.g., Faster R-CNN) with the ability to learn discriminative motion features (DiMoFs) from RGB video data. We do so by training the detector on pairs of time-separated frames, with the objective of predicting pose in one frame using features from the other frame. This task forces the network to learn motion "offsets" relating the two frames but that are discriminatively optimized for detection. Our learned DiMoFs can be used on a variety of applications: salient motion localization, human motion estimation, improved pose detection and keypoint tracking, spatiotemporal action localization, and fine-grained action recognition.

using the features from Frame B. To achieve this goal, our model leverages deformable convolutions [111] across space and time. Through this mechanism, our model learns to sample features from Frame B that maximize pose detection accuracy in Frame A. As a byproduct of our optimization, our model also learns "offsets" capturing the motion relating Frame A to Frame B. We refer to these offsets as *DiMoFs* (**Di**scriminative **Mo**tion **F**eatures) to emphasize that they are discriminatively optimized for detection. In our experiments, we show that our DiMoFs model pretrained on pose estimation can subsequently be used for salient motion localization, human motion estimation, improved pose detection, keypoint tracking, spatiotemporal action localization, and fine-grained action recognition.

6.2. Related Work

Detection in Images. Modern object detectors [98, 99, 100, 101, 102, 46, 103, 104, 105, 114, 107] are built using deep CNNs [26, 60, 96]. One of the earlier of such object detection systems was R-CNN [46], which operated in a two-stage pipeline, first extracting object proposals, and then classifying each of them using a CNN. To reduce the computational cost, RoI pooling operation was introduced in [98, 102]. A few years ago, Faster R-CNN [101] replaced region proposal methods by another network, thus eliminating a two stage pipeline. Several methods [114, 107] extended Faster R-CNN into a system that runs in real time with little loss in performance. The recent Mask R-CNN [99] introduced an extra branch that predicts a mask for each region of interest, Finally, Deformable CNNs [111] leveraged deformable convolution to model deformations of objects more robustly. While these prior detection methods work well on images, they are not designed to exploit motion cues in a video– a shortcoming we aim to address.

Detection in Videos. Several recent methods proposed architectures capable of aligning features temporally for improved object detection in video [18, 5, 123]. The method in [123] proposes a spatial-temporal memory mechanism, whereas [5] leverages spatiotemporal sampling for feature alignment. Furthermore, the work in [18] uses an optical flow CNN to align the features across time.

While the mechanisms in [5, 123] are useful for improved detection, it is not clear how to use them for motion cue extraction, which is our primary objective. Furthermore, models like [18] are redundant since they compute flow for every single pixel, which is rarely necessary for higher level video understanding tasks. Using optical flow CNN also adds $40M$ extra parameters to the model, which is costly.

Two-Stream CNNs. Recently, two-stream CNN architectures have been a popular choice for incorporating motion cues into modern CNNs [117, 118, 124, 125, 126, 119, 127, 128]. In these types of models, one stream learns appearance features from RGB data, whereas the other stream learns motion representation from the manually extracted optical flow inputs. The work in [124, 125] leverages two-stream architectures for learning more effective spatiotemporal representations. Recent methods in [129, 130, 127] explored the use of different backbone networks for action recognition tasks. Furthermore, various techniques

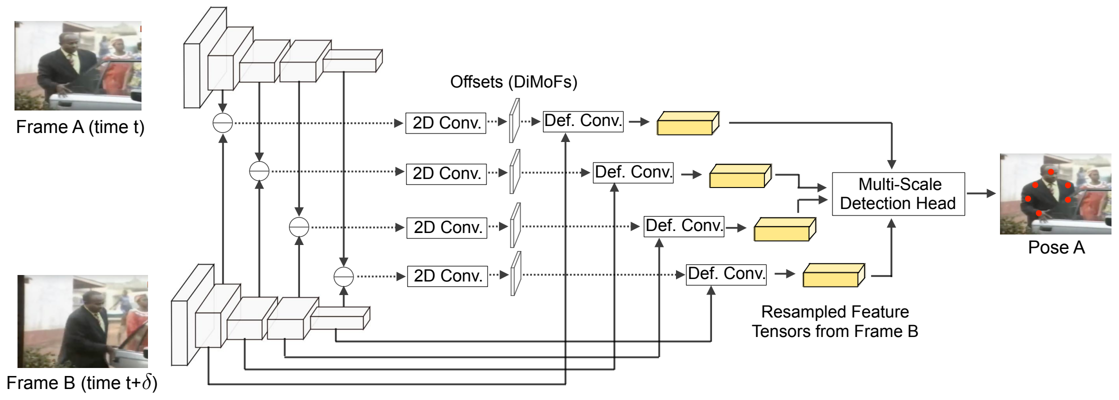


Figure 34: An illustration of our Discriminative Motion Feature (DiMoFs) training procedure. Given Frame A and Frame B, which are separated by δ steps in time, our goal is to detect pose in Frame A using the features from Frame B. First, we extract multi-scale features from both frames via a backbone CNN with shared parameters. Then, at each scale, we compute the difference between feature tensors A and B. From these tensor differences, offsets Δp_n are predicted for each pixel location p_n. The predicted offsets are used to re-sample feature tensor B. As a last step, the resampled feature tensors from each scale are fed into a multi-scale detection head, which is used to predict the pose in Frame A. Our scheme optimizes end-to-end the DiMoFs network so that the feature tensors re-sampled from Frame B maximize the pose detection accuracy in Frame A.

have explored how to fuse the information from two streams [119, 128, 126]. However, these two-stream CNNs are costly and consume lots of memory. Learning discriminative motion cues from the RGB video data directly instead of relying on manually extracted optical flow inputs could alleviate this issue.

3D CNNs. Currently the most common approach for learning features from raw RGB videos is via 3D convolutional networks [120]. Whereas the method in [120] proposes a 3D network architecture for end-to-end feature learning, the recent I3D method [121] inflates all 2D filters to 3D, which allows re-using the features learned in the image domain. Additionally, there have been many recent attempts at making 3D CNNs more effective by replacing 3D convolution with separable 2D and 1D convolutions [131, 24, 132, 133].

While modern 3D CNNs are effective on popular action recognition datasets [121, 134, 135], 3D CNNs are not designed for tasks that require detection of fine-grained visual cues since they operate on small spatial resolution to accommodate long video clips. Furthermore, it has been shown that 3D CNNs do not actually learn motion cues [122], and that they still need to rely on two stream architectures.

Relational Reasoning. There have been several methods modeling temporal relations in videos [136, 23]. The work in [136] learns weights for modeling ”cause and effect” type of relationship. Furthermore, a similar method to ours [23] proposes a temporal relational module for reasoning about temporal dependencies between video frames. However, the proposed relational module does not actually learn motion cues, and works effectively only when videos have strong temporal order [23], which many datasets do not [121, 134, 135]. In contrast, we aim to learn discriminative motion cues, which encode more general information and should be useful even if videos are not temporally ordered.

6.3. Learning Discriminative Motion Features

Our goal is to define a training procedure to learn discriminative motion features (DiMoFs) from RGB videos using a detection model (e.g., Faster R-CNN). Consider the example in Figure 34, which shows a man closing the car door. In order to recognize this action (i.e., closing the door), we do not need to know the motion of every single pixel in the image. Just knowing how the hand of the person moves relative to the car gives us enough information about the action. The key question is: how can we learn such motion information discriminatively? To do this, we formulate the following task. Given two video frames—Frame A and Frame B—our model is allowed to compare Frame A to Frame B but it must predict Pose A (i.e., the pose in Frame A) using the features from Frame B.

At first glance, this task may look overly challenging: how can we predict Pose A by merely using features from Frame B? However, suppose that we had body joint correspondences between Frame A and Frame B. In such a scenario, this task would become trivial, as we would simply need to spatially “warp” the feature maps computed from frame B according to the set of correspondences relating frame B to frame A. Based on this intuition, we design a learning procedure that achieves two goals: 1) By comparing Frame A and Frame B, our

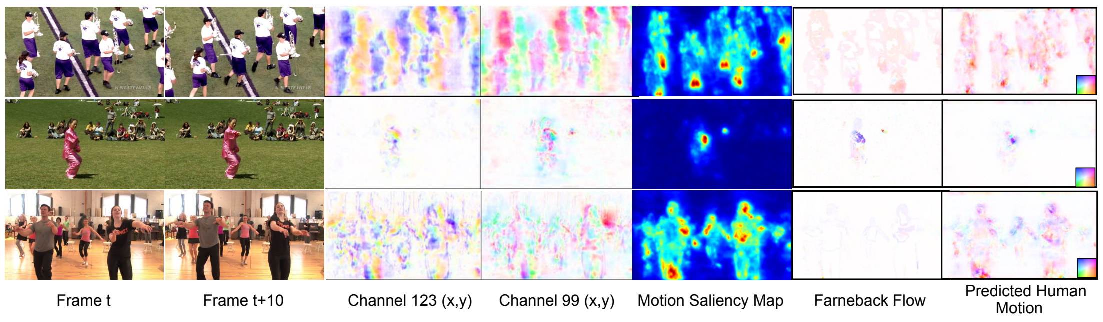


Figure 35: A figure with our DiMoFs visualizations. In the first two columns, we visualize a pair of video frames that are used as input by our model. The 3^{rd} and 4^{th} columns depict 2 (out of 256) randomly selected DiMoFs channels visualized as a motion field. It can be noticed that the DiMoFs capture primarily human motion, as they have been optimized for pose detection. Different channels appear to capture the motion of different body parts, thus performing a sort of motion decomposition of discriminative regions in the video. In the 5^{th} column, we display the magnitudes of summed DiMoFs channels, which highlight salient human motion. Finally, the last two columns illustrate the standard Farneback flow, and the human motion predicted from our DiMoFs. To predict human motion we train a **linear** classifier to regress the (x, y) displacement of each joint from the offset maps. The color wheel, at the bottom right corner encodes motion direction.

model must be able to extract motion offsets relating these two frames. 2) Using these motion offsets our model must be able to rewarp the feature maps extracted from Frame B in order to optimize the accuracy of pose detection in Frame A.

To achieve these goals, we first feed both frames through a backbone CNN with shared parameters. The aim of the backbone is to extract high-level discriminative features facilitating the computation of pose correspondences from the two frames. Then, the backbone feature maps from both frames are used to determine which feature locations from Frame B should be sampled for detection in Frame A. Finally, the resampled feature tensor from Frame B is used as input to the pose detection subnetwork which is optimized to maximize accuracy of Pose A.

Backbone Architecture. As our backbone network we use a Feature Pyramid Network [137] based on a ResNet-101 [96] design. We note, however, that our system is not constrained to this specific architecture design, and that it can easily integrate other backbone architectures as well.

Learning to Sample Features. Initially, we feed Frame A and Frame B through our backbone CNN, which outputs feature tensors $f_A^{(s)}$ and $f_B^{(s)}$ at four scales $s \in \{1, 2, 3, 4\}$. Then, we compute the difference $d_{A,B}^{(s)} = f_A^{(s)} - f_B^{(s)}$, at each of the four scales. The resulting feature tensor $d_{A,B}^{(s)}$ is provided as input to a 2D convolutional layer, which predicts offsets Δp_n (DiMoFs) at all locations p_n. The offsets are used to spatially rewarp (sample) the feature tensor $f_B^{(s)}$.

We implement the sampling mechanism via a deformable convolutional layer [111], which takes 1) the predicted offsets Δp_n, and 2) the feature tensor $f_B^{(s)}$ as its inputs, and then outputs a newly sampled feature tensor $g_{A,B}^{(s)}$. Intuitively, the newly resampled feature tensor $g_{A,B}^{(s)}$ should encode all the relevant information needed for accurate pose detection in Frame A. We use subscript (A, B) for $g_{A,B}^{(s)}$ to indicate that even though $g_{A,B}^{(s)}$ was resampled from tensor $f_B^{(s)}$, the offsets for resampling were computed using $d_{A,B}^{(s)}$, which contains information about both feature tensors. An illustration of our architecture is presented in Figure 34.

Multi-Scale Detection Head. We employ a multi-scale detection head as is done in the Feature Pyramid Network [137]. The outputs of RoI Align [99] applied on the resampled feature tensor $g_{A,B}^{(s)}$ are then fed into three branches optimized for the following 3 tasks: 1) binary classification (person or not person), 2) bounding box regression (predicting region bounds) and 3) pose estimation (outputting a probability heatmap per joint). Our classification and bounding box regression branches are based on the original Faster R-CNN design [101], whereas our pose heatmap branch is designed according to the architecture of the ICCV17 PoseTrack challenge winner [21].

6.4. Interpreting DiMoFs

Before discussing how DiMoFs can be employed in discriminative tasks (Section 6.5), we would like to first understand how the motion cues are encoded in DiMoFs. It turns out that interpreting what the DiMoFs have learned, is nearly as difficult as analyzing any other CNN features [138, 139]. The main challenge comes from the high dimensionality of DiMoFs: we are predicting 256 (x, y) displacements for every pixel at each of the four scales.

In Columns $3, 4$ of Figure 35, we visualize two randomly selected offset channels as a motion field. The DiMoFs visualized here were obtained by training our model for pose estimation on the PoseTrack dataset, as further discussed in the experiments section. Based on this figure, it is clear that DiMoFs focus on people in the videos. However, it is quite difficult to tell what kind of motion cues each of these channels is encoding. Specifically, different DiMoFs maps seem to encode different motions rather than all predicting the same solution (say, the optical flow between the two frames). This makes sense, as our DiMoFs are discriminatively trained. The network may decide to ignore motions of uninformative regions, while different DiMoFs maps may capture the motion of different human body parts (say, a hand as opposed to the head). Thus, our DiMoFs can be interpreted as producing a motion decomposition in different channels of the most discriminative cues in the video.

6.4.1. Human Motion Localization and Estimation

First, we observe that the magnitudes of our learned DiMoFs encode salient human motion, which we visualize in Column 5 of Figure 35. Then, to verify that our learned DiMoFs encode human motion relating the two frames we propose a simple visualization. For each point p_n denoting a body joint, we extract 2048-dimensional (512 channels concatenated across 4 scales) DiMoFs feature vector f_n at that specific point. Then, a *linear* classifier is trained to regress the ground truth (x, y) motion displacement of this body joint from the feature vector f_n. During inference, we apply our trained classifier on every pixel of the image in a fully convolutional manner via a 1×1 convolution.

In Column 7 of Figure 35, we visualize our predicted motion outputs. We show Farneback's optical flow in Column 6 of the same Figure. Note that our predicted human motion matches quite well Farneback optical flow, especially in regions containing people. The fact that we can recover flow with our very simple linear prediction scheme suggests that our learned

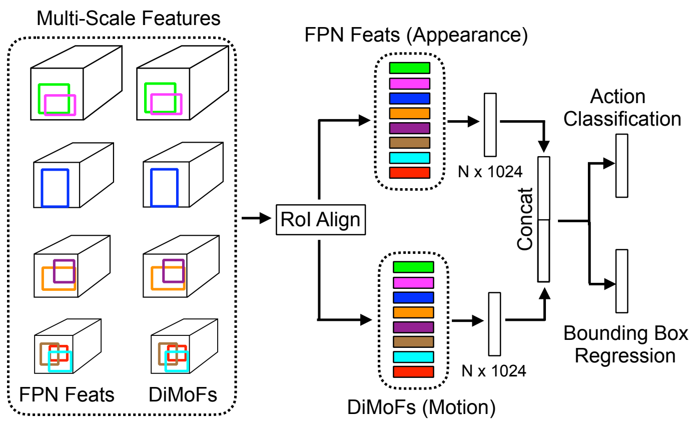


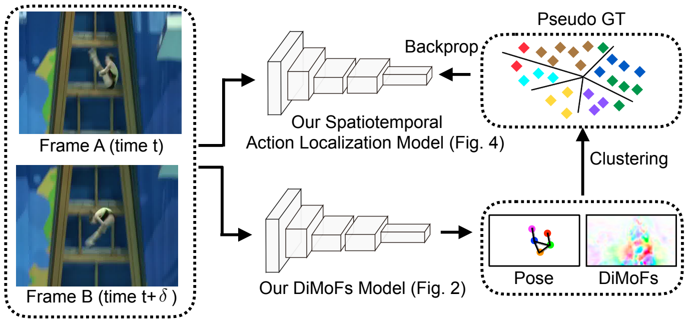
Figure 36: We extend our DiMoFs architecture for spatiotemporal action localization. Given a pair of video frames–Frame A and Frame B—we output for each person detected in Frame A, a bounding box and an action class. Up until the RoI Align, our model operates in the same way as for the pose detection task. Then, RoI Align is applied on 1) the multi-scale FPN feature tensors from Frame A (appearance), and 2) the multi-scale DiMoFs (motion). These RoI features are then fed into separate MLPs, and the resulting 1024-dimensional features are used to predict a bounding box and an action class for each RoI in Frame A.

DiMoFs capture cues related to human motion. Furthermore, we point out that compared to the standard Farneback optical flow, our motion fields look less noisy.

6.5. Detection and Recognition with DiMoFs

In this section, we discuss how DiMoFs benefit pose estimation and tracking. We also show how to adapt the DiMoFs architecture for the tasks of spatiotemporal action localization and fine-grained action recognition.

Pose Detection. During *training* for a video pose detection task (see Section 6.3), we select Frame B by sampling a frame δ time-steps from Frame A, where δ is randomly picked for each training frame from the set $\{-10, \dots, 10\}$. During *inference*, we similarly feed a pair of frames into our model, and output pose predictions for Frame A by using the features from Frame B. Note that this also includes a special case when $\delta = 0$, meaning that Frame B is chosen to be the same as Frame A. In this special case, we can simply initialize the feature difference tensors $d_{A,B}^{(s)}$ to zeros, and then proceed as before. This allows us to train our pose detector on videos, and then later test it on still-images.



Frame A (time t)
Frame B (time t+δ)
Our Spatiotemporal Action Localization Model (Fig. 4)
Backprop
Pseudo GT
Clustering
Our DiMoFs Model (Fig. 2)
Pose
DiMoFs

Figure 37: A high-level overview of our approach for learning fine-grained action recognition features. Initially, we use our DiMoFs model to extract pose and DiMoFs features from a given pair of video frames. We then accumulate these pose and DiMoFs features across the entire Diving48 training set, and cluster them using a k-means algorithm. The resulting cluster assignment IDs are then used as pseudo ground truth labels to optimize our action recognition features, as illustrated in Figure 36.

Keypoint Tracking. We consider the problem of tracking human body keypoints in video. We approach this task using the same ICCV17 PoseTrack winning model [21] as we did for pose detection. The tracking system in [21] consists of two stages: 1) keypoint estimation in frames, and 2) bipartite graph matching to link the tracks across adjacent frames. We simply replace the baseline model in stage 1) with our DiMoFs learned model. Note that both networks have the same network architecture, but differ in their training procedures. As will be shown later, our DiMoFs training scheme leads to better keypoint tracking results.

6.5.1. Spatiotemporal Action Localization

We introduce a few simple modifications to incorporate our learned DiMoFs into the Faster R-CNN architecture for spatiotemporal action localization (see Figure 36). Just as before, the model takes a pair of video frames as input. However, in this case the model outputs bounding boxes for Frame A, and an action class for each bounding box. We conjecture that a good spatiotemporal action localization model needs 1) strong visual appearance features associated with Frame A, and 2) motion features that encode how the person is moving between Frames A and B. While the appearance information is provided by the Frame A features computed through the backbone network, the motion cues are captured by our learned DiMoFs.

We keep the operations in the backbone the same as they were for the pose detection task. To adapt our pose estimation architecture to the problem of action localization, we remove all deformable convolutional layers as we do not need to resample features from B into Frame A anymore. Afterwards, we predict regions of interest (RoIs) in the same manner as we did for the pose detection task and then apply RoI Align separately on the visual features, and on the DiMoFs. Afterwards, separate 2-layer MLP head is applied to each type of features, and the resulting feature tensors are concatenated together to predict the bounding box, as well as its associated action class (See Figure 36).

6.5.2. Fine-Grained Action Recognition

Our model operates on frame pairs and it is not designed to process long video sequences, unlike 3D CNNs for action recognition [120, 121, 131, 24, 132, 133]. However, our DiMoFs model is explicitly designed to detect fine-grained cues such as human pose and subtle movements of various body parts. This suggests that DiMoFs can potentially be leveraged for fine-grained action recognition. We achieve this goal through a procedure inspired by the work of Caron et al. [140] who propose a two-stage deep clustering method for unsupervised learning of still-image features. Initially, we use our DiMoFs model (see Fig. 34) to extract the coordinates of each body part of every person in a video. We also extract DiMoFs features associated with each body joint and reduce them to 15 dimensions using PCA. We then concatenate these features and cluster them via K-means using $K = 100$ clusters. Intuitively, such a clustering procedure should yield clusters sharing similar pose and body motion patterns. We illustrate this procedure in Figure 37.

Afterwards, we train our spatiotemporal action localization network (see Figure 36) to predict such cluster IDs. Thus, instead of predicting action classes, we use cluster IDs as pseudo-ground truth labels. The rationale behind this choice is that directly predicting action classes from pairs of frames is a nearly-impossible task, especially in the case of fine-grained classification where the same poses occur in many different categories. On the other hand, predicting pose cluster IDs is a better posed task. At the same time, the aggregation

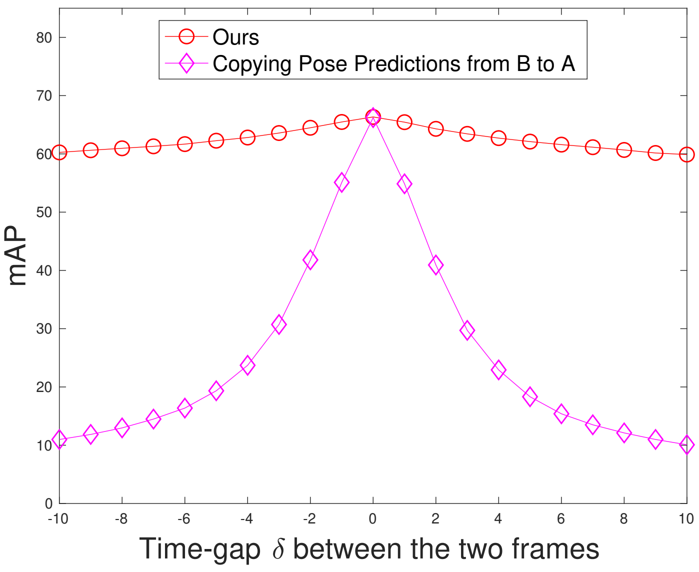


Figure 38: Pose detection results on the PoseTrack dataset. Our model predicts pose in Frame A using features from a Frame B that is δ time-steps away. During **training**, our model is optimized on frame pairs with random time-gaps $\delta \in \{-10, \ldots, 10\}$. Here, we present the results obtained by using different values of δ **at inference** time. As δ becomes large the motion between frames may become substantial. Yet, the accuracy of our model drops gracefully as δ deviates from 0. This confirms that our model estimates the human motion between the two frames reliably.

of pose cluster IDs over the entire video (e.g. in the form of a simple histogram) can provide a strong descriptor for action recognition, disambiguating the recognition problem we had in a single pair of frames. Inspired by this intuition, we initialize our spatiotemporal action localization with weights learned during our DiMoFs training procedure and optimize it for pose cluster ID prediction. Afterwards, we evaluate the model on all pairs of frames in the video (sampled with time-gap $\delta = 5$) and obtain the 100 dimensional pose cluster ID prediction for each pair. The final classification is performed by training a shallow 2-layer MLP on the 100-dimensional vector obtained by summing up the pose ID activations for all pairs in the same video.

6.6. Experiments

In this section, we present results showing the benefit of our DiMoFs on the tasks discussed in the previous section: 1) pose detection, 2) keypoint tracking, 3) spatiotemporal action

Method	Head	Shou	Elbo	Wri	Hip	Knee	Ank	Mean
Girdhar et al. [21]	72.8	75.6	65.3	54.3	63.5	60.9	51.8	64.1
[21] + DiMoFs	**75.2**	**78.5**	**66.8**	**56.0**	**66.7**	**62.7**	**54.0**	**66.3**

Table 20: Pose detection results on the PoseTrack dataset measured in mAP. The first row reports the results of a single-frame baseline [21], which won the ICCV17 PoseTrack challenge. In the second row, we present the results of our method, which is based on the same exact model as [21] but it is *trained* on pairs of frames with our DiMoFs training scheme (the *testing* is still done on individual frames). These results indicate consistent performance gains for each body joint, and a mean mAP improvement of 2.2%.

localization, and 4) fine-grained action recognition.

6.6.1. Pose Detection and Tracking on PoseTrack

In this section, we train and test our method on the PoseTrack [141] dataset, which contains 514 videos, 300 for training, 50 for validation and 208 for testing. The dataset includes 23,000 frames with annotated poses. Our evaluations are performed on the validation set. We demonstrate the effectiveness of our DiMoFs procedure by showing improved results for both pose estimation and keypoint tracking with respect to an analogous model trained on individual frames and thus unable to use motion information.

The Figure 38 illustrates our pose detection results according to the mAP metric for different values of the time-gap δ used at *testing* time. The δ values on the x-axis of Figure 38 captures how far apart Frames A and B are from each other. As expected, the more δ deviates from 0, the lower the accuracy becomes because the motion between two frames becomes more severe. We discuss below two key findings from the results we obtain.

Results when $\boldsymbol{\delta \neq 0}$. We first observe that the accuracy of our model drops quite gracefully as the time-gap δ deviates from 0. By comparison, note the poor performance of the magenta curve which depicts a naïve baseline that simply copies pose detections from B to A (i.e., without using Frame A at all). For $\delta = \pm 10$, the accuracy of such baseline drops by 56.5% mAP with respect to the single-frame baseline [21]. In contrast, our model drops by only 6.1% mAP for the same $\delta = \pm 10$. The ability to predict poses from far-away frames

Method	Head	Shou	Elbo	Wri	Hip	Knee	Ank	Mean
Girdhar et al. [21]	61.7	65.5	57.3	45.7	54.3	53.1	45.7	55.2
[21] + DiMoFs	**62.9**	**67.2**	**57.4**	**45.8**	**55.5**	**53.5**	**46.5**	**56.0**

Table 21: Keypoint tracking results on the PoseTrack dataset using Multi-Object Tracking Accuracy (MOTA) metric. As our baseline, we use the ICCV17 PoseTrack challenge winning method [21]. Our model is trained using the same architecture as [21], but using our DiMoFs training scheme.

suggests that our model reliably estimates the motion between the two frames and warps accurately the features of frame B for detection in frame A.

Results when $\delta = 0$. In Table 20, we compare our model with the single frame baseline of Girdhar et al. [21], which won the ICCV17 PoseTrack challenge. We note that our method is based on the same CNN architecture as [21], but is instead *trained* on pairs of frames using our DiMoFs training procedure. The *testing* when $\delta = 0$, however, is done on individual frames for both methods.

We observe that DiMoFs outperforms this strong baseline for all joints and by 2.2% on average. What are the reasons behind these gains? This happens primarily because our DiMoFs training procedure on pairs of frames enables a form of data augmentation. In a single-frame training setting, the model learns a pattern connecting Frame A to Pose A. However, in our DiMoFs training, for each Frame A the model is learning patterns connecting many different Frames B (as δ is varied) to Label A. Thus, our model leverages many more (frame, label) pairs, which is beneficial.

Keypoint Tracking. In Table 21, we also demonstrate that our DiMoFs training procedure is helpful for the keypoint tracking task on the same PoseTrack dataset. The results are measured using the standard Multi-Object Tracking Accuracy (MOTA). We observe that DiMoFs yields gains in keypoint tracking across all body joints although not as substantial as in the case of pose estimation. We note that both methods in Table 21 use the same CNN architecture, but our model is trained using DiMoFs procedure.

6.6.2. Results of Action Localization on JHMDB

Next, we evaluate the effectiveness of our spatiotemporal action localization model (described in subsection 6.5.1). To do this, we use the JHMDB [142] dataset, which contains 928 temporally-trimmed clips representing 21 action classes, with bounding box annotations. We report our results on split 1 using the standard frame-mAP metric with an intersection-over-union (IOU) threshold set to 0.5.

To show that our new model can leverage the learned DiMoFs for spatiotemporal action localization, we test our model by feeding it pairs of frames, where Frame B is sampled by choosing $\delta = \{0, 2, 4, 6, \ldots, 38, 40\}$. We illustrate these results in Figure 39, which displays spatiotemporal action localization mAP versus the time-gap δ between the two frames. From this figure, we observe that the accuracy is at its lowest when $\delta = 0$, which makes sense because the model cannot leverage any motion cues. However, once we increase δ, the accuracy starts climbing steeply, which suggests that the model is leveraging the learned DiMoFs features for better spatiotemporal action localization performance. Based on the figure, we observe that the model reaches its peak performance at $\delta = 12$, which corresponds to approximately 0.5s in the original video. After that, the accuracy starts going down, which is expected because 1) the frames that are further away are less informative, and 2) it becomes harder to estimate long-range motion.

Furthermore, in Table 22, we ablate how different factors affect spatiotemporal action localization performance. First, we note that inflating a standard Faster R-CNN with the same backbone as our model to 3D [121] produces poor results (see column 1 of Table 22). Next, in columns 2 and 3, we report the accuracy of the model that uses exactly the same model architecture but where DiMoFs have not been trained for detection. From the table, we observe that such a model performs substantially worse (44.3% and 44.7% for $\delta = 0$ and 10 respectively) than our DiMoFs model (49.6% and 54.4% for $\delta = 0$ and 10 respectively). Furthermore, we point out that a model without DiMoFs training is not able to exploit motion cues effectively, which is indicated by a marginal 0.4% improvement between $\delta = 0$

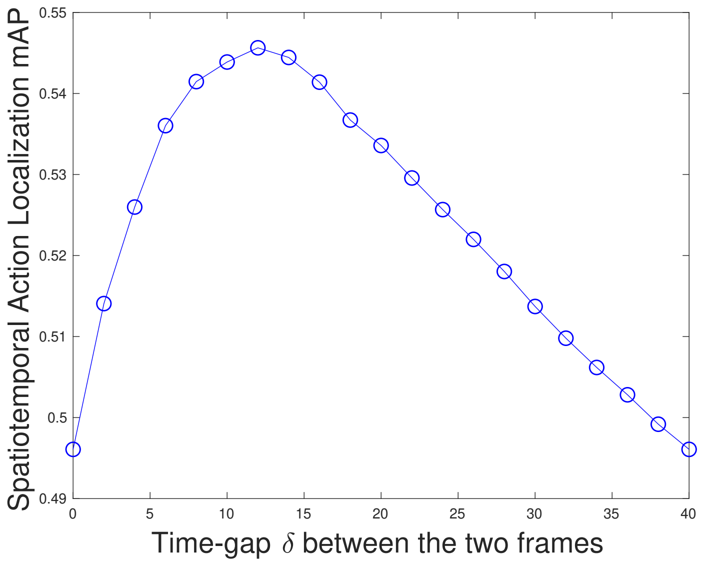


Figure 39: Our spatiotemporal action localization results on JHMDB dataset as we vary the time-gap δ separating Frame B from Frame A during inference. Based on these results, we observe that the performance is lowest at $\delta = 0$, which makes sense as there are no motion cues to leverage. However, once δ gets larger, the accuracy increases sharply indicating that the learned DiMoFs contain useful motion cues for spatiotemporal action localization task.

and $\delta = 10$ settings. In contrast, our model exhibits a substantial 4.8% performance boost when increasing the δ from 0 to 10, which suggests the importance of learning DiMoFs discriminatively through a detection task. It is also worth noting, that pose information learned via DiMoFs training is important, i.e., even in a setting where $\delta = 0$, our DiMoFs model outperforms an equivalent Faster R-CNN by 5.3% (see columns 2 and 4 in Table 22). Lastly, we point out that learning the appearance features as before, but replacing implicit motion cues encoded by DiMoFs with explicit optical flow yields 50.6% mAP at $\delta = 10$, which is substantially worse than our 54.4%.

We also note that while models pretrained using a Kinetics dataset yield better performance [143, 144, 145], those results are not directly comparable to our evaluation since Kinetics is larger and contains many examples from the same classes as JHMDB. In this section, we aim to show that our model successfully leverages DiMoFs to improve action localization without resorting to additional action labels.

	1	2	3	4	5	6
DiMoFs Training?				✓		✓
Replacing DiMoFs with Optical Flow?					✓	
Faster R-CNN Inflated to 3D?	✓					
$\delta = 0$ at Inference?		✓		✓		
$\delta = 10$ at Inference?			✓		✓	✓
Mean Frame Average Precision (mAP)	43.8	44.3	44.7	49.6	50.6	**54.4**

Table 22: We examine how various factors affect spatiotemporal action localization performance on JHMDB. We observe that learning DiMoFs discriminatively through detection outperforms the model with the same architecture but that was not trained with DiMoFs scheme by 5.3% and 9.7% mAP for $\delta = 0$, and $\delta = 10$ respectively (compare column 2 to 4, and 3 to 6). We also note that replacing DiMoFs offsets with optical flow yields 50.6% mAP, which is 3.8% worse than using DiMoFs (column 5 vs 6).

6.6.3. Fine-Grained Action Recognition on Diving48

Finally, we evaluate our DiMoFs model on a fine-grained action recognition task (see subsection 6.5.2 for model description). For this experiment, we use the newly released Diving48 dataset [146], which contains 15,943 training and 2096 testing videos of professional divers performing 48 types of dives. We choose this dataset because unlike datasets such as Kinetics or UCF101, Diving48 is designed to minimize the bias towards particular scenes or objects. To do well on Diving48 dataset, it is necessary to model the subtle differences in body motion cues, and use them to predict the action class (i.e., the type of dive).

In Table 23, we present our quantitative results. In the top half of the table, we examine performance of our model in comparison to recent 2D CNNs: TSN [22] and TRN [23]. We also include an interesting baseline that uses directly the histogram of cluster ID from pose detection (bottom branch in Fig. 37) as features for action recognition (without training the top-branch in Fig. 37). This yields a pretty solid accuracy of 17.2%. By including DiMoFs (in addition to pose) for clustering, the accuracy jumps to 21.4%. Note, that if we replace implicit motion cues encoded by DiMoFs with explicit optical flow the accuracy drops to 18.8%, which is substantially worse than our 21.4%. We also point out that training our spatiotemporal action recognition model to predict pose cluster IDs further improves the

2D Models	Pre-training Data	Accuracy
TSN (RGB) [22]	ImageNet (objects)	16.8
TSN (RGB + Flow) [22]	ImageNet (objects)	20.3
TRN [23]	ImageNet (objects)	22.8
Ours-init (Pose)	PoseTrack (poses)	17.2
Ours-init (Pose+Optical Flow)	PoseTrack (poses)	18.8
Ours-init (Pose+DiMoFs)	PoseTrack (poses)	21.4
Ours-final (Pose+DiMoFs)	PoseTrack (poses)	**24.1**

3D Models	Pre-training Data	Accuracy
R(2+1)D [24]	None	21.4
C3D [120]	Sports1M (actions)	27.6
R(2+1)D [24]	Kinetics (actions)	28.9
Ours + R(2+1)D [24]	Kinetics + PoseTrack	**31.4**

Table 23: Our results on Diving48 dataset. In the top half of the table, we show that our system achieves 24.1% accuracy and outperforms 2D methods TSN [22] and TRN [23]. We also observe that even using the initial cluster ID histograms as features produces solid results (i.e. 17.2% and 21.4%). Note that replacing DiMoFs with optical flow reduces the accuracy by 2.6% (compared to our 21.4%). Our model does not perform as well as the pre-trained R(2+1)D [24] baseline. However, our pre-training is done on a much smaller dataset, and on a more general pose detection task. Without Kinetics pre-training, we outperform R(2+1)D [24] by a solid 2.7% margin. Finally, we show that combining DiMoFs with pre-trained R(2+1)D [24] improves the results, suggesting complementarity of the two methods.

performance, and allows our model to achieve 24.1%. These results suggest 1) the usefulness of our learned pose and DiMoFs features, and also 2) highlight the benefits of optimizing the network to pseudo ground-truth pose cluster IDs.

In the lower half of the table, we compare our method with popular 3D models [120, 24] . Our model does not perform as well as the pre-trained long-range 3D CNNs. However, this comparison is not exactly fair as these 3D models have been pre-trained on larger-scale action recognition dataset such as Kinetics [121]. In contrast, our model is pre-trained on a much smaller PoseTrack dataset, which requires significantly less computational resources. Furthermore, our pre-training is done on a more general pose estimation task, which allows our learned DiMoFs to generalize to a variety of different tasks as shown in the paper. We note that without Kinetics pre-training, our model outperforms R(2+1)D [24] baseline by a solid 2.7% margin. Finally, we also show that combining our model with the pre-trained R(2+1)D [24] allows us to further improve the performance, and achieve state-of-the-art results, which suggests that the two models learn complementary cues.

6.7. Conclusions

In this work, we introduced DiMoFs, discriminatively trained offsets that encode human motion cues. We showed that our DiMoFs can be used to localize and estimate salient human motion. Furthermore, we show that as a byproduct of our DiMoFs learning scheme, our model also learns features that lead to improved pose detection, and better keypoint tracking. Finally, we showed how to leverage our learned DiMoFs features for spatiotemporal action localization and fine-grained action recognition tasks. Our future work involves designing unified CNN architectures that can leverage the strengths of modern 3D CNNs, and detection-based systems. We will release our source code and our trained models upon publication of the paper.

Part III

Seeing to Learn

CHAPTER 1 : First-Person Action-Object Detection with EgoNet

1.1. Introduction

Our visual sensation is developed along with the neuromotor system while interacting with surrounding objects [147, 148, 149, 150, 151]. As the visual sensation and motor signal reinforce each other, it characterizes the way we progressively interact with objects in 3D, which provides a strong cue to understand the core attributes of a person's behavior such as his attention or intentions. For instance, consider a woman entering a canned food corner at a grocery store as shown in Figure 40. When she schemes through hundreds of canned foods to find the tuna can that she looks for, she remains 3-5m from the food stand for efficient search. Once she finds the tuna, she approaches it (1-3m), and then reaches her left hand to pick the tuna can (<1m). While she gazes at the expiration date in the label of the can, the distance gets smaller (<0.5m). Not only does the tuna can stimulate her visual attention but it also affects her physical actions, such as head or hand movements.

We define such object as an action-object—an object that triggers conscious visual and motor signals. The key properties of an action-object are: (1) it facilitates a person's tactile (touching a cup) or (2) visual (watching a TV) interactions and (3) it exhibits a characteristic distance and orientation with respect to the person.

While in-situ wearable sensors such as gaze tracker with EEG measurements or tactile and force/torque sensors for muscle movement are viable solutions to identify action-objects more accurately than distant sensors such as third-person mounted cameras, their integration into our daily life is highly limited. A fundamental question is "can we detect action-objects as we observe the person interacting with her/his environment from a first-person video alone?". This is challenging despite recent success of robot/computer vision systems because (a) a person's gaze direction does not necessarily correspond to action-objects. In other words, not all objects within the person's field of view are consciously attended; (b) action-objects are often task-dependent, which makes it difficult to detect them without

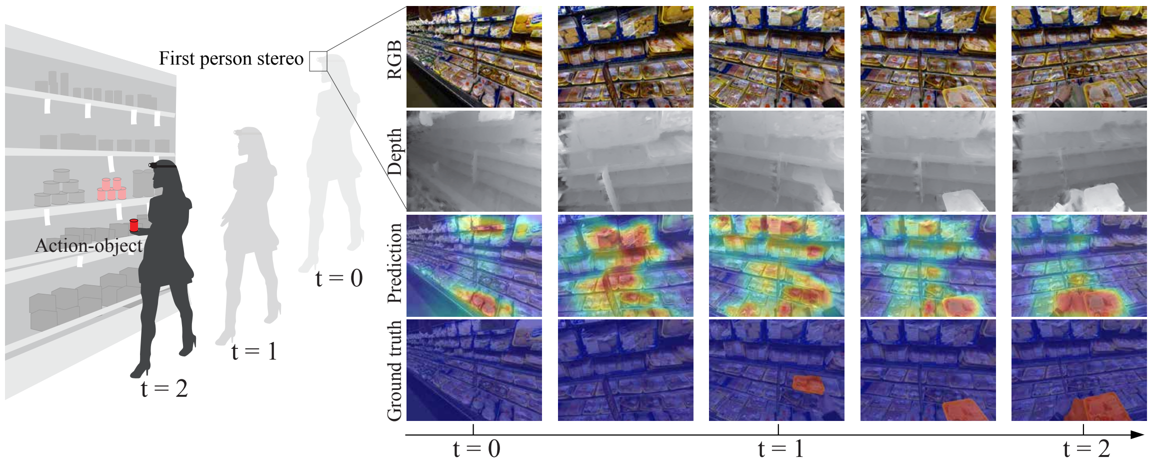


Figure 40: We predict action-objects from first-person RGBD images (best viewed in color) where action-objects are defined as objects that facilitate people's conscious tactile (grabbing a food package) or visual interactions (watching a TV). Left: a woman approaches a shelf to pick up a food item (red). Right: The food (action-object) is detected progressively as she approaches and reaches her hand to pick it up.

knowing the task beforehand; (c) action-objects are not specific to object category, i.e., many object categories correspond to the same action, e.g., TV and a mirror both afford a seeing action. Therefore, an object specific model cannot represent action-objects.

In this paper, we address these challenges by leveraging a first-person stereo camera system and our proposed EgoNet model, a joint two-stream network that integrates visual appearance (RGB) and 3D spatial cues (depth and height). These two pathways are complementary: one of which learns visual appearance cues, while the other exploits 3D spatial information indicative of action-objects. These are combined via a joint pathway, which incorporates a first-person coordinate embedding that learns an action-object spatial distribution in the first-person image. The entire network is jointly optimized to the action-object ground truth that is provided by the camera wearer. We quantitatively justify the architecture choices of our EgoNet model and show that it outperforms all prior approaches for the action-object detection task across multiple first-person datasets. We also show that EgoNet generalizes well on a variety of novel datasets, even without being adapted to a specific task as is commonly done [152, 153, 154].

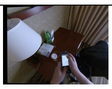
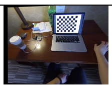
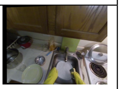
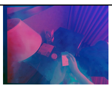
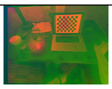
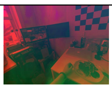
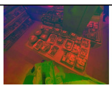
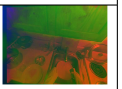
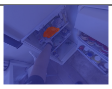
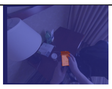
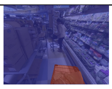
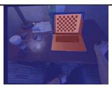
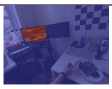
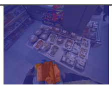
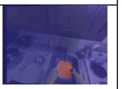
Scene	cooking	hotel	grocery	desk work	dining	shopping	washing
Frames	1030	410	463	491	515	646	674
RGB							
DHG							
Ground Truth							

Table 24: The summary of our first-person action-object RGBD dataset, which captures people performing 7 different activities. Our dataset contains 4247 frames with RGB, and DHG (depth, height, grayscale image) inputs as well as annotated per-pixel action-object masks.

In conjunction with the EgoNet, we present a new first-person action-object RGBD dataset that includes object interactions during diverse activities such as cooking, shopping, dish-washing, working, etc. The camera wearer who is aware of the task and who can disambiguate conscious visual attention and subconscious gaze activities provides per-pixel binary labels of first-person images, which we then use to build our action-object model in a form of EgoNet.

Our framework is different from a classic object detection task because action-objects are associated with actions without explicit object categories. It also differs from a visual saliency detection because visual saliency does not necessarily correspond to a specific action. Finally, our action-object task differs from activity recognition because we detect action-objects by exploiting common person-object spatial configurations instead of modeling activity-specific interactions, which makes our model applicable to different activities.

Why First-Person? Precisely identifying action-objects is critical for measuring the internal state of humans and answering questions such as "what is that person doing?", "what will he do next?". However, answering these questions from a third-person perspective is often challenging because a person's actions are captured from a relatively large distance and possibly from a suboptimal orientation, which makes it difficult to recognize that person's actions and understand his intentions. Furthermore, a first-person view contains inherent

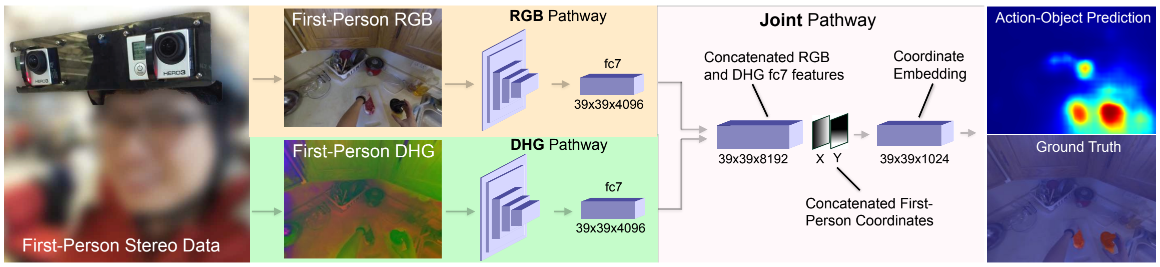


Figure 41: Our proposed EgoNet architecture (best viewed in color) takes as input first-person RGB and DHG images, which encode 2D visual appearance and 3D spatial cues respectively. The fully convolutional RGB pathway then uses the visual appearance cues, while the fully convolutional DHG pathway exploits 3D spatial information to detect action-objects. The information from both pathways is combined via the joint pathway, which also implements the first-person coordinate embedding, and then outputs a per-pixel action-object probability map.

task-intention via a person's head and body orientation relative to the objects, and therefore, the task information does not need to be modeled explicitly using action recognition as is commonly done in prior work [154, 152].

1.2. Related Work

Complementary Object and Activity Recognition in Third-Person. Actions are performed in the context of objects. This coupling provides a complementary cue to recognize actions. Wu et al. [155] leveraged object information to classify fine-grained activities. Yao and Fei-Fei [156, 157] have presented a spatial model between human pose and objects for activity recognition. Some approaches also used low level bag-of-feature models to learn the spatial relationship between objects and activities from a single third-person image [158]. Conversely, the activity can provide a functional cue to recognize objects [159, 160, 161]. Such a model becomes even more powerful when incorporating the cues of how the object is physically manipulated [162, 163, 164]. In addition, object affordance can be learned by simulating human motion in the 3D space [165, 166]. Furthermore, Gori et al. [167] proposed to recognize activities from a third-person robot's view using people detections.

Whereas most of these methods require detected people or body pose as an input, our work leverages a first-person view and does not need to detect people or human pose a priori.

First-Person Object Detection. There exist multiple prior methods that explore object detection from first-person images as a main task [153, 168], or as an auxiliary task for activity recognition [169, 152, 154, 170, 171] or video summarization [172, 173]. Below we summarize how our action-object detection task is different from this prior work.

The work in [152, 174] attempts to predict gaze from the first-person images and use it for activity recognition. However, we know that a person's gaze direction does not always correspond to action-objects but instead capture noisy eye movement patterns, which may not be useful for activity recognition. In the context of our problem, the camera wearer who was performing the task and who can disambiguate conscious visual attention and subconscious gaze activities provides per-pixel binary labels of the first-person images, which we then use to build our action-object model.

The methods in [169, 152] perform object detection and activity recognition disjointly: first an object detector is applied to find all objects in the scene, and then those detections are used for activity recognition without necessarily knowing, which objects the person may be interacting with. Furthermore, these methods employ a set of predefined object classes. However, many object categories can correspond to the same action, e.g., TV and a mirror both afford a seeing action, and thus, an object class specific model may not be able to represent the action-objects accurately.

Some prior work focused specifically on handled object detection [153, 175]. However, action-object detection task also requires detecting conscious visual interactions that do not necessarily involve hand manipulation (e.g. watching a TV). Furthermore, from a development point of view, conscious visual attention is one way for a person to interact. For instance, for the babies who lack motor skills, their conscious visual attention is the only thing that indicates their action-objects, and thus detecting only handled objects is not enough.

The most relevant to our action-object detection task is the work in [154] that detects objects of interest for activity recognition [154]. However, this method is designed specifically for recognizing various cooking activities, which requires detecting mostly handled-objects as in [153]. Thus, the authors [154] manually bias their method to recognize objects near hands. Such an approach may not work for other types of activities that do not involve hand manipulation such as watching a TV, interacting with a person, etc.

Finally, none of the above methods, fully exploit common person-object spatial configuration. We hypothesize that a first-person view contains inherent task-intention via a person's head and body orientation relative to the objects. In other words, during an interaction with an object, people position themselves at a certain distance and orientation relative to that object. Thus, 3D spatial information provides essential cues that could be used to recognize action-objects. We leverage such 3D cues, by using first-person stereo cameras, and building an EgoNet model that uses 3D depth and height cues to reason about action-objects.

Novelty of Action-Object Concept. We acknowledge that our defined concept of action-objects overlaps with several concepts from prior work such as object-action complexes (OAC) [176], handled-objects [153, 175], objects-in-action [162], or object affordances [161]. However, we point out that these prior methods typically focus exclusively on physically manipulated objects, that are specific to certain tasks (e.g. cooking). Instead, the concept of action-objects requires detecting not only tactile but also conscious visual interactions with the objects (e.g. watching a TV), without making any a-priori assumptions about the task that the person will be performing as is commonly done in prior work [153, 175].

Structured Prediction in First-Person Data. A task such as action-object detection or visual saliency prediction requires producing a dense probability output for every pixel. To achieve this goal most prior first-person methods employed a set of hand-crafted features combined with a probabilistic or discriminative classifier. For instance, the work in [172] uses manually engineered set of egocentric features with a linear regression classifier to assign probabilities to each region in a segmented image. The method in [177] exploits the combination of geometric and egocentric cues and trains random forest classifier to predict saliency in first-person images. The work in [153] uses optical flow cues and Graph Cuts [77] to compute handled-object segmentations, whereas [168] employs transductive SVM to compute foreground segmentation in an unsupervised manner. Finally, some prior work [174] integrates a set of hand-crafted features in the graphical model to predict per pixel probabilities of camera wearer's gaze.

We note that the recent introduction of the fully convolutional networks (FCNs) [27] has led to remarkable results in a variety of structured prediction tasks such as edge detection [2, 66, 90] and semantic image segmentation [29, 72, 82, 75, 83, 74, 88, 84]. Following this line of work, a recent method [154], used FCNs for joint object segmentation and activity recognition in first-person images using a two stream appearance and optical flow network with a multi-loss objective function.

We point out that these prior methods [154, 174, 168, 153, 77] focus mainly on the RGB or motion cues, which is a very limiting assumption for an action-object task. When interacting with an object, people typically position themselves at a certain distance and orientation relative to that object. Thus, 3D information plays an important role in action-object detection task. Unlike prior work, we integrate such 3D cues into our model for a more effective action-object detection.

Additionally, the way a person positions himself during an interaction with an object, affects where the object will be mapped in a first-person image. Prior methods [172, 177] assume that this will most likely be a center location in the image, which is a very general assumption. Instead, in this work, we introduce the first-person coordinate embedding to learn an action-object specific spatial distribution.

In this work, we show that our proposed additions are simple and easy to integrate into existing FCN framework, and yet they lead to a significant improvement in the action-object detection accuracy in comparison to all the prior methods.

1.3. First-Person Action-Object RGBD Dataset

We use two stereo GoPro Hero 3 cameras with 100mm baseline to capture first-person RGBD videos as shown in Figure 41. The stereo cameras are synchronized manually and each camera is set to 1280×960 with 100 fps. The fisheye lens distortion is pre-calibrated and depth image is computed by estimating disparities between cameras via dense image matching with dynamic programming.

Two subjects participated in capturing their daily interactions with objects in activities such as cooking, shopping, working at their office, dining, buying groceries, dish-washing, and staying in a hotel room. 7 scenes were recorded and 4229 frames with per-pixel action-objects were annotated by the subjects with GrabCut [178].

In Table 24, we provide a brief summary of our first-person action-object dataset. The dataset consists of 7 sequences, which capture various people's interactions with objects. In comparison to the existing first-person datasets such as GTEA Gaze+ [152], which records a person's interactions only during a cooking activity, our dataset captures more diverse person's interaction with objects. This allows us to leverage common person-object spatial configurations and build a more general action-object model without constraining ourselves to a specific task, as is done in [152]. In the experimental section, we will show that our model generalizes well on a variety of first-person datasets even if they contain previously unseen scenes, objects or activities.

1.4. EgoNet

In this section, we describe EgoNet, a predictive network model that detects action-objects from a first-person RGBD image. EgoNet is a two-stream FCN that holistically integrates visual appearance, head direction, and 3D spatial cues, and that is specifically adapted for first-person data via a first-person coordinate embedding. EgoNet consists of 1) an RGB pathway that learns object visual appearance cues; 2) a DHG pathway that learns to detect action-objects based on 3D depth and height measurements around the person; and a 3) a joint pathway that combines the information from both pathways, and which also incorporates our proposed first-person coordinate embedding to model a spatial distribution of action-objects in the first-person view. The detailed illustration of EgoNet's architecture is presented in Figure 41. We now explain each of EgoNet's components in more detail.

1.4.1. RGB Pathway

An action-object that stimulates person's visual attention typically exhibits a particular visual appearance. For instance, we are more likely to look at the objects that are colored brightly and stand out from the background visually. Thus, our model should detect visual appearance cues that are indicative of action-objects. To achieve this goal we use DeepLab [29], which is a fully convolutional adaptation of a VGG network [60]. DeepLab has been shown to yield excellent results on problems such as semantic segmentation [29]. Just like the segmentation task, action-object detection also requires producing a per-pixel probability map. Thus, inspired by the success of a DeepLab system on semantic segmentation task, we adopt a pretrained DeepLab's network architecture as our RGB pathway.

1.4.2. DHG Pathway

An action-object also possesses the following 3D spatial properties. It exhibits characteristic distance to the person due to anthropometric constraints, e.g., arm length. For example, when a man picks up a tuna can, his distance from the can is approximately 0.5m. The action-object also has a specific orientation relative to a person because of its design. For instance, when the person carries a cup, he holds it via the handle, which determines the pose of the cup with respect to that person. These 3D spatial properties are essential for predicting the action-objects.

Considering this intuition, we use our collected first-person stereo data to encode spatial 3D cues in a DHG (depth, height, and grayscale) image input. We use depth and height [69] to represent the 3D environment around the person, and to handle the pitch movements of the head, i.e., the height information tells us about the orientation of the person's head with

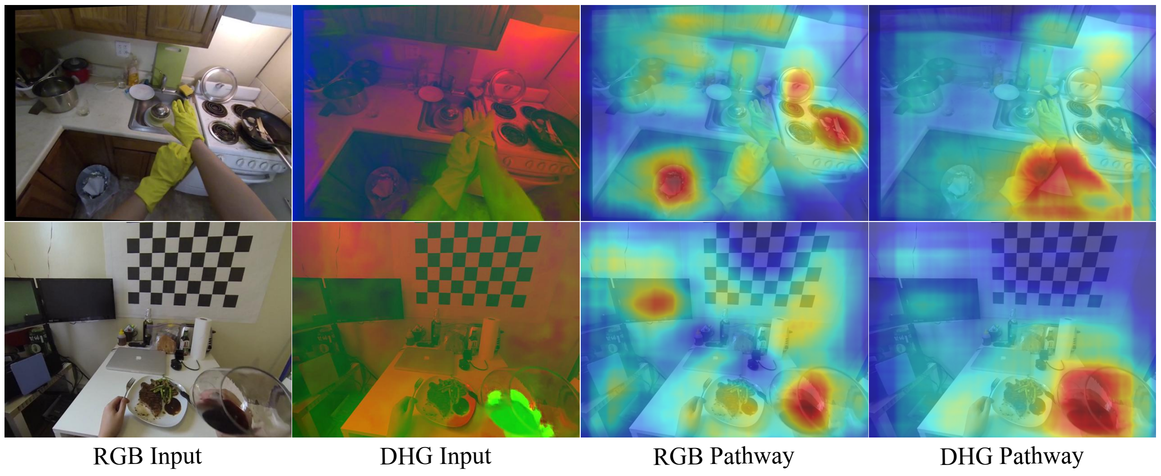


Figure 42: The visualization of the *fc7* activation values from the RGB and DHG pathways. The RGB pathway has higher activations around objects that stand out visually (e.g. a TV, a frying pan, a trash bin), while the DHG pathway detects objects that are at a certain distance and orientation relative to the person (e.g. a wine glass, the gloves).

respect to 3D environment. In addition, the gray scale image is used to capture basic visual appearance cues. Note that we do not use full RGB channels so that the DHG pathway would focus more on depth and height cues than the visual appearance cues. This diversifies the information learned by the two pathways, which allows EgoNet to learn complementary action-object cues. In Figure 42, we visualize the activation values from the *fc7* layer of RGB and DHG pathway averaged across all channels. Note that the two pathways learn to detect complementary action-object cues. Whereas RGB pathway detects objects that are visually prominent, DHG pathway has high activation values around the objects with a certain distance and orientation relative to the person.

1.4.3. Joint Pathway

To combine the information from RGB and DHG pathways for action-object prediction we introduce a joint pathway. The joint pathway first concatenates $39 \times 39 \times 4096$ dimensional *fc7* features from both RGB and DHG pathways to obtain a $39 \times 39 \times 8192$ dimensional tensor. This concatenated *fc7* tensor is then also concatenated with the downsampled $X \in \mathbb{R}^{39\times39}$ and $Y \in \mathbb{R}^{39\times39}$ first-person coordinates, which are obtained by generating X, Y

coordinate mesh-grids associated with every pixel in the original first-person image, and then downsampling these mesh-grids by a factor of 8, which is how much the resolution of the output is reduced inside the FCN. Then, the input to the joint pathway is a $39 \times 39 \times 8194$ dimensional tensor.

Afterwards, we perform a first-person coordinate embedding, which consists of (1) using the first-person spatial coordinates in a standard convolution operation, and then (2) attaching another convolutional layer to blend the visual and spatial information in the new layer. The first-person coordinate embedding allows EgoNet to use visual and spatial features in conjunction, which we show is beneficial for an accurate action-object detection in first-person images.

Intuitively the importance of first-person coordinate embedding can be explained as follows. The way a person positions himself during an interaction relative to an action-object, affects a location where such an action-object will be mapped in a first-person image. For instance, a laptop keyboard is often seen at the bottom of a first-person image because we often look down at it while typing with our hands. Our proposed first-person coordinate embedding allows EgoNet to learn such an action-object spatial distribution, which is different than most prior work that assumes a universal object spatial prior (e.g. a center prior) [152, 172].

One may think that our proposed first-person coordinate embedding should have a minimal effect to the network's performance because traditional FCNs also incorporate certain amount of spatial information in its prediction mechanism. However, unlike traditional FCNs, EgoNet uses first-person coordinates as features directly in the 2D convolution operation, which forces EgoNet to produce a different convolutional output than traditional FCNs would. Despite the simplicity of our first-person coordinate embedding scheme, in our experiments, we show that it significantly boosts the action-object detection accuracy.

1.4.4. Implementation Details

We implement EgoNet using Caffe [16]. The RGB and DHG pathways are built upon DeepLab architecture [29] The entire EgoNet network is trained jointly for 3000 iterations, at the learning rate of 10^{-6}, momentum of 0.9, weight decay of 0.0005, batch size of 15, and the dropout rate of 0.5. To optimize the network, we used a per-pixel softmax loss with respect to the action-object ground truth.

1.5. Experimental Results

In this section, we present quantitative and qualitative results of our EgoNet method on (1) our collected first-person action-object RGBD, (2) GTEA Gaze+ [152], (3) Social Children Interaction [25]. Additionally, to gain a deeper insight into an action-object detection problem we also include an action-object human study where 5 human subjects perform this task on our dataset.

To evaluate an action-object detection accuracy we use maximum F-score (MF), and average precision (AP) evaluation metrics, which are obtained by thresholding probabilistic action-object maps at small intervals and computing a precision and recall curve. Our evaluations provide evidence for four main conclusions:

- Our human-study indicates that humans achieve better action-object detection accuracy than the machines.
- We also demonstrate that our EgoNet model outperforms all other approaches by a considerable margin on our first-person action-object RGBD dataset.
- Furthermore, we empirically justify the design choices for our EgoNet model.
- Finally, we show that EgoNet performs well on the other novel first-person datasets.

	cooking		dining		grocery		hotel		desk work		shopping		dishwashing		mean	
	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP
DeepLab-Obj [29]	0.091	0.033	0.181	0.084	0.052	0.020	0.158	0.054	0.225	0.105	0.110	0.044	0.077	0.021	0.128	0.051
Judd [179]	0.188	0.091	0.160	0.092	0.048	0.021	0.286	0.189	0.523	0.428	0.102	0.051	0.063	0.030	0.182	0.107
GBVS [180]	0.213	0.113	0.188	0.097	0.043	0.014	0.216	0.128	0.499	0.487	0.098	0.047	0.124	0.063	0.197	0.136
Handled+Viewed-Obj	0.243	0.119	0.436	0.357	0.115	0.034	0.122	0.018	0.197	0.062	0.269	0.175	0.140	0.069	0.217	0.119
DeepLab-RGB [29]	0.342	0.220	0.220	0.143	0.134	0.063	0.146	0.065	0.292	0.213	0.158	0.073	0.262	0.128	0.222	0.129
AOP	0.366	0.264	0.195	0.084	0.180	0.086	**0.394**	0.222	0.421	0.327	0.137	0.074	0.267	0.178	0.280	0.176
FP-MCG [181]	0.224	0.113	0.400	0.283	**0.243**	**0.126**	0.274	0.136	0.597	0.389	0.200	0.093	0.281	0.170	0.317	0.187
DeepLab-DHG [29]	0.330	0.230	0.525	0.246	0.208	0.117	0.267	0.159	0.340	0.264	0.301	0.102	0.290	0.154	0.323	0.181
salObj+D [182]	0.306	0.182	**0.551**	0.451	0.188	0.105	0.361	**0.238**	0.501	0.378	**0.404**	**0.284**	0.257	0.184	0.367	0.260
EgoNet	**0.482**	**0.415**	0.509	**0.473**	0.193	0.121	0.298	0.183	**0.643**	**0.597**	0.242	0.134	**0.406**	**0.272**	**0.396**	**0.313**
Subject 1	0.419	-	0.576	-	0.408	-	0.477	-	0.556	-	0.357	-	0.415	-	0.458	-
Subject 2	0.453	-	0.551	-	0.327	-	0.444	-	0.516	-	0.518	-	0.436	-	0.464	-
Subject 3	0.453	-	0.604	-	0.197	-	0.453	-	0.581	-	0.340	-	0.439	-	0.438	-
Subject 4	0.506	-	0.631	-	0.358	-	0.489	-	0.579	-	0.371	-	0.407	-	0.477	-
Subject 5	0.488	-	0.660	-	0.227	-	0.444	-	0.598	-	0.373	-	0.435	-	0.461	-

Table 25: The quantitative results for action-object detection task on our first-person action-object RGBD dataset according to max F-score (MF) and average precision (AP) metrics. All methods except GBVS were trained on our dataset using a leave-one-out cross validation. The result indicates that EgoNet has the strongest predictive power with at least 2.9% (MF) and 5.3% (AP) gain over other methods. We also include our human study results, which suggest that human subjects achieve better action-object detection accuracy than the machines across most activities from our dataset.

1.5.1. Action-Object Human Study

To gain a deeper insight into an action-object detection task, we conduct a human study experiment to see how well humans can detect action-objects from first-person images. We randomly select 100 different first-person images from each of 7 activities from our first-person action-object RGBD dataset, and ask human subjects to identify a location of each action-object in such a first-person image.

We use 100 different images from each activity to keep the experiment's duration under an hour. Also, instead of collecting per-pixel or bounding box labels, we ask the subjects to identify action-objects by clicking at the center of an action-object, which is very efficient. We collect experimental action-object detection data from 5 subjects.

To obtain full action-object segmentations from the points selected by the subjects, we place a Gaussian with a width of 60 around the location of human selected point and project it on MCG [181] regions as is done in [183]. We acknowledge that due to the use of Gaussian and the errors in the MCG algorithm, our scheme of obtaining per-pixel segmentations out of a single point may slightly degrade human subject results . However, even under

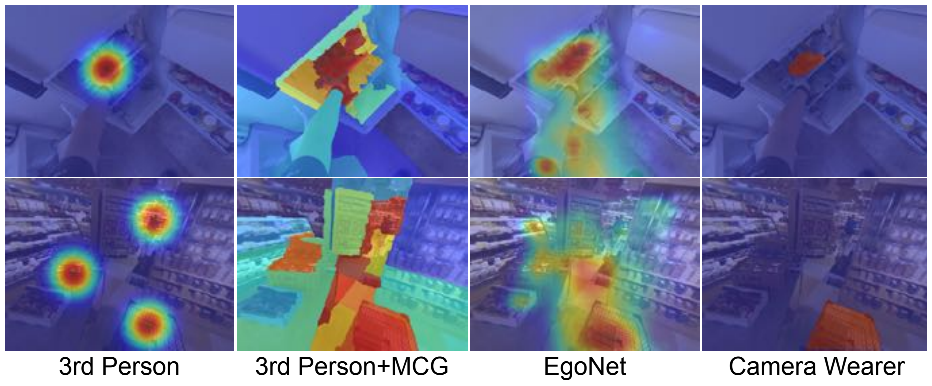


Figure 43: Qualitative human study results averaged across 5 human subjects. In many cases, third-person human subjects detect action-objects correctly and consistently. However, some activities such as shopping makes this task difficult even for a human observer since he does not know what the camera wearer was thinking.

current conditions a single experiment took about an hour or even more to complete. Thus, we believe that our chosen experiment conditions provided the best trade-off between the experiment duration and the quality of the action-object detection results provided by the subjects.

In the bottom of Table 25, we provide human subject results according to the MF evaluation metric. Unlike in our other experiments, the action-object masks obtained by the human subjects were skewed towards the extreme probability values of 1 and 0, which made the AP metric less informative. Thus, we only used MF score in this case.

Based on these results, we observe that in most cases, each of the 5 subjects perform better than the machines. We also observe that the action-object detection results achieved by the human subjects are quite consistent across most of different activities from our dataset. This indicates that humans can perform action-object detection from first-person images pretty effectively, despite not knowing exactly what the camera wearer was thinking (but possibly predicting it).

In Figure 43, we also present some qualitative human study results, where we average the predictions across all 5 human subjects. These results indicate that in some instances (second row), detecting action-objects is pretty difficult even for the human subjects since they do not know what the camera wearer was thinking.

1.5.2. Results on Our Action-Object RGBD Dataset

In Table 25 we present the results on our first-person action-object dataset, which contains 4247 annotated images. All the results are evaluated according to the MF and AP metrics. We include the following baseline methods in our comparisons: (1-2) GBVS [180] and Judd [179]: two bottom-up visual saliency methods; (3) FP-MCG [181]: a multiscale object segmentation proposal method that was trained on our first-person dataset; (4) Handled+Viewed Object: our trained method that detects objects around hands, if no hands are detected it predicts objects near the center of an image, which is where a person may typically look at; (5) Action-Object Prior (AOP): the average action-object location mask obtained from our dataset; (6) DeepLab-Obj [29]: a DeepLab network trained for traditional object-segmentation on our dataset with 41 object classes; (7-8) DeepLab-RGB [29] and DeepLab-DHG [29]: a DeepLab network trained for action-object detection using RGB and DHG images as its inputs respectively; (9) salObj+depth [182]: a salient object detection system, which we adapt to also handle a depth input.

Note that all methods except for GBVS are trained on our dataset. The training is done using a leave-one-out cross validation as is standard. Based on the results in Table 25, we observe that EgoNet outperforms all baseline methods by at least 2.9% (MF) and 5.3% (AP). In Figure 44, we also show our qualitative action-object results. Unlike other methods, EgoNet correctly detects and localizes action-objects in all cases.

1.5.3. Analysis of EgoNet Architecture

In this section, we quantitatively characterize the design factors of our EgoNet architecture on our dataset.

Are Separate RGB and DHG Pathways Necessary? An intuitive alternative to our EgoNet architecture is a single-stream network that concatenates RGB, DHG, and first-

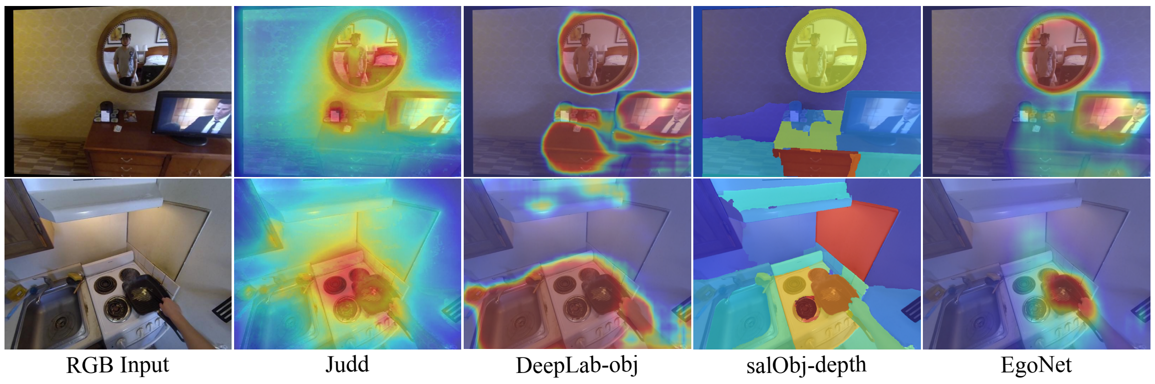


Figure 44: An illustration of qualitative results on our dataset (the mirror and the fry pan are the action-objects). Unlike other methods, our EgoNet model correctly recognizes and localizes action-objects in both instances.

person coordinate inputs and feeds them through the network. We test such a baseline and report that it yields 0.249 and 0.166 MF and AP scores, which is significantly worse than 0.396 (MF) and 0.313 (AP) attained by our EgoNet model.

What is the Contribution of RGB and DHG Pathways? Our EgoNet model predicts action-objects based on the visual appearance and 3D spatial cues, which are learned in the separate RGB and DHG pathways. To examine how important each pathway is, we train two independent RGB and DHG single-stream networks (both with the first-person coordinate embedding). Whereas the RGB network obtains 0.363 and 0.250 MF and AP scores, the DHG network achieves 0.369 and 0.208 MF and AP results. These results indicate that the 3D spatial cues are equally or even more informative than the RGB cues.

How Beneficial is the Joint Pathway? We note that combining RGB and DHG pathways via the joint pathway yields 0.396 and 0.313 according to MF and AP metrics, which is a substantial improvement over the independent RGB and DHG networks, which achieve 0.363 (MF), 0.250 (AP), and 0.369 (MF), 0.208 (AP) scores respectively. This suggests, that RGB and DHG pathways learn complementary action-object information, and combining them via the joint pathway is beneficial.

Comparison with a Deeper Single Stream RGB Model. We also include a single-

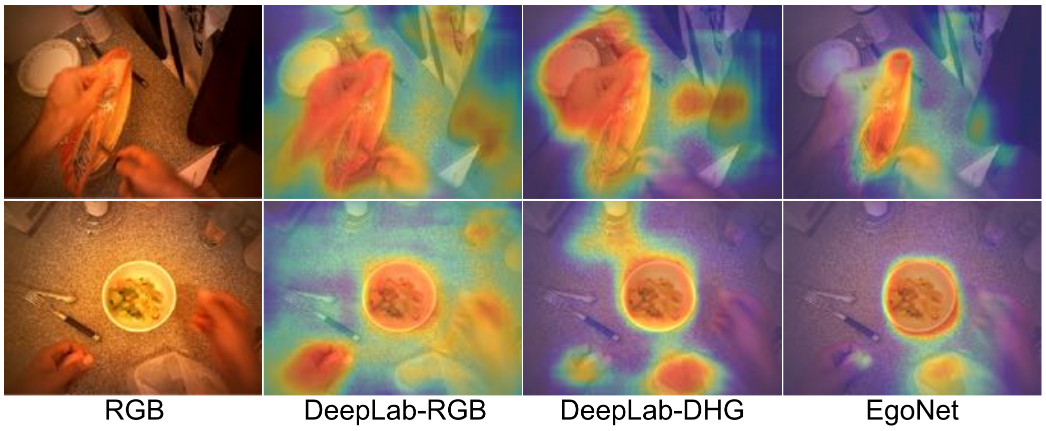


Figure 45: Qualitative results on GTEA Gaze+ dataset. EgoNet predicts action-objects more accurately and with better localization compared to DeepLab [29] based methods.

stream network baseline that only uses RGB information, but that has about $17M$ more parameters than the RGB pathway in our EgoNet architecture. We report that such a baseline achieves 0.363 (MF) score, which is identical to the RGB pathway in our EgoNet's design. Thus, these results indicate that simply adding more parameters to a single-stream network does not lead to better results.

Do First-Person Coordinates Help? Earlier we claimed that using first-person coordinates in the joint pathway is essential for a good action-object detection performance. To test this claim, we train a network with an identical architecture as EgoNet except that it **does not** use first-person coordinates. Such a network yields 0.333 and 0.232 MF and AP scores, which is considerably lower than 0.396 and 0.313 MF and AP results produced by our proposed EgoNet model, that uses first-person coordinates. Such a big accuracy difference between the two models suggests that first-person coordinates play a crucial role in an action-object detection task.

Is the Coordinate Embedding Useful? In the previous section, we also claimed that the first-person coordinate embedding (see Fig. 41) is crucial for a good action-object detection accuracy. To test this claim we train a network with an identical architecture as EgoNet except that we remove the last layer before softmax loss (i.e. where the coordinate

	breakfast		pizza		snack		salad		pasta		sandwich		burger		mean	
	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP	MF	AP
salObj+D [182]	0.192	0.113	0.233	0.147	0.303	0.225	0.191	0.112	0.202	0.117	0.346	0.248	0.211	0.108	0.240	0.153
DeepLab-RGB [29]	0.358	0.291	0.342	0.261	0.507	0.504	0.344	0.274	0.339	0.291	0.475	0.435	0.348	0.294	0.388	0.336
Judd [179]	0.310	0.223	0.318	0.214	0.522	0.482	0.404	0.329	0.393	0.315	0.452	0.374	0.403	0.304	0.400	0.320
FP-MCG [181]	0.364	0.281	0.456	0.380	0.486	0.399	0.401	0.310	0.411	0.346	0.511	0.454	0.321	0.207	0.422	0.340
AOP	0.403	0.307	**0.499**	**0.443**	0.488	0.380	0.337	0.211	0.408	0.273	0.472	0.375	0.369	0.268	0.425	0.322
DeepLab-DHG [29]	0.400	0.344	0.440	0.429	0.577	0.564	0.429	0.374	0.467	0.429	0.456	0.419	0.368	0.304	0.448	0.409
EgoNet	**0.433**	**0.357**	0.475	0.383	**0.607**	**0.568**	**0.512**	**0.450**	**0.532**	**0.505**	**0.576**	**0.486**	**0.454**	**0.358**	**0.513**	**0.443**

Table 26: Quantitative results on GTEA Gaze+ dataset for an action-object detection task. To test each model's generalization ability on GTEA Gaze+ dataset, we train each method **only** on our first-person action-object RGBD dataset. Based on the results, we observe that EgoNet exhibits the strongest generalization power.

embedding is performed). We observe that the network that **does not** use the coordinate embedding produces a 0.313 and 0.202 MF and AP scores, which is considerably lower than 0.396 and 0.313 achieved by our full EgoNet model.

1.5.4. Results on GTEA Gaze+ Dataset

To show strong EgoNet's generalization ability, in Table 26, we present our action-object detection results on the GTEA Gaze+ dataset [152], which consists of first-person videos that capture people cooking 7 different meals. The dataset provides the annotations of objects that people are interacting with during a cooking activity. In comparison to our first-person action-object RGBD dataset, GTEA Gaze+ contains many novel scenes, and many new object classes, which makes it a good dataset for testing EgoNet's generalization ability. Additionally, we note that, GTEA Gaze+ does not have depth information in the scene. Thus, we augment the dataset with depth predictions using [184].

Note that to test each model's generalization ability, all the methods are trained **only** on our first-person action-object RGBD dataset. Based on the results in Table 26, we can conclude that EgoNet shows the strongest generalization power: EgoNet achieves 0.513 (MF) and 0.443 (AP), whereas the second best method yields 0.448 (MF) and 0.409 (AP). We also include qualitative detection results from this dataset in Figure 45, which shows that our method detects action-objects more accurately and with much better localization than the DeepLab-FCN [29] baselines.

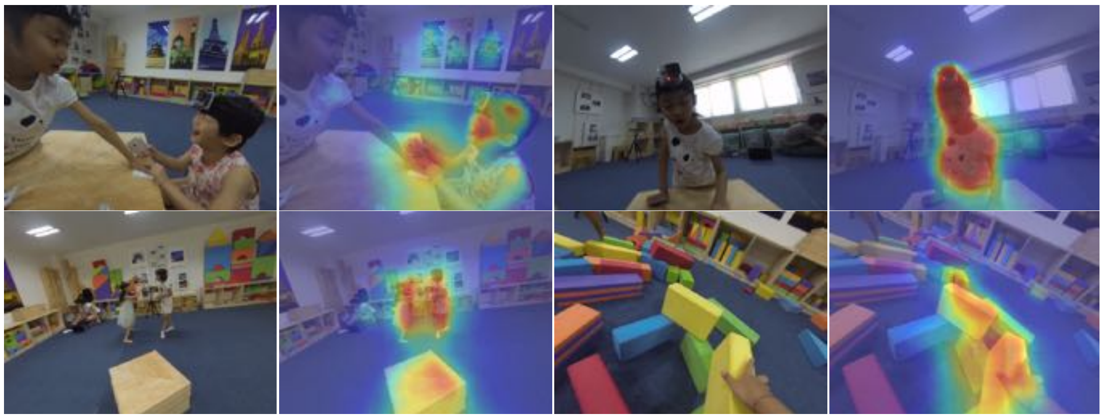


Figure 46: Our results on Social Children Interaction Dataset. Strong EgoNet's generalization power allows it to predict action-objects in a novel scenes, that contain previously unseen objects, and activities.

1.5.5. Results on Social Children Interaction Dataset

Furthermore, In Table 27, we present our results on Social Children Interaction dataset [25], which includes 9 first-person videos of three children playing a card game, building block towers, and playing hide-and-seek. The dataset consists of 2189 frames that are annotated with the location of children's attention location [25]. To evaluate all methods on this dataset, we place a fixed size Gaussian around the ground truth attention location, and use it as our ground truth mask to evaluate the results according to MF and AP metrics. Similar to GTEA Gaze+ dataset, this dataset only contains RGB images so we complement it with the depth predictions using the method in [184].

To test the generalization power, we train each method **only** on our first-person action-object RGBD dataset, and then test it on Social Children Interaction dataset. We report that EgoNet achieves 0.285 and 0.185 MF and AP, whereas the second best method yields 0.254 (MF) and 0.106 (AP) scores. We also illustrate several qualitative predictions in Figure 46.

	mean	
	MF	AP
DeepLab-RGB [29]	0.115	0.061
DeepLab-DHG [29]	0.123	0.063
FP-MCG [181]	0.141	0.064
AOP	0.254	0.106
EgoNet	**0.285**	**0.185**

Table 27: Quantitative results on Social Children Interaction dataset [25] for the visual attention prediction task. The dataset contains 9 first-person videos of children performing activities such as playing cards games, etc. We note that to test each method's generalization ability, none of the methods were trained on this dataset. We show that EgoNet outperforms all the other methods on this dataset, which indicates strong EgoNet's generalization power.

1.6. Conclusions

In this work, we use a concept of an action-object to study a person's visual attention and his motor actions from a first-person visual signal. To do this, we introduce EgoNet, a two-stream network that holistically integrates visual appearance, head direction, 3D spatial cues, and that also employs first-person coordinate embedding for an accurate action-object detection from first-person RGBD data. Our EgoNet leverages common person-object spatial configurations, which allows it to predict action-objects without an explicit adaptation to a specific task as is done in prior work [152, 153, 154]. We believe that EgoNet's predictive power and its strong generalization ability makes it well suited for the applications, such as personalized video dialog creation, video summarization, dexterous hand skill acquisition, and empirical understanding of visual sensorimotor systems of humans.

CHAPTER 2 : Unsupervised Learning of Important Objects from First-Person Videos

2.1. Introduction

A question "what is where?" attempts to delineate a picture as a spatial arrangement of objects rather than a collection of unordered visual words, which inspires core computer vision tasks such as recognition, segmentation, and 3D reconstruction. This spatial arrangement encodes not only the physical relationship between objects in front of the camera but also the interactions with the photographer standing behind the camera. A picture is always taken by a photographer reflecting what is important to her/him, which provides a strong cue to infer the internal states such as his/her intent, attention, and emotion. In particular, first-person videos capture unscripted interactions with scenes suggesting that the spatial layout is arranged such that the objects can afford the associated actions, e.g., a cup appears to be held by right hand from the holder's point of view.

In this paper, we aim to detect objects that are important to the photographer from a first-person video. Since importance is a subjective matter, the photographer is the only one who can identify an important object. However, we conjecture that it is possible to detect important objects without the supervision by the photographer or even third-person labelers because an important object exhibits common visual semantics (what it looks like) and a spatial layout (where it is in the first-person image).

To achieve this goal, we formulate an important object detection task as an interaction between the 1) segmentation and 2) recognition agents. Initially, the segmentation agent generates a candidate important object mask for each image, and relays this mask to the recognition agent, which then tries to learn a classifier to predict such an important object mask using visual semantics and spatial cues.

Our segmentation agent is implemented using an MCG projection scheme, which employs

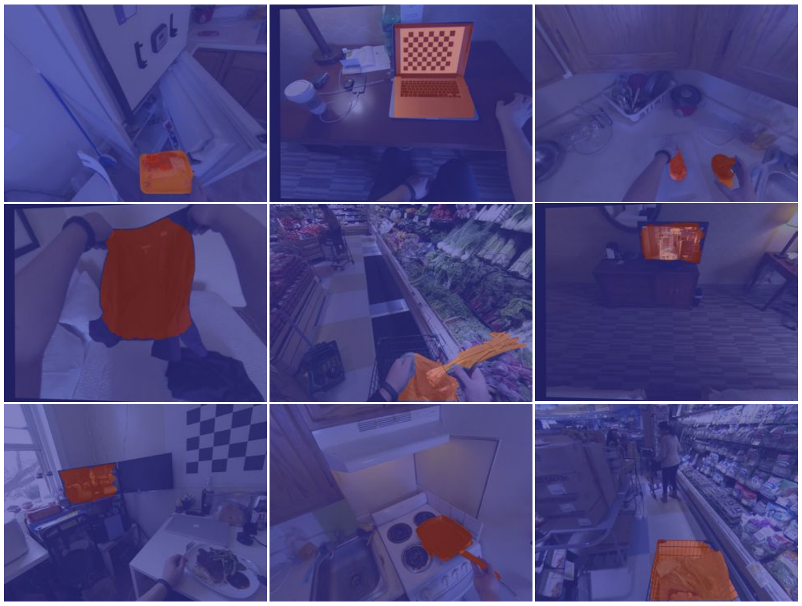


Figure 47: Given an **unlabeled** set of first-person images our goal is to find all objects that are important to the camera wearer. Unlike most prior methods, we do so without using ground truth importance labels.

the samples generated from an unsupervised segmentation method [181] to propose important object segmentation masks to the recognition agent. Our recognition agent is implemented using the visual (“what”) and spatial (“where”) pathways of our proposed Visual-Spatial Network (VSN), each of which learns to predict important object masks by asking questions “what an important object looks like?” and “where an important object is in the first-person image?”. We design these pathways using a fully convolutional network (FCN) while also embedding a location dependent layer in the spatial pathway to learn the first-person spatial location prior.

Our VSN then learns to detect important objects without using manually annotated importance labels. We do so via an alternating cross-pathway supervision, in a synergistic interplay between visual (“what”) and spatial (“where”) pathways, and a segmentation agent. Each pathway’s output is provided to a segmentation agent, which first generates a possible important object segmentation mask and then relays it to the other pathway to be used as a supervisory signal. The supervision proceeds in such an alternating fashion as each pathway improves each other, and as the segmentation agent becomes better as well.

Why Unsupervised Learning? Building a framework that can learn without manually collected labels is particularly essential for first-person important object detection because the annotation task is not scalable at all unlike object detection/segmentation [185, 186] where a consensus of third parties from crowdsourcing mechanism can be used. In the important object detection task, only the camera wearer can perform the annotation task by looking back on his/her past experiences. Prior methods [152] have used a wearable gaze tracker to label the camera wearer's visual attention. However, gaze tracker is invasive and the data that it captures has no notion of objects. Instead, our paper addresses these issues via an unsupervised alternating cross-pathway learning scheme, which allows our method to achieve similar or even better results as the supervised methods do.

2.2. Related Work

Important Object Detection in First-Person. There have been a number of first-person methods that explored important object detection task either as a main task [187, 153, 168], or as an auxiliary task for an activity recognition [169, 152, 154, 170] or video summarization [172, 173]. The work in [172, 168, 152, 169] employ hand-crafted appearance features, egocentric and optical flow features to describe a first-person image, and then train a discriminative classifier to detect the regions that correspond to the important objects. The more recent work [154, 187] use FCNs [56] to predict important objects end-to-end. Whereas the method in [187] employs a two stream visual appearance and 3D network, the work in [154] exploits the connection between the activities and objects and proposes a two stream appearance and optical flow network with a multi-loss objective function.

All of these methods use manually annotated important object labels, which may be costly and difficult to obtain. Our approach, on the other hand, introduces a new unsupervised learning scheme that allows us to learn important objects without manually labeled importance annotations.

Training FCNs with Weakly-Labeled Data. Recently, there have been several deep learning approaches that proposed learning with weakly labeled or unlabeled datasets [188, 73, 189, 190, 191, 183, 192, 193, 194] . Due to the high cost of obtaining per-pixel labels, this has been a particularly relevant problem for semantic segmentation.

The weakest form of supervision for semantic segmentation includes image-level labels, which were used to train FCNs in several prior approaches [183, 193, 194, 190]. Some recent work [189] used point supervision, which requires almost as much effort as the image-level labels but also provides some spatial information. Several approaches employed free form squiggles as a supervisory signal [191, 192] which provides even more information, and are still easy enough to annotate. Furthermore, several approaches utilized bounding box level annotations for FCN training [190, 73]. Finally, recent work achieved excellent edge detection results without using any annotations at all [188].

In comparison to prior work, which focuses on the third-person data, our method focuses on the first-person data. Unlike third-person object detection/segmentation tasks where annotations can be obtained via a crowdsourcing mechanism, important object detection task requires the camera wearer to provide the labels, which severely limits its scalability. Due to such a constraint, an unsupervised learning framework is particularly important for the important object detection task in the first-person setting.

2.3. Approach Motivation

Our goal is to 1) recognize and 2) segment important objects from a first-person image in an unsupervised setting. Thus, we want our method to have two key properties: 1) it needs to segment the important objects from the background based on the low-level grouping cues and 2) it needs to be discriminative, i.e, recognize objects that are important and ignore all the irrelevant objects.

To achieve these goals, we frame an important object prediction task as an interplay between the 1) recognition and 2) segmentation agents, where a segmentation agent first proposes a

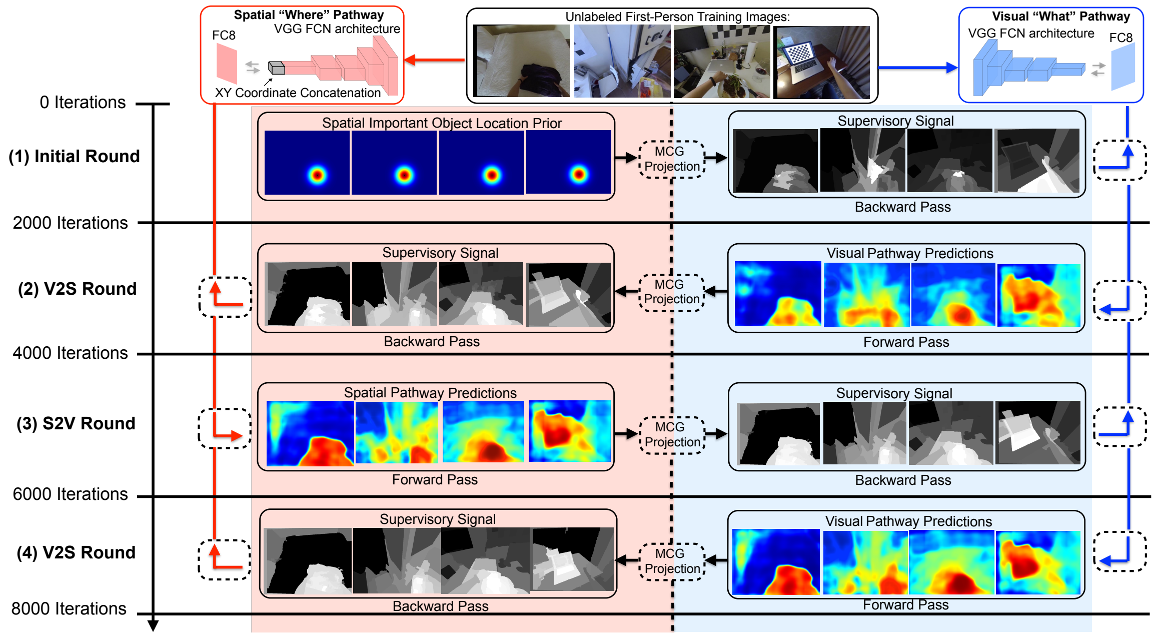


Figure 48: We implement an interplay between the segmentation and recognition agents via an alternating cross-pathway supervision scheme inside our proposed Visual-Spatial Network (VSN). Our VSN consists of the 1) visual ("what") and 2) spatial ("where") pathways, which both act as recognition agents. In between these two pathways, the VSN uses an MCG projection scheme, which acts as a segmentation agent. Then, given a set of **unlabeled** first-person training images, we first guess "where" an important object is in the first-person image and use an MCG projection scheme to propose important object segmentation masks. These masks are then used a supervisory signal to train a visual pathway such that it would learn "what" an important object looks like. Then, in the V2S round, the predictions from the visual pathway are passed through the MCG projection, and transfered to the spatial pathway. The spatial pathway then learns "where" an important object is in the first-person image. Such an alternating cross-pathway supervision scheme is repeated for several rounds.

possible important object mask, which a recognition agent then uses as a supervisory signal to learn an important object classifier based on visual ("what") and spatial ("where") cues.

The main challenge of our unsupervised learning framework is to prevent overfitting of either a segmentation or a recognition agent. If the segmentation agent proposes too many different segments, the recognition agent will not learn a concept of important objects (particularly if these segments are not recurring). On the other hand, if the recognition agent narrowly focuses on predicting one type of object, or an object that appears at a particular location, it will not generalize across all images. We address the first issue by feeding the predictions from the recognition agent to the segmentation agent, so that the target segmentations would consistently improve as the recognition agent gets better. To tackle the second issue, we force the recognition agent to learn a diverse model by making it focus on visual ("what") and spatial ("where") cues in an alternating fashion.

We now provide more details related to the 1) segmentation and 2) recognition agents that we want to use for our unsupervised learning task.

2.3.1. Segmentation Agent

The goal of a segmentation agent is to propose segmentation masks of the important objects, which could then be used by a recognition agent as a supervisory signal. We implement such a segmentation agent via our introduced MCG projection scheme. We define MCG projection as a function $h(A, R)$ that takes two inputs: 1) a coarse per-pixel important object mask prediction A, and 2) a set of regions R obtained from a segmentation method MCG [181]. The output $h(A, R)$ then captures an important object segmentation mask proposed by a segmentation agent.

We first run an MCG [181] segmentation algorithm, which segments a given image into regions R. Then, for every MCG region R, we compute the mean value of all values in A that fall in the region R, and assign that value to the entire **region** R. Since MCG regions overlap with each other, the pixels belonging to multiple overlapping regions, get assigned multiple values (from each region they belong to). To assign a single value to a given **pixel**, we perform max-pooling, over the values of that pixel in each of the regions that contains that **pixel**. This then produces a candidate important object segmentation mask.

2.3.2. Recognition Agent: Motivation

To build a recognition agent that is discriminative, and yet generalizable, we focus on two distinct aspects of an important object prediction task: the "what" (what does an important object look like?) and the "where" (where does an important object appear in the first-person image?).

The Visual Cues (What it looks like?) A natural way to predict important objects is by learning "what" they look like. Such learned visual appearance cues can then be used to predict important objects in an image. This is exactly what is done by the supervised methods, which use the ground-truth data to learn the visual characteristics of "what" a prototypical important object looks like in a first-person image. However, in the context of our problem, we do not have access to such ground-truth data. Thus, the key question becomes whether we can learn to detect important objects despite not knowing "what" they look like beforehand?

The Spatial Cues (Where it is?) We conjecture that important objects are spatially arranged in the first-person image to afford the camera wearer's interactions with those objects. In other words, by performing activities, and looking at things, the camera wearer is implicitly labeling what is important to him, which is also captured in a first-person image. For instance, a cup often appears at the *bottom right* of a first-person image, because most people look down at it and also hold it with their right hand.

Thus, since 1) people typically look down at an object, with which they interact, and 2) since most people are right-handed, we conjecture that many important objects appear at the *bottom-right* of a first-person image, which we guess to be at (x, y) location $(0.6W, 0.75H)$, where W and H denote the width and height of the first-person image. We refer to this location as a spatial important object location prior.

Since we do not have ground truth labels, we cannot directly supervise our network by telling it “what” an important object looks like. However, we can tell the network “where” we think an important object is such that the network would learn the visual appearance cues necessary to recognize “what” appears at that location. In the best case, there will be a true important object at our specified location, and the network will then learn “what” that important object looks like. Otherwise, if our guess is incorrect, the network will try to learn a meaningless pattern of “what” something that is **not** an important object looks like. If we make enough correct guesses of “where” the true important objects are, our network will learn “what” important objects look like without ever using ground truth importance labels.

2.4. Visual-Spatial Network

To holistically integrate both segmentation and recognition agents, we introduce a Visual-Spatial Network (VSN) that learns to detect important objects from unlabeled first-person data. Our network consists of the 1) visual (“what”) and 2) spatial (“where”) pathways, which act as recognition agents. In between these two pathways, the VSN employs an MCG projection scheme, which acts as a segmentation agent.

During training, we first use an MCG projection to propose a candidate important object segmentation mask, which is then used by the visual “what” pathway as a supervisory signal. Then, the predictions from the visual pathway are used by the segmentation agent to generate an improved important object segmentation mask, which is used as a supervisory signal by the spatial “where” pathway. Such a supervision scheme between the two pathways proceeds in an alternating fashion, allowing each pathway to improve each other, while the segmentation agent also improves. We refer to such a learning scheme as a cross-pathway supervision, which we illustrate in Fig. 48.

2.4.1. Visual "What" Pathway

The visual pathway of our VSN is based on a fully convolutional VGG architecture [60], which is pretrained for the segmentation task on Pascal VOC dataset with 20 distinct classes such as airplane, bus, cow, etc. We note that the classes in Pascal VOC dataset are quite different compared to the important object classes in the datasets that we use for our experiments. For instance, Pascal VOC segmentation dataset does not include annotations for classes such as food package, knife, suitcase, sweater, pizza and many more object classes. In the experimental section, we also verify this claim by showing that the VGG FCN [60] that was pretrained for the Pascal VOC semantic segmentation task alone produces poor important object detection results.

We want to make it clear that we do not claim that our method does not use any annotations at all. Our main claim is that we can learn to detect important objects in first-person images without manually annotated first-person importance labels. Our network still needs a general visual recognition capability to differentiate between various visual appearance cues. Otherwise, due to a noisy supervisory signals that we use to train each pathway, our network would struggle to learn the visual cues that are indicative of true important objects.

2.4.2. Spatial "Where" Pathway

The spatial pathway is also based on the pretrained VGG FCN [60]. However, unlike the visual pathway, the spatial pathway incorporates a two-channel grid of normalized X and Y coordinates that correspond to every pixel in the first-person image. These X, Y coordinate meshgrids could be obtained by calling a matlab command *[X,Y]=meshgrid(1:W,1:H)*, where W, H are the width and height of an image respectively. We then use a bilinear interpolation to downsample these grids 8 times and concatenate them to the visual *fc7* features. Such concatenated representation is then used as an input to the *fc8* layer that predicts important objects. Note that we do not concatenate X, Y grids with the input image so that we could preserve the original structure in the early layers of a VGG network, and use the VGG weights as an initialization.

2.4.3. Alternating Cross-Pathway Supervision

We now describe our alternating cross-pathway supervision scheme, which is implemented via a synergistic interplay between the spatial and the visual pathways, and with a segmentation agent in between these two pathways.

Initial Round. In the initial round, we want the visual pathway to predict important objects based on "what" they look like. It should learn to do so from the important object segmentation masks provided by an MCG projection step. These initial segmentation masks are constructed based on our guesses "where" important objects might appear in the first-person image.

Formally, we are given a batch of unlabeled first-person RGB images, which we denote as $B \in \mathbb{R}^{N \times C \times H \times W}$, where N depicts a batch size, H and W refer to the height and width of an image, and C refers to the number of channels ($C = 3$ for RGB images). Then, let $G \in \mathbb{R}^{N \times H \times W}$ denote images with a Gaussian placed around a spatial important object prior location $(0.6W, 0.75H)$.

Furthermore let h denote the MCG projection function that takes two inputs: 1) a coarse important object mask A, and 2) MCG regions R, and outputs a candidate important object segmentation mask $h(A, R)$.

Finally, let $f(B) \in \mathbb{R}^{N \times H \times W}$ depict the output of the visual pathway that takes a batch of first-person images as its input and outputs a per-pixel important objects map for every image in the batch. Then the cross-entropy loss that we minimize during the initial round is:

$$
\begin{aligned}
L = -\sum_{i=1}^{N}\sum_{j=1}^{H\times W} \Big[& h_j(G^{(i)}, R^{(i)}) \log\left(f_j(B^{(i)})\right) \\
& + (1 - h_j(G^{(i)}, R^{(i)})) \log\left(1 - f_j(B^{(i)})\right) \Big]
\end{aligned}
$$

V2S Round. During the V2S (Visual to Spatial) round, given the important object masks based on "what" they look like, we want spatial pathway to find image segments in the first-person image "where" such important objects appear.

Formally, let $g(B, X, Y) \in \mathbb{R}^{N\times H\times W}$ depict the output of the spatial pathway, where in this case X, Y denote a batch of normalized coordinate grids (each with dimensions $N \times H \times W$). Then the cross-entropy loss that we minimize during the V2S round is:

$$
\begin{aligned}
L = -\sum_{i=1}^{N}\sum_{j=1}^{H\times W} \Big[& h_j(f(B^{(i)}), R^{(i)}) \log\left(g_j(B^{(i)}, X^{(i)}, Y^{(i)})\right) \\
& + (1 - h_j(f(B^{(i)}), R^{(i)})) \log\left(1 - g_j(B^{(i)}, X^{(i)}, Y^{(i)})\right) \Big]
\end{aligned}
$$

S2V Round. In the S2V (Spatial to Visual) round, the visual pathway receives important object masks from the spatial pathway. Then, based on the spatial pathway's predictions "where" an important object is, the visual pathway tries to learn "what" those important objects look like. The cross-entropy loss function that we minimize during the S2V round is:

$$
\begin{aligned}
L = -\sum_{i=1}^{N}\sum_{j=1}^{H\times W} \Big[& h_j(g(B^{(i)}, X^{(i)}, Y^{(i)}), R^{(i)}) \log\left(f_j(B^{(i)})\right) \\
& + (1 - h_j(g(B^{(i)}, X^{(i)}, Y^{(i)}), R^{(i)})) \log\left(1 - f_j(B^{(i)})\right) \Big]
\end{aligned}
$$

Alternation. We alternate our cross-pathway supervision process between the V2S and S2V rounds until there is no significant change in performance (3-4 rounds). Such an alternating learning scheme is beneficial because different visual/spatial feature inputs to the two pathways, force each pathway to maintain focus on objects that exhibit different spatial/visual characteristics. For instance, the spatial pathway can focus on objects that are at the same spatial location, but exhibit different visual features. In contrast, the visual pathway is able to focus on the objects that look similar but are at different locations. Such an alternation between the two pathways provides diversity to our learning scheme, which we empirically show to be beneficial.

2.4.4. Using Extra Unlabeled Data for Training

We note that unlike supervised methods that use manually annotated importance labels, we use unlabeled data, which leads to a much harder learning task. We compensate the lack of importance labels with large amounts of unlabeled data, a strategy, which was also used by an unsupervised edge detector [188]. For all of our experiments, we train our VSN on the combined datasets of (1) first-person important object RGBD [187], (2) GTEA Gaze+ [152], and (3) five relevant first-person videos downloaded from YouTube (without using the labels even if they exist). We note that using more unlabeled data to train our model is essential for achieving the results that are competitive with the supervised methods' performance.

We point out that our method's ability to use unlabeled data for training is a big advantage in comparison to the supervised methods. The performance of CNNs typically improves with more training data, and unlabeled data is easy and cheap to obtain. In comparison, getting labeled data is costly and time consuming, especially if it requires per-pixel labels as in our work.

2.4.5. Prediction during Testing

During testing, we average the predictions from the visual and spatial pathways. Such a prediction scheme allows each pathway to correct some of the other pathway's mistakes, and achieve a better important object prediction accuracy than any individual pathway alone would.

2.4.6. Implementation Details

For all of our experiments, we used a Caffe deep learning library [16]. We employed visual and spatial pathways that adapted the VGG FCN architecture [60]. During training, each of the optimization rounds was set to 2000 iterations. During those rounds one of the selected pathways was optimized to minimize the per-pixel sigmoid cross entropy loss, while the other was fixed. We performed 3 rounds in total, which was enough to reach convergence. During the training we used a learning rate of 10^{-7}, the momentum equal to 0.9, the weight decay of 0.0005, and the batch size of 15.

2.5. Experimental Results

In this section, we present quantitative and qualitative results of our VSN method. We test our method on two first-person datasets, that have per pixel important object annotations: (1) First-Person Important Object RGBD [187], and (2) GTEA Gaze+ [152] datasets. Even though both datasets have annotated importance labels, they are quite different. GTEA Gaze+ dataset captures the activities of cooking different meals, and thus there is less variation in the scene and the activity itself. In comparison, the first-person important object RGBD dataset is smaller but captures people doing seven different activities in pretty different scenes, which makes the dataset more diverse and slightly more challenging. The First-Person Important Object RGBD dataset has 4247 annotated examples from seven video sequences, whereas for the GTEA Gaze+ dataset we use 6332 images from 22 different sequences.

We evaluate the important object detection accuracy using max F-score (MF), and average precision (AP), which are obtained by thresholding the probabilistic important object maps at small intervals and computing a precision and recall curve against the ground-truth important objects.

	FP-AO-RGBD		GTEA Gaze+		mean	
Method	*MF*	*AP*	*MF*	*AP*	*MF*	*AP*
VGG FCN [60]	0.166	0.106	0.325	0.214	0.246	0.160
GBVS [180]	0.197	0.136	0.383	0.296	0.290	0.216
Judd [179]	0.182	0.107	0.406	0.328	0.294	0.218
DCL [195]	0.255	0.068	0.427	0.120	0.341	0.094
SIOLP	0.278	0.148	0.416	0.209	0.347	0.179
Trained SIOLP‡	0.282	0.176	0.446	0.351	0.364	0.264
FP-MCG [181]‡	0.317	0.187	0.447	0.361	0.382	0.274
DeepLab [29]‡	0.370	0.266	0.472	0.390	0.421	0.328
EgoNet [187]‡	0.396	0.313	**0.536**	0.449	**0.466**	0.381
VSN	**0.421**	**0.316**	0.482	**0.472**	0.452	**0.394**
VSN+EgoNet‡	**0.455**	**0.382**	**0.588**	**0.604**	**0.522**	**0.493**

Table 28: The quantitative important object prediction results on the first-person important object RGBD and GTEA Gaze+ datasets according to the max F-score (MF) and average precision (AP) metrics. Our results indicate that even without using important object labels our VSN achieves similar or even better results than the supervised baselines. Supervised methods are marked with ‡.

As our baselines we use a collection of the methods that were recently shown to perform well on this task as well as some of our own baselines. EgoNet [187] is a two-stream network that incorporates appearance and $3D$ cues to detect important objects. We also include a DeepLab [29] system, which we train for the important object detection task. Additionally, we incorporate a MCG [181] method trained for first-person important object detection (FP-MCG). Furthermore, we include three popular visual saliency methods: (1) Judd [179], (2) GBVS [180], and (3) Deep Contrast Saliency method [195]. Additionally, we also evaluate the results achieved by (1) a spatial important object location prior (SIOLP), and (2) a spatial important object location prior that was obtained by extracting it from the training data using ground-truth important object labels. Furthermore, to show that the network that we used to pretrain our VSN performs poorly by itself, we include a VGG FCN [60] baseline. To obtain important object predictions we simply sum up the probabilities for all 20 predicted Pascal VOC classes. Finally, to show that the predictions of our VSN method are highly complementary to the best performing EgoNet method's predictions, we combine these two methods via averaging, and demonstrate that for each dataset VSN significantly improves EgoNet's results.

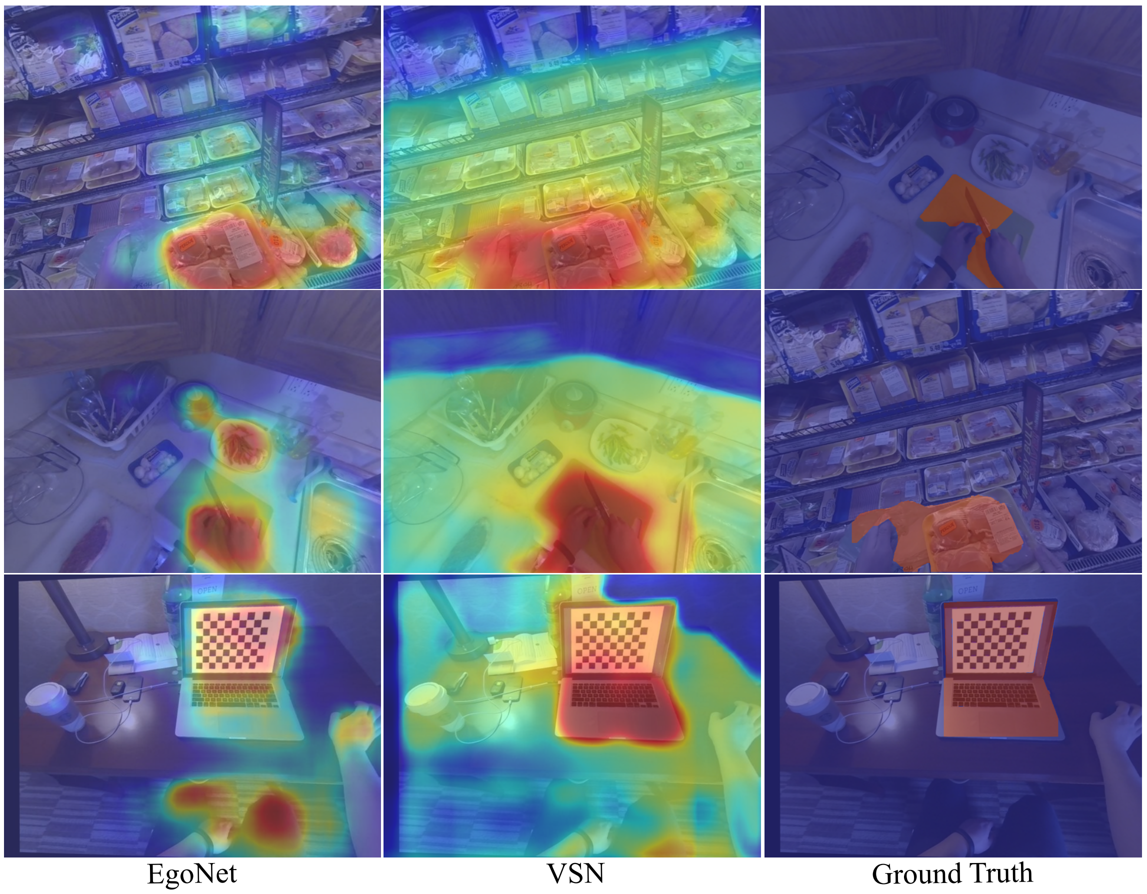


Figure 49: The qualitative important object predictions results. Despite not using any importance labels during training, our VSN correctly recognizes and localizes important objects in all three cases.

We also note that Judd [179], GBVS [180], DCL [195], SIOLP, VGG-FCN, and our VSN methods do not use any important object annotations. All the other methods are trained using the manually annotated important object labels. We also note that all the FCN baselines (VGG-FCN, DeepLab, EgoNet and VSN) were pretrained for semantic segmentation under the same conditions.

We used publicly available implementations of VGG-FCN, GBVS, Judd, FP-MCG [181], and DeepLab [29] and trained and evaluated all these baselines ourselves. We obtained the results for EgoNet from the technical report in [187]. To the best of our knowledge EgoNet is currently the best performing method in this task, and thus, to compare to the most recent and best performing system, we adopted the evaluation procedure from [187].

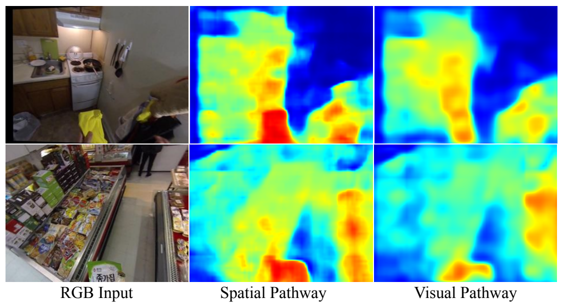
Our evaluations provide evidence for several conclusions. In Subsections 2.5.1, 2.5.2, we show that despite **not** using any important object labels our VSN achieves results similar or even better than the supervised methods do. Furthermore, In Subsection 2.5.3, we provide a few ablation experiments, which show that 1) using both visual and spatial pathways is beneficial, 2) the location of an important object spatial prior is important, and that 3) using more unlabeled training data leads to better results.

2.5.1. Results on the Important Object RGBD Dataset

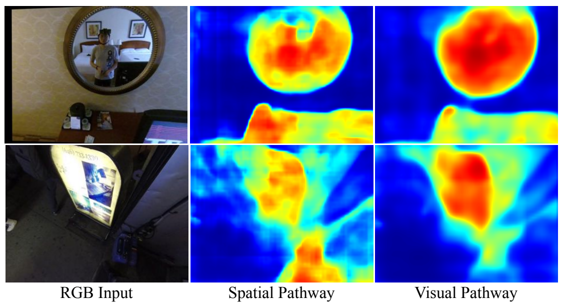
In Table 28, we present important object detection results on the First-Person Important Object RGBD dataset [187], averaged over 7 video sequences from different activities. The results indicate that our VSN achieves the best per-class mean MF and AP scores. These results may seem surprising, because unlike EgoNet and all the other supervised baselines, our VSN does not use any important object annotations. However, VSN uses a larger amount of **unlabeled** data for its training, which leads to a better performance.

We also note that the VGG-FCN, which we use as an initialization for both of our VSN pathways, achieves the worst performance among all the baselines, which suggests that predicting 20 Pascal VOC classes alone is not enough to achieve a good performance on the important object detection task. We also point out that combining VSN and EgoNet predictions, leads to a greatly improved accuracy according to both metrics, which implies that both methods learn complementary important object information.

In Figure 49, we also compare qualitative important object detection results of our VSN and a supervised EgoNet model. We show that unlike EgoNet, our VSN correctly detects and segments important objects in all three cases.



(a) Spatial pathway performing better than the visual pathway



(b) Visual pathway performing better than the spatial pathway

Figure 50: A figure illustrating a qualitative important object prediction comparison between the visual and spatial pathways (best viewed in color). Subfigure on the left illustrates instances where the spatial pathway's reliance on location features is beneficial: it detects small and partially occluded important objects, which the visual pathway fails to detect accurately. The Subfigure on the right shows instances where the spatial pathway's reliance on location features leads to incorrect results: it falsely marks regions in the first-person image as important objects just because they appear at a certain location in the first-person image. In contrast, the visual pathway correctly predicts important objects in those instances.

2.5.2. Results on the GTEA Gaze+ Dataset

In Table 28, we present MF and AP important object detection results on the GTEA Gaze+ dataset [152] averaged over 22 videos. The results indicate that our VSN outperforms all the other methods according to AP metric, and is outperformed only by EgoNet according to the MF metric. We also note that just like with the previous dataset, combining VSN and EgoNet predictions leads to a dramatic accuracy boost according to both metrics.

2.5.3. Ablation Experiments

The Need for Spatial and Visual Pathways. One may notice that the spatial pathway is a more powerful version of a visual pathway since it can use both spatial and visual cues to predict important objects. Therefore, a natural question is whether we need a visual pathway at all.

To answer this question we quantitatively compare our approach to the baselines that use either two visual pathways (VVN) or two spatial pathways (SSN). We present these results

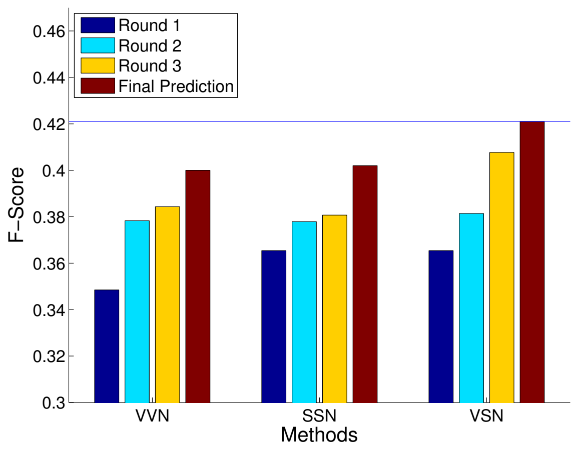


Figure 51: Our results demonstrate that using a visual and a spatial pathway (VSN) yields better important object detection accuracy than using either two visual (VVN) or two spatial pathways (SSN).

in Figure 51, where we show that our VSN method achieves 0.421 MF accuracy,whereas the VVN and SSN baselines yield 0.402 and 0.400 MF accuracy respectively, suggesting that having a visual and a spatial pathway in the network is beneficial.

In Figure 50, we also present a few qualitative comparisons between the predictions from the visual and spatial pathways. In Subfigure 3.55(c), we illustrate instances where the spatial pathway's reliance on location features is beneficial: unlike the visual pathway, it is able to detect small and partially occluded important objects because they appear at a certain location. However, in Subfigure 3.55(d), we present instances where the spatial pathway's reliance on location features leads to incorrect results: it falsely marks regions in the first-person image as important objects just because those regions appear at a certain location in an image. In contrast, in those cases, the visual pathway correctly predicts important objects because it makes the predictions based on "what" those objects look like.

Thus, these qualitative and quantitative results suggest that the spatial and visual pathways can complement each other, and thus, having both of them is beneficial.

Selecting a Spatial Important Object Prior. The initial selection of a location prior is critical to the success of our method. To validate its importance, we run several experiments on important object RGBD dataset with different location priors. First, we experiment with a location $(0.5W, 0.5H)$ where H and W are the height and the width of an image. This is a center prior commonly used by first-person methods. Using this location prior, the pathway that was trained the last yields 0.22 MF score, suggesting that in this case, a center location does not capture important objects well. In comparison, the pathway that was trained the last in our original model (i.e. the location prior $(0.6W, 0.75H)$) yields 0.407 MF.

Given how important the selection of a location prior is, we need a principled way to select it. To do this we propose to utilize a generic hand detector or an unsupervised visual saliency detector on our dataset (**not** trained on our dataset). Then, for each image we can compute a weighted average of XY locations in the image (weighted by the hand detector probabilities), and then compute an average of these weighted average locations across the entire dataset.

We report that when applied on an important object RGBD dataset, such a scheme yields a location prior of $(0.542W, 0.713H)$. Training the VSN using this spatial prior then yields almost equivalent results as our original model that uses $(0.6W, 0.75H)$ location prior. We also note that simply detecting hands is not enough to detect important objects. A baseline that detects all objects and hands in the scene, and then uses objects that are closest to the hands as important object predictions yields 0.259 MF.

Importance of Unlabeled Training Data Size. Finally, we also want to verify that using more unlabeled training data leads to better results. To do this, we train VSN on the same amount of **unlabeled** data, as there is labeled data used by the supervised methods (4247 samples). We report that using less unlabeled data leads to 0.316 MF, which is substantially lower than our original model.

2.6. Conclusions

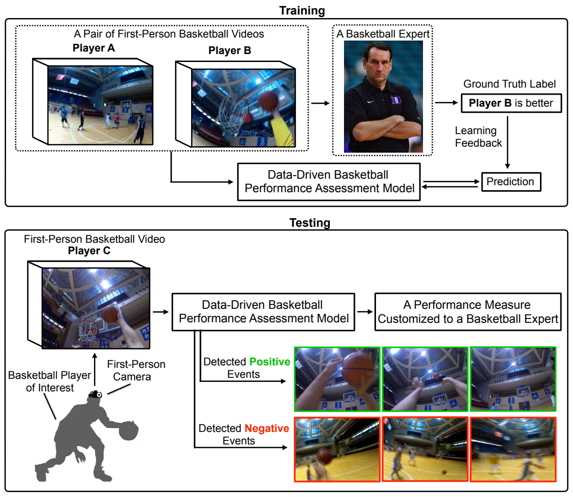
In this work, we propose to detect important objects from unlabeled first-person images by formulating our problem as an interplay between the 1) recognition and 2) segmentation agents. To do this, we integrate these two agents inside an alternating cross-pathway supervision scheme of our proposed Visual-Spatial Network (VSN). The MCG projection scheme (a segmentation agent) proposes important object segmentation masks, whereas the spatial and visual pathways (recognition agents) use these masks as a supervisory signal to predict important object masks based on visual semantics and spatial features. We demonstrate the effectiveness of such scheme by showing that it achieves similar or even better results than the supervised methods.

We believe that in the future, our method could be extended to other tasks such as first-person activity recognition, or egocentric video summarization. Furthermore, our method's ability to learn without manually annotated labels could be used to learn from large-scale unlabeled first-person datasets on the web, and in the long run, replace the supervised methods, which are constrained by the amount of available annotated data.

CHAPTER 3 : Am I a Baller? Basketball Performance Assessment from First-Person Videos

3.1. Introduction

A gifted offensive college basketball player, Kris Jenkins (Villanova), made a three point buzzer beater against UNC (2015-2016 season), and recorded one of the greatest endings in NCAA championship history. He was arguably one of the best players in the entire NCAA tournament. A question is "what makes him stand out from his peer players?". His stats, e.g., average points and rebounds per game, can be a measure to evaluate his excellence. However, these measures do not capture every basketball aspect that a coach may want to use for assessing his potential impact in the future team, which is difficult to measure quantitatively. NBA coaches and scouts are eager to catch every nuance of a basketball player's abilities by watching a large number of his basketball videos.



Training
A Pair of First-Person Basketball Videos
Player A
Player B
A Basketball Expert
Ground Truth Label
Player B is better
Learning Feedback
Data-Driven Basketball Performance Assessment Model
Prediction
Testing
First-Person Basketball Video
Player C
Data-Driven Basketball Performance Assessment Model
A Performance Measure Customized to a Basketball Expert
Basketball Player of Interest
First-Person Camera
Detected Positive Events
Detected Negative Events

Figure 52: Our goal is to assess a basketball player's performance from unscripted first-person basketball videos. During training, we learn such a model from the pairs of weakly labeled first-person basketball videos. During testing, our model predicts a performance measure customized to a particular evaluator from a given video. Our model also discovers basketball events that contribute positively and negatively to a player's performance.

Now consider a college recruitment process where there is a massive number of high school players. In such conditions, the searching task for the best players becomes much more challenging, more expensive and also more labor intense. More importantly, the recruiters need to measure and evaluate a sequence of atomic decision makings, e.g., when does a player shoot, whether he makes a shot, how good is his passing ability, etc. There exists neither universal measure nor golden standard to do this, i.e., most scouts and coaches have their own subjective evaluation criterion.

In this paper, we address a problem of computational basketball player assessment customized to a coach's or scout's evaluation criterion. Our conjecture is that a first-person video captures a player's basketball actions and his/her basketball decision making in a form of the camera motion and visual semantics of the scene. A key challenge of first-person videos is that it immediately violates primary assumptions made for third-person recognition systems: first-person videos are highly unstable and jittery and visual semantics does not appear as iconic as in third-person [185].

Our first-person approach innovates the traditional assessment methods, e.g., watching hours of third-person videos taken by non professional videographers and assessing the players in them. In contrast, a first-person video records what the player sees, which directly tells us what is happening to the player himself, e.g., the body pose of a point guard who is about to pass at HD resolution while a third-person video produces a limited visual access to such subtle signals. Furthermore, the 3D camera egomotion of the first person video reflects the decision making of how the player responds to the team configuration, e.g., can I drive towards the basket and successfully finish a layup? Finally, a first-person camera eliminates the tracking and player association tasks of the third-person video analysis, which prevents applications of computational approaches for amateur games[1].

Our system takes a first-person video of basketball players and outputs a basketball assessment metric that is specific to an evaluator's preference. The evaluator provides the comparative weak labels of the performance of the players, e.g., the player A is better than B based on his own subjective criteria.

Our method first uses a convolutional LSTM to detect atomic basketball events from a first-person video. Our network's ability to localize the most informative regions in a first-person image, is essential for first-person videos where the camera undergoes severe head movement,

[1]Usage of commercial tracking systems using multiple calibrated cameras is limited due to a high cost [196]

which causes videos to be blurry. These atomic events are then passed through the Gaussian mixtures to produce a highly non-linear visual spatiotemporal basketball assessment feature. Finally, our basketball assessment model is learned from the pairs of labeled first-person basketball videos by minimizing a hinge loss function. We learn such a basketball skill assessment model from our new 10.3 hour long first-person basketball dataset that captures 48 distinct college level basketball players in an unscripted basketball game.

Impact Ample money and effort have been invested in recruiting, assessing, and drafting basketball players every year. However, limited progress has been made on developing computational models that can be used to automatically assess an athlete's performance in a particular sport [197, 198]. As wearable technology advances, cameras can be non-invasively worn by players, which delivers a vivid sense of their dynamics, e.g., Spanish Liga ACB has demonstrated a possibility of a jersey camera that allows you to put yourself in the court [199]. This trend will open up a new opportunity to share experiences and evaluate performance across players in different continents without bias and discrimination. Our work takes a first step towards enabling a computational analysis for such first-person data.

Contribution To the best of our knowledge, this is the first paper that addresses practical behavioral assessment tasks using first-person vision specific to an evaluator's preference. The core technical contributions of the paper include 1) a basketball assessment model that assesses the players based an an evaluator's assessment criterion, which we learn from the pairs of weakly labeled first-person basketball videos; 2) a predictive network that learns the visual semantics of important actions and localizes salient regions of first-person images to handle unstable first-person videos and 3) a new 10.3 hour long first-person basketball video dataset capturing 48 players in an unscripted basketball game.

3.2. Related Work

Talent wins games, but teamwork and intelligence wins championships. — Michael Jordan

Accurate diagnosis and evaluation of athletes is a key factor to build a synergic teamwork. However, it is highly subjective and task dependent, and the psychological and financial cost of such process is enormous. A large body of sport analytics and kinesiology has studied a computational approaches to provide a quantitative measure of the performance [197, 198, 200, 201, 202].

Kinematic abstraction (position, orientation, velocity, and trajectory) of the players offers a global centric representation of team behaviors, which allows a detailed analysis of the game such as the probability of shoot success, rebound, and future movement prediction [200, 201, 203]. Not only an individual performance, but also team performance can be measured through the kinematic abstraction [202, 198].

These kinematic data are often obtained by multiple third-person videos [196, 201, 198] where the players and ball are detected using recognition algorithms combined with multiple view geometry [204]. Tracking and data association is a key issue where the role of the players provides a strong cue to disambiguate appearance based tracking [205]. Events such as ball movement, can be also recognized using a spatiotemporal analysis [206]. As players behave strategically and collectively, their group movement can be predicted [207] and the ball can be localized without detection. Various computational models have been used for such tasks, e.g., Dynamic Bayesian Network [208], hierarchical LSTM [209], attention based LSTM [210] learned from a large collection of third-person videos.

Unlike third-person videos, first-person cameras closely capture what the players see. Such property is beneficial to understand activities highly correlated with visual attention, e.g., object manipulation and social communications. Important objects to the camera wearer are detected and segmented [172, 211, 153, 168, 187], which can be used to compress life-log videos [172, 173]. As visual attention is also related with the intent of the camera wearer, her/his future movement can be predicted [212]. Beyond individual behaviors, joint attention is a primary indicator of social interactions, which can be directly computed from first-person videos [213, 214], and further used for human-robot interactions [171, 167].

In sports, the complex interactions with a scene in first-person videos can be learned through spatiotemporal visual patterns. For instance, the scene can tell us about the activity [215] and the egomotion can tell us about the physical dynamics of activity [216]. Joint attention still exists in team sports which can be described by the team formation [25] and future behaviors [203].

Unlike previous work that mainly focuses on recognizing and tracking objects, activities, and joint attention, we take one step further: performance assessment based on the evaluator's preference. We introduce a computational model that exhibits strong predictive power when applied on the real world first-person basketball video data.

3.3. Basketball Performance Assessment Model

We define a measure of performance assessment using a first-person video:

$$S(\mathcal{V}) = \frac{\sum_{t=1}^{T} p_t^{(1)} \mathbf{w}^\mathsf{T} \boldsymbol{\phi}(\mathbf{V}_t, \mathbf{x})}{\sum_{t=1}^{T} p_t^{(1)}} \tag{3.1}$$

where $\mathcal{V}$ is a first-person video of T number of frames, ϕ is a visual spatiotemporal basketball assessment feature, and $\mathbf{w}$ is a weight vector of performance regressor. $\mathbf{V}_t \subset \mathcal{V}$ is a segmented video starting at the t^{th} frame with a fixed length, T_s. $p_t^{(1)} \in [0, 1]$ is a relevance of $\mathbf{V}_t$ to evaluate a given player's performance. $\mathbf{x} \in \mathbb{R}^2$ is the 2D coordinate of the basketball player, i.e., the projection of 3D camera pose computed by structure from motion [204] onto the canonical basketball court. In Figure 53, we provide a detailed illustration of our basketball assessment prediction framework.

3.3.1. Visual Spatiotemporal Assessment Feature

Our first goal is to use a first-person basketball video to build a powerful feature representation that could be used for an effective player's performance assessment. We identify three key challenges related to building such a representation from first-person basketball videos: 1) our system needs to handle severe camera wearer's head motion, 2) we need to have an interpretable basketball representation in terms of its atomic events, and 3) our feature representation has to be highly discriminative for a player's performance prediction task.

To address these problems, we propose to represent the visual feature of the segmented video, $\mathbf{V}_t$, as follows, where each function below addresses one of the listed challenges:

$$\boldsymbol{\phi}(\mathbf{V}_t, \mathbf{x}) = f_{\text{gm}}\left(f_{\text{event}}\left(f_{\text{crop}}\left(\mathbf{V}_t\right), \mathbf{x}\right)\right), \tag{3.2}$$

where f_{crop} is a function that handles a severe camera wearer's head motion by producing a cropped video by zooming in on the important regions, f_{event} is a function that computes the probability of atomic basketball events, and f_{gm} is a Gaussian mixture function that computes a highly non-linear visual feature of the video.

Zooming-In. A key property of f_{crop} is the ability to zoom-in to relevant pixels which allows to learn an effective visual representation for the basketball performance assessment. Using this regional cropping, we minimize the effect of jittery and unstable nature of first person videos that causes larger variation of visual data. In our experimental section, we demonstrate that using f_{crop} in our model substantially improves the prediction performance. Thus, initially we process a first-person video to produce a cropped video:

$$\overline{\mathbf{V}}_t = f_{\text{crop}}(\mathbf{V}_t; \mathbf{w}_{\text{crop}}),$$

where f_{crop} is parametrized by $\mathbf{w}_{\text{crop}}$, $\overline{\mathbf{V}}_t$ is the cropped video with fixed size $C_w \times C_w \times 3 \times T_s$, and C_w is the width and height of the cropping window.

We predict the center of the cropping window by learning $\mathbf{w}_{\text{crop}}$ using a fully convolutional network [88]. To do this, we train the network to predict the location of a ball, which is typically where most players are looking at. Afterwards, for each frame in a video, we compute a weighted average of XY location coordinates weighted by the detected ball probabilities and then crop a fixed size patch around such a weighted average location. We illustrate some of the qualitative zoom-in examples in Figure 57.

Atomic Basketball Event Detection. To build an interpretable representation in terms of atomic basketball events, we predict basketball events of 1) sombeody shooting a ball, 2) the camera wearer possessing the ball, and 3) a made shot respectively. Note that the cropped video focuses on the ball and its visual context, which allows to learn the visual semantics of each atomic event more effectively. To do this we use a multi-path convolutional LSTM network, where each pathway predicts its respective atomic basketball event. We note that such a multi-path architecture is beneficial as it allows each pathway to focus on learning a single atomic basketball concept. In contrast, we observed that training a similar network with a single pathway failed to produce accurate predictions for all three atomic events. Given a cropped video, our multi-path network is jointly trained to minimize the following cross-entropy loss:

$$\mathcal{L}_{\text{event}} = -\sum_{t=1}^{T_s}\sum_{b=1}^{3} y_t^{(b)} \log p_t^{(b)} + (1 - y_t^{(b)}) \log \left(1 - p_t^{(b)}\right),$$

where $p_t^{(b)}$ depicts a network's prediction for an atomic basketball event b at a time step t; $y_t^{(b)} \in \{0, 1\}$ is a binary atomic basketball event ground truth value for frame t and basketball event b.

We also note that because many important basketball events occur when somebody shoots the ball [217, 218], the detected probability $p_t^{(1)}$ is also later used in Equation (3.1), as a relevance indicator for each video segment, $\mathbf{V}_t$.

As our fourth atomic basketball event $p_t^{(4)}$, we use a binary value indicating whether a player is in the 2 point or 3 point zone, which is obtained from a player's (x, y) location coordinates on the court.

We then split each of the 4 basketball event predictions in half across the temporal dimension, and perform temporal max pooling for each of the 8 blocks. All the pooled values are then concatenated into a single vector $\mathbf{b}_t$:

$$\mathbf{b}_t = f_{\text{event}}(\overline{\mathbf{V}}_t, \mathbf{x}; \mathbf{w}_{\text{event}})$$

Gaussian Mixtures. To build a representation that is discriminative, and yet generalizable, we construct a highly non-linear feature that works well with a linear classifier. To achieve these goals we employ Gaussian mixtures, that transform the atomic basketball event feature, into a complex basketball assessment feature, which we will show to be very effective in our assessment model. Formally, given a vector $\mathbf{b}_t$ over T_s, we compute the visual spatiotemporal assessment features for a given video segment as:

$$\boldsymbol{\phi}_t = f_{\text{gm}}\left(\mathbf{b}_t; \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}\right)$$

where f_{gm} is parametrized by Gaussian mixtures, $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}$, and N is the number of mixtures. Each mixture j is defined by a function $z(y_{t_1}^{(1)}, y_{t_2}^{(1)}, \dots, y_{t_1}^{(4)}, y_{t_2}^{(4)}) = j$. Here $y_{t_1}^{(i)}, y_{t_2}^{(i)} \in \{0, 1\}$ refer to the binary ground truth values associated with an atomic basketball event $i \in \{1, 2, 3, 4\}$; the index t_1 indicates the first half of an input video segment, whereas t_2 indicates the second half. Every possible combination of these values define one of the $2^8 = 256$ Gaussian mixtures. We learn the parameters of each Gaussian mixture using

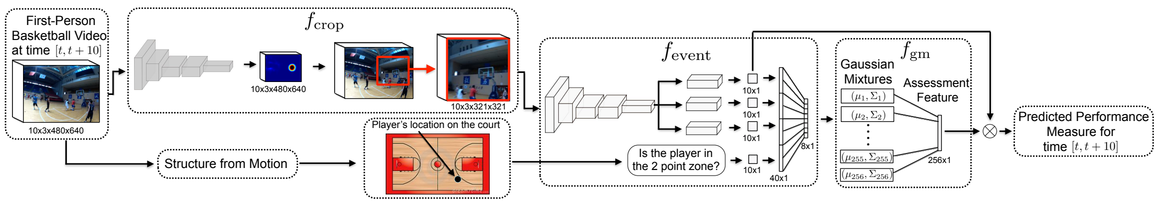

First-Person Basketball Video at time $[t, t+10]$
10x3x480x640
f_{crop}
10x3x480x640
10x3x321x321
Structure from Motion
Player's location on the court
f_{event}
10x1
10x1
10x1
10x1
Is the player in the 2 point zone?
8x1
40x1
f_{gm}
Gaussian Mixtures
Assessment Feature
(μ_1, Σ_1)
(μ_2, Σ_2)
$(\mu_{255}, \Sigma_{255})$
$(\mu_{256}, \Sigma_{256})$
256x1
Predicted Performance Measure for time $[t, t+10]$

Figure 53: A detailed illustration of our basketball assessment prediction scheme. Given a video segment from time interval $[t, t+10]$, we first feed it through a function f_{crop}, which zooms-in to the relevant parts of a video. We then apply f_{event} to predict 4 atomic basketball events from a zoomed-in video and a player's (x, y) location on the court. We then feed these predictions through a Gaussian mixture function f_{gm}, which produces a highly non-linear visual spatiotemporal assessment feature. Finally, we use this feature to compute a player's assessment measure by multiplying it with linear weights learned from the data, and with a predicted relevance indicator for a given video segment.

maximum likelihood from the training data with diagonal covariances.

3.3.2. Basketball Assessment Prediction

We learn a linear weight $\mathbf{w}$ in Equation (3.1) based on the comparative assessment of players provided by a former professional basketball player in Section 3.4. We minimize the following hinge loss:

$$\mathcal{L}_{\mathbf{w}} = \sum_{i=1}^{D} \max\left(0, \left(\frac{1}{2} - Y_i\right)\left(S(\mathcal{V}_1^i) - S(\mathcal{V}_2^i)\right)\right), \tag{3.3}$$

where $Y_i = 1$ if a basketball expert declared Player 1 to be better than Player 2; otherwise $Y_i = 0$. $S(\mathcal{V}_1^i), S(\mathcal{V}_2^i)$ depict our predicted performance measure for Players 1, and 2 respectively, $\mathcal{V}_1^i$ and $\mathcal{V}_2^i$ are the first-person basketball videos of Player 1 and Player 2 respectively, and D is the number of data points. Then based on Equation 3.1, we can compute the subgradients of this loss function with respect to w and find w by minimizing it via a standard gradient descent. In Figure 54, we provide an illustration of such a learning framework.

Why Linear Classifier? We only have 250 labeled pairs for learning the weights, which is a small amount of training data. Thus, making a classifier more complex typically results

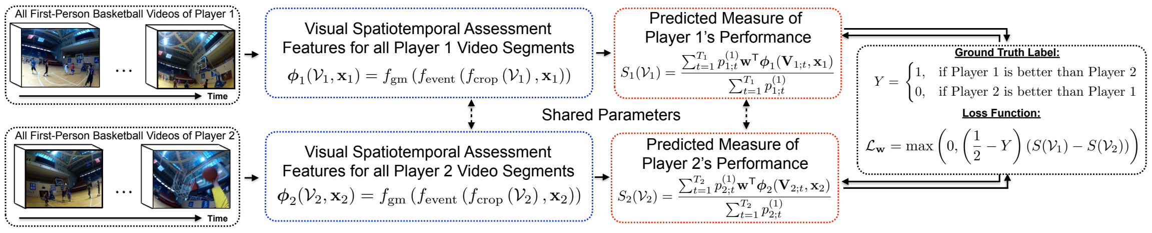

All First-Person Basketball Videos of Player 1
Time
All First-Person Basketball Videos of Player 2
Time
Visual Spatiotemporal Assessment Features for all Player 1 Video Segments
$\phi_1(\mathcal{V}_1, \mathbf{x}_1) = f_{\text{gm}}(f_{\text{event}}(f_{\text{crop}}(\mathcal{V}_1), \mathbf{x}_1))$
Visual Spatiotemporal Assessment Features for all Player 2 Video Segments
$\phi_2(\mathcal{V}_2, \mathbf{x}_2) = f_{\text{gm}}(f_{\text{event}}(f_{\text{crop}}(\mathcal{V}_2), \mathbf{x}_2))$
Shared Parameters
Predicted Measure of Player 1's Performance
$S_1(\mathcal{V}_1) = \frac{\sum_{t=1}^{T_1} p_{1;t}^{(1)} \mathbf{w}^\top \phi_1(\mathbf{V}_{1;t}, \mathbf{x}_1)}{\sum_{t=1}^{T_1} p_{1;t}^{(1)}}$
Predicted Measure of Player 2's Performance
$S_2(\mathcal{V}_2) = \frac{\sum_{t=1}^{T_2} p_{2;t}^{(1)} \mathbf{w}^\top \phi_2(\mathbf{V}_{2;t}, \mathbf{x}_2)}{\sum_{t=1}^{T_2} p_{2;t}^{(1)}}$
Ground Truth Label:
$Y = \begin{cases} 1, & \text{if Player 1 is better than Player 2} \\ 0, & \text{if Player 2 is better than Player 1} \end{cases}$
Loss Function:
$\mathcal{L}_{\mathbf{w}} = \max\left(0, \left(\frac{1}{2} - Y\right)(S(\mathcal{V}_1) - S(\mathcal{V}_2))\right)$

Figure 54: An illustration of of our training procedure to learn the linear weights w that are used to assess a given basketball player's performance. As an input we take a pair of labeled first-person basketball videos with a label provided by a basketball expert indicating, which of the two players is better. Then, we compute visual spatiotemporal basketball assessment features for all input video segments, and use them to learn weights w by minimizing our formulated hinge loss function.

in a severe overfitting. Through our experiments, we discovered that linear weights work the best.

3.3.3. Implementation Details

For all of our experiments involving CNNs, we used a Caffe library [16]. Both networks were based on DeepLab's [88] architecture and were trained for 4000 iterations with a learning rate of 10^{-8}, 0.9 momentum, the weight decay of $5 \cdot 10^{-5}$, and 30 samples per batch. The LSTM layers inside the atomic basketball event network spanned 10 consecutive frames in the video input. Each pathway in the atomic basketball event network was composed of two 1024 dimensional convolution layers with kernel size 1×1 and a 1024 dimensional LSTM layer. The networks were trained using standard data augmentation. To learn the weights w we used a learning rate of 0.001 and ran gradient descent optimization for 100 iterations.

3.4. First-Person Basketball Dataset

We present a first person basketball dataset composed of 10.3 hours of videos with 48 college players. Each video is about 13 minutes long captured by GoPro Hero 3 Black Edition mounted with a head strip. It is recorded at 1280×960 with 100 fps. We record 48 videos during the two days, with a different group of people playing each day. We use 24 videos from the first day for training and 24 videos from the second day for testing. We

	Atomic Events			
	$p^{(1)}$	$p^{(2)}$	$p^{(3)}$	mean
Tran et al. [120]	0.312	0.428	0.193	0.311
Singh et al [219]	0.469	0.649	0.185	0.434
Bertasius et al [7]	0.548	0.723	0.289	0.520
Ma et al [154]	0.622	0.718	0.364	0.568
Ours: no LSTM & no zoom-in	0.711	0.705	0.192	0.536
Ours: no zoom-in	0.693	0.710	0.248	0.550
Ours: single path	0.678	0.754	0.308	0.580
Ours: no LSTM	0.718	0.746	**0.397**	0.620
Ours	**0.724**	**0.756**	0.395	**0.625**

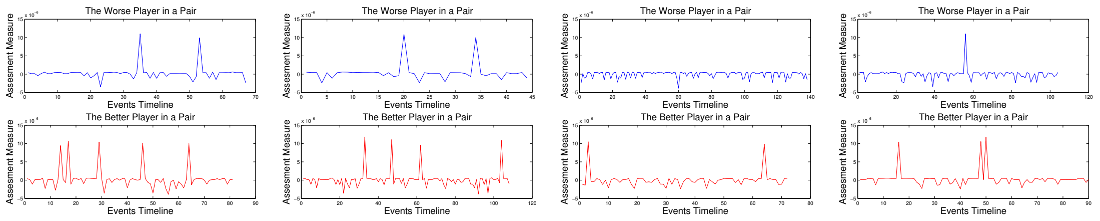
Table 29: The quantitative results for atomic basketball event detection on our first-person basketball dataset according to max F-score (MF) metric. These results show that our method 1) outperforms prior first-person methods and 2) that each component plays a critical role in our system.

extract the video frames at 5 fps to get $98,452$ frames for training, and $87,393$ frames for testing.

We ask a former professional basketball player (played in an European national team) to label which player performs better given a pair of first-person videos. Total 500 pairs are used: 250 for training and 250 for testing. Note that there were no players overlapping between the training and testing splits.

We also label three simple basketball events: 1) somebody shooting a ball, 2) the camera wearer possessing the ball, and 3) a made shot. These are the key atomic events that drive a basketball game. In total, we obtain $3,734$, $4,502$, and $2,175$ annotations for each of these three events respectively.

Furthermore, to train a ball detector we label the location of a ball at $5,073$ images by clicking once on the location. We then place a fixed sized Gaussian around those locations and use it as a ground truth label.



(a) A Pair of Players #1 (b) A Pair of Players #2 (c) A Pair of Players #3 (d) A Pair of Players #4

Figure 55: We randomly select 4 pairs of basketball players, and visualize how our assessment model evaluates each player over time. The red plot denotes the better player in a pair, whereas the blue plot depicts the worse player. The y-axis in the plot illustrates our predicted performance measure for an event occurring at a specific time in a player's first-person video.

3.5. Experimental Results

3.5.1. Quantitative Results

Atomic Basketball Event Detection. In Table 29, we first illustrate our results for atomic basketball event detection task. The results are evaluated according to the maximum F-score (MF) metric by thresholding the predicted atomic event probabilities at small intervals and then computing a precision and recall curve. First, we compare our model's predictions with several recent first-person activity recognition baselines [219, 7, 154] and also with the successful video activity recognition baseline C3D [120]. We show that our model outperforms all of these baselines for each atomic event.

Furthermore, to justify our model's design choices, in Table 29 we also include several experiments studying the effect of 1) a multi-path architecture, 2) LSTM layers, and 3) zooming-in scheme. Our experiments indicate that each of these components is crucial for achieving a solid atomic event recognition accuracy, i.e. the system achieves the best performance when all three of these components are included in the model.

Basketball Assessment Results. In Table 30, we present our results for assessing 24 basketball players from our testing dataset. To test our method's accuracy we evaluate our method on 250 labeled pairs of players, where a label provided by a basketball expert

	Accuracy	
	Pred. Events	GT Events
LRCN [220] 2-pt made shot detector	fail	-
LRCN [220] 3-pt made shot detector	fail	-
Ours: no GMs	0.477	-
Ours: no $p^{(3)}$	0.496	-
Ours: no $p^{(2)}$	0.515	-
Ours: no $p^{(1)}$	0.536	-
Ours: single GM-top2	0.537	-
Ours: all weights w set to 1	0.583	-
Ours: single GM-top1	0.609	-
Ours: no $p^{(4)}$	0.649	-
Ours	**0.765**	**0.793**

Table 30: The quantitative results for our basketball assessment task. We evaluate our method on 250 labeled pairs of players, and predict, which of the two players in a pair is better. We then compute the accuracy as the fraction of correct predictions. We report the results of various baselines in two settings: 1) using our predicted atomic events, and 2) using ground truth atomic events. These results show that 1) our model achieves best results, 2) that each of our proposed components is important, and 3) that our system is pretty robust to atomic event recognition errors.

indicates, which of the two players is better. For each player, our method produces an assessment measure indicating, which player is better (the higher the better). To obtain the accuracy, we compute the fraction of correct predictions across all 250 pairs.

We note that to the best of our knowledge, we are the first to formally investigate a basketball performance assessment task from a first-person video. Thus, there are no well established prior baselines for this task. As a result, we include the following list of baselines for a comparison.

First, we include two basketball activity baselines: the detectors of 1) 2-point and 2) 3-point shots made by the camera wearer. We label all instances in our dataset where these activities occur and discover ≈ 100 of such instances. Note that such a small number of instances is not a flaw of our dataset, but instead an inherent characteristic of our task. Such basketball activities belong to a long-tail data distribution, i.e. they occur pretty rarely, and thus, it is difficult to train supervised classifiers for such activity recognition. We then train an LRCN [220] model as 1) a 2 point made shot detector, and 2) a 3 point made shot detector. We report that due to a small amount of training data, in all cases the network severely overfit the training data and did not learn any meaningful pattern.

Furthermore, to justify each of our proposed components in the model, in Table 30 we also include several ablation baselines. First, we study how 1) Gaussian Mixtures (GM) and 2) the process of learning the weights affect the performance assessment accuracy. We do it 1) with our predicted and 2) with the ground truth atomic events. We show that in both cases, each of our proposed components is beneficial. In addition, we also observe that our system is robust to atomic event recognition errors: the accuracy when using the ground truth atomic events is only 2.8% better compared to our original model.

We also present the performance assessment results when we remove one of the four atomic events from our system. We show that our method performs the best when all four atomic events are used, suggesting that each atomic event is useful. Finally, as two extra baselines we manually select two Gaussian mixtures with the largest weight magnitudes and use each of their predictions independently (denoted as single GM-top1,2 in Table 30). We show that our full model outperforms all the other baselines, thus, indicating that each of our proposed component in our model is crucial for an accurate player performance assessment.

3.5.2. Qualitative Results

In addition, in Figure 55, we also include a more dynamic visualization of how our assessment model works over time. To do this, we randomly select 4 pairs of basketball players, and visualize how our model evaluates each player over time. The red plot in each pair denotes the better player, whereas the blue plot depicts the worse player. The y-axis in the plot illustrates our predicted performance measure for an event occurring at a specific time in a player's first-person video.

Furthermore, in Figure 57 we also include examples of short sequences, illustrating 1) a player's actions that contributed most positively to his/her performance assessment and also 2) actions that contributed most negatively. We select these action sequences by picking the first-person video sequences with a largest positive and negative values of the terms inside

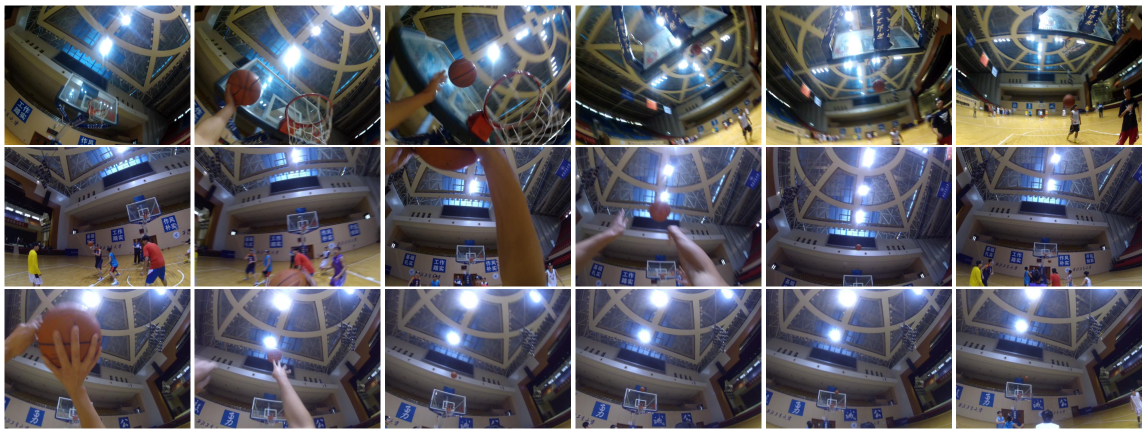
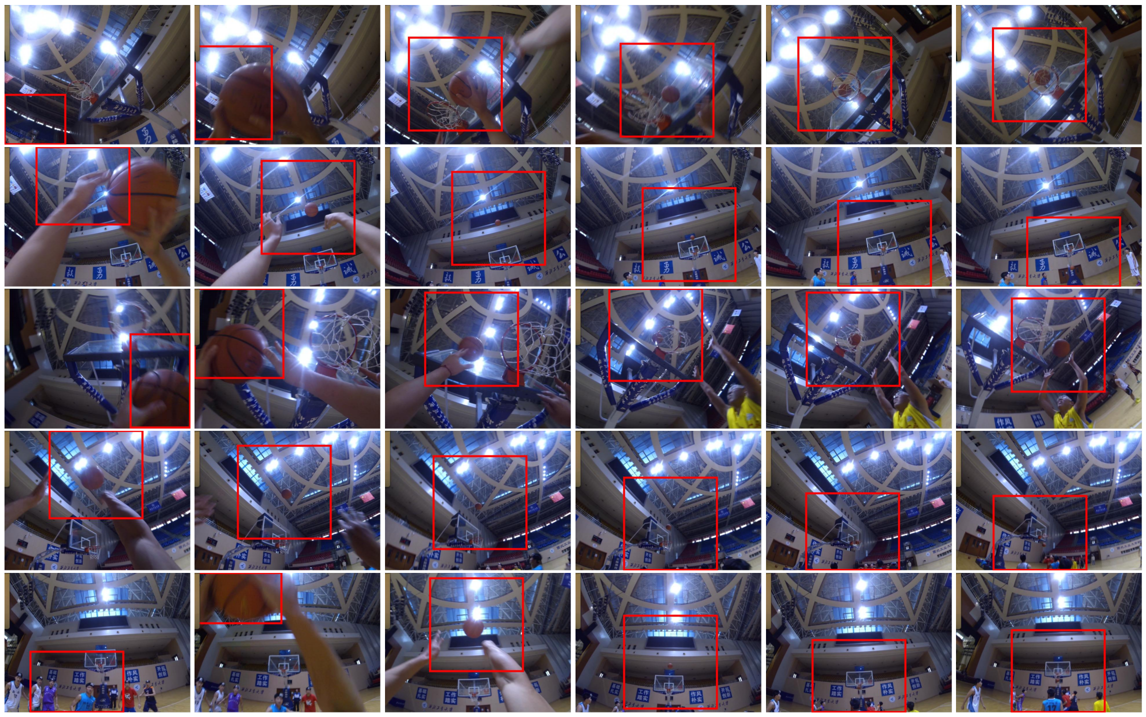


Figure 56: A visualization of basketball activities that we discovered by manually inspecting Gaussian mixtures associated with the largest basketball assessment model weights w. Each row in the figure depicts a separate event, and the columns illustrate the time lapse of the event (from left to right), We discover that the two most positive Gaussian mixtures correspond to the events of a player making a 2 point and a 3 point shot respectively (the first two rows), while the mixture with the most negative weight captures an event when a player misses a 2 point shot (last row).

the summation of Equation 3.1 (which also correspond to positive and negative peaks from Figure 55). Such terms depict each video segment's contribution to the overall basketball skill assessment measure.

We would like to note that it is quite difficult to include such results in an image format, because 1) images are static and thus, they cannot capture the full content of the videos; 2) images in the paper, appear at a very low-resolution compared to the original 480×640 videos, which makes it more difficult to understand what kind of events are depicted in these images. To address some of these issues, we include more of such qualitative examples in a video format at: `https://www.youtube.com/watch?v=lxEuihbPJXA`.

Understanding the Feature Representation. Earlier, we claimed that Gaussian mixtures produce a highly non-linear feature representation. We now want to get a better insight into what it represents. To do so we analyze the learned weights w, and then manually inspect the Gaussian mixtures associated with the largest magnitude weights in w. Upon doing so we discover that the two mixtures with the most positive weights learn to


(a) The detected events that contributed most **positively** to a player's performance.

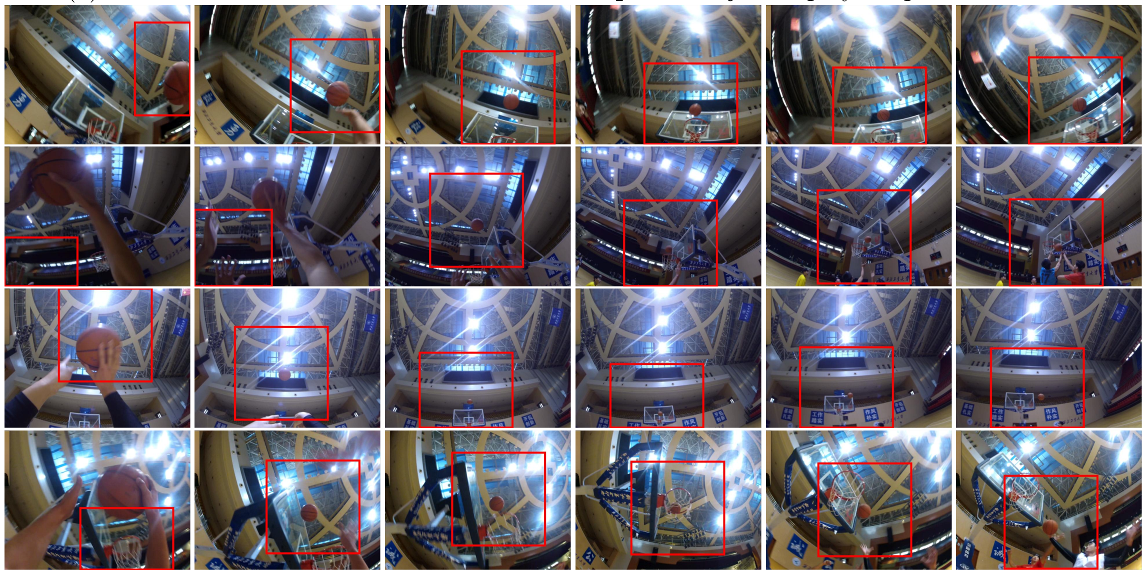

(b) The detected events that contributed most **negatively** to a player's performance.

Figure 57: A figure illustrating the events that contribute most positively (top figure) and most negatively (bottom figure) to a player's performance measure according to our model. The red box illustrates the location where our method zooms-in. Each row in the figure depicts a separate event, and the columns illustrate the time lapse of the event (from left to right). We note that among the detected positive events our method recognizes events such as assists, made layups, and made three pointers, whereas among the detected negative events, our method identifies events such as missed layups, and missed jumpshots.

capture basketball activities when camera wearer makes a 2 point shot, and a 3 point shot respectively. Conversely, the mixtures with the two most negative weights represent the activities of the camera missing a 2 point shot, and the camera wearer's defender making a shot respectively. In Figure 56, we include several sequences corresponding to such discovered activities.

3.6. Conclusions

In this work, we introduced a basketball assessment model that evaluates a player's performance from his/her first-person basketball video. We showed that we can learn powerful visual spatiotemporal assessment features from first-person videos, and then use them to learn our skill assessment model from the pairs of weakly labeled first-person basketball videos. We demonstrated that despite not knowing the labeler's assessment criterion, our model learns to evaluate players with a solid accuracy. In addition, we can also use our model to discover the camera wearer's activities that contribute positively or negatively to his/her performance assessment.

We also note that performance assessment is an important problem in many different areas not just basketball. These include musical instrument playing, job related activities, and even our daily moments such as cooking a meal. In our future work, we plan to investigate these new areas, and try to generalize our model to such activities too.

CHAPTER 4 : Egocentric Basketball Motion Planning from a Single First-Person Image

4.1. Introduction

Consider LeBron James, arguably the greatest basketball player in the world today. People often speculate about which of his traits contributes most to his success as a basketball player. Some may point to his extraordinary athleticism, while others may credit his polished basketball mechanics (i.e., his ability to accurately pass and shoot the ball). While both of these characteristics are certainly important, there seems to be a general consensus, among sports pundits and casual fans alike, that what makes LeBron truly exceptional is his ability to make the right decision at the right time in almost any basketball situation.

Now, imagine yourself inside the dynamic scene shown in Figure 58, where, as a basketball player, you need to make a series of moves to outmaneuver your defender and score a basket. In doing so, you must evaluate your position on the court, your defender's stance, your defender's court position relative to the basket, and many other factors, if you want to maximize your probability of scoring. For instance, if you observe that your defender is positioned close to you and leaning more heavily on *his* right leg, you might exploit this situation by taking a swift left jab-step followed by a hard right drive to the basket, throwing your defender off balance and earning you an easy layup. However, if your defender is standing farther away from you, you may decide to take a few dribbles into an area where your defender cannot get to you quickly, before stopping for a jump-shot. These are all complex decisions that have to be made within a split second – doing this consistently well is challenging.

In this paper, we aim to build a model that mimics this basketball decision-making process. Specifically, our model maps a first-person visual signal to a plausible egocentric basketball motion sequence. This is a difficult problem because little is known about how skilled players make subsecond-level decisions. What we do know is that players make decisions

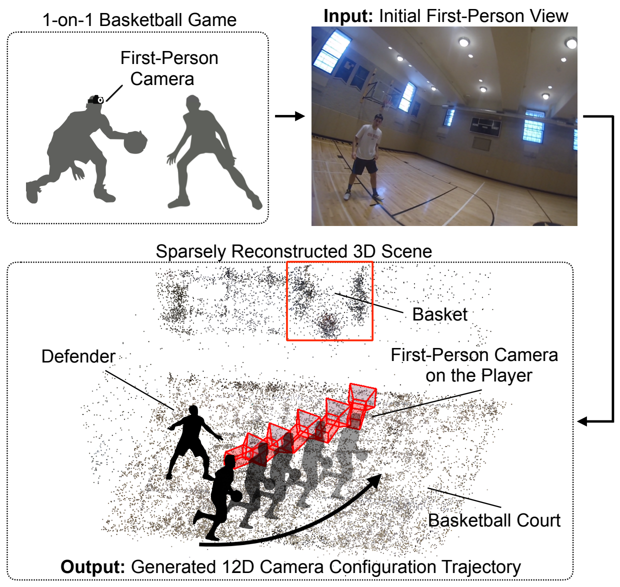

1-on-1 Basketball Game
First-Person Camera
Input: Initial First-Person View
Sparsely Reconstructed 3D Scene
Basket
Defender
First-Person Camera on the Player
Basketball Court
Output: Generated 12D Camera Configuration Trajectory

Figure 58: Given a single first-person image from a one-on-one basketball game, we aim to generate an egocentric basketball motion sequence in the form of a 12D first-person camera configuration trajectory, which encodes a player's 3D location and 3D head orientation. In this example, we visualize our generated motion sequence within a sparse 3D reconstruction of an indoor basketball court.

based on what they see: a player may look at how his or her defender is positioned (i.e., feet orientation, torso orientation, etc.) and then, based on that visual information, choose to drive right or left. Leveraging this insight, in this work, we learn our model from data recorded using first-person cameras. First-person cameras allow us to see various subtle details in the basketball game, just as the players do. In contrast, a standard third-person camera typically records a low-resolution view from a suboptimal orientation, making it difficult to observe such details. Because these subtle details may be crucial for predicting where a player will move next, first-person modeling is beneficial.

Our model takes a single first-person image as input and generates an egocentric basketball motion sequence in the form of a 12D first-person camera configurations, encoding a player's 3D location and 3D head orientation throughout the sequence. To do this, we first introduce a future convolutional neural network (CNN) that predicts an initial sequence of 12D camera configurations, aiming to capture how real players move during a one-on-one basketball game. We also introduce a goal verifier network, which is trained to verify that a given camera configuration is consistent with the final goals of real players. Next, we use our proposed inverse synthesis procedure to synthesize a refined sequence of 12D camera configurations by optimizing the following objectives: (1) minimize the difference between the refined configurations and initial configurations predicted by future CNN while also (2) maximizing the goal verifier network output. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence.

Our egocentric basketball motion model learns to generate smooth and realistic sequences that capture the goals of real basketball players. Additionally, our model is learned in an unsupervised fashion; this is an important advantage, since obtaining labeled behavior data is costly. Finally, in our experimental section, we show that our method consistently outperforms standard deep learning approaches such as RNNs [221], LSTMs [222], and GANs [223].

4.2. Related Work

Using Vision for Behavior Modeling. Developing models that characterize human behavior in everyday tasks, such as walking or driving, has been a long-standing problem in computer science. The work in [224] uses a hidden Markov model (HMM) to learn human driving patterns. Recent work in [225, 226] uses third-person videos of humans performing simple tasks, like opening a door, to teach robots how to do the same. By observing them from a third-person view, the method in [227] learns to remind humans of actions they forgot to do. Recently, there also has been a surge of methods designed to capture various aspects of social human behavior. For instance, the work in [228] uses a flow field model to track crowds. Also, the work in [229] develops a tracking method that can predict walking trajectories for multiple agents simultaneously, while more recent work in [230] predicts walking trajectories by using a game theoretic approach. Furthermore, the methods in [231, 232, 233, 234] develop social force or social affinity based methods for predicting pedestrian behaviors. In addition, the method in [235], uses Markov decision processes to predict the wide receiver trajectories in football, whereas the work in [236] learns to predict the behavior of soccer players. Finally, the work in [237] uses large amounts of "top-view" basketball tracking data [196] to train deep hierarchical networks for basketball trajectory prediction.

In contrast to these prior methods, our data are obtained via first-person cameras, which allows us to build a model that connects first-person visual sensation to egocentric motion planning ability. Using such first-person data is our main advantage compared to prior methods.

First-Person Vision. Most first-person methods have focused on first-person object detection [172, 211, 153, 168, 7, 8] or activity recognition [215, 238, 219, 169, 152, 154, 170]. Several methods have also employed first-person videos for video summarization [172, 173]. Furthermore, recent work in [239] introduced a first-person method for predicting the camera wearer's engagement, while [9] developed a model to assess a basketball performance from a first-person video. Additionally, [240] proposed applying inverse reinforcement learning on first-person data to infer the goals of the camera wearer during daily tasks. Furthermore, the methods in [212, 203] propose to generate plausible walking trajectories from first-person images. However, the methods in [203, 212] generate walking trajectories without really understanding what the camera-wearer's actual goals are, often resulting in overly simplistic and unrealistic trajectories.

In contrast to these prior methods, we consider a complex one-on-one basketball game scenario, for which we not only infer the goals of the camera-wearer, but also generate an egocentric basketball motion sequence that is aligned with these goals. We also point out that, unlike the methods in [203, 212] which require depth input or the positions of other people in the scene, our method operates using only a single first-person RGB input image.

Representing a State. Generating an egocentric basketball motion sequence can be viewed as the problem of mapping a first-person image x to a sequence $y_{1:M}$, where each entry y_i is a vector representing a state in this sequence of length M. In this paper, we use a notation where the entire sequence is denoted as y (without subscript indices), whereas an i^{th} state in the sequence y is denoted as y_i.

We propose to encode each state y_i in the sequence as a 12D camera configuration, which captures 3D location and 3D orientation of the camera on a player's head. In contrast to prior models [212, 203] that just use a 2D (x, y) location representation, our selected camera configuration representation allows us to represent more complex basketball motion patterns, while still being compact and interpretable. Such representation also enables us to learn our model without using manually labeled behavioral data.

Generating Motion Sequences. Generating motion sequences that are realistic and smooth is a challenging problem because the actions taken by the camera wearer are noisy. As a result, it is difficult to accurately learn the transitions between two adjacent states from a limited amount of data. Furthermore, due to the recurrent nature of such predictions (i.e., the predicted state $\hat{y}_{i+1}$ depends on $\hat{y}_i$), the error starts accumulating and exploding, which makes it difficult to generate longer sequences accurately. This issue is especially prevalent with the standard RNN and LSTM approaches, which try to learn such transitions sequentially.

In this work, we propose a model that first predicts a set of intermediate states that are consistent with how real basketball players move. Afterwards, we use our proposed inverse synthesis procedure to generate a full motion sequence. In the experimental section, we verify the effectiveness of our approach and show that it outperforms methods such as RNNs, LSTMs, and GANs.

In Figure 59, we present a detailed illustration of our egocentric basketball motion model. First, we use our proposed future CNN to predict an initial sequence of 12D camera configurations, which aim to capture how real players move during a one-on-one basketball game. After that, we use our proposed inverse synthesis procedure to generate a refined camera configuration sequence that (1) matches the initial camera configuration sequence predicted by the future CNN and (2) maximizes the output of the goal verifier network, which is trained to verify whether a given 12D camera configuration aligns with realistic one-on-one basketball goals. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence. We now discuss each component of our model in more detail.

4.4.1. Egocentric Camera (EgoCam) CNN

An egocentric camera (EgoCam) CNN is used to map a given first-person image x_i into a 12D camera configuration $\hat{y}_i \in \mathbb{R}^{1\times 12}$ that encodes the 3D location and 3D orientation of the camera on a player's head. We implement our EgoCam CNN using a popular ResNet-101 [241] architecture. The network is optimized using the following L2 loss:

$$L_{cam} = ||\hat{y}_i(x_i) - K(x_i)||_2^2 \tag{4.1}$$

where $\hat{y}_i \in \mathbb{R}^{1\times 12}$ is the predicted camera configuration for an image x_i, and $K(x_i) \in \mathbb{R}^{1\times 12}$ is a vector that encodes a true player's 3D location and 3D head orientation. $K(x_i)$ is obtained by flattening the egocentric 3×4 camera matrix, which is produced by an unsupervised structure from motion (SfM) algorithm. However, due to the jittery nature of first-person images, SfM works only on a small portion of the input images (i.e.$\approx$ 20% of all images). This motivates the need for an EgoCam CNN, which allows us to generate

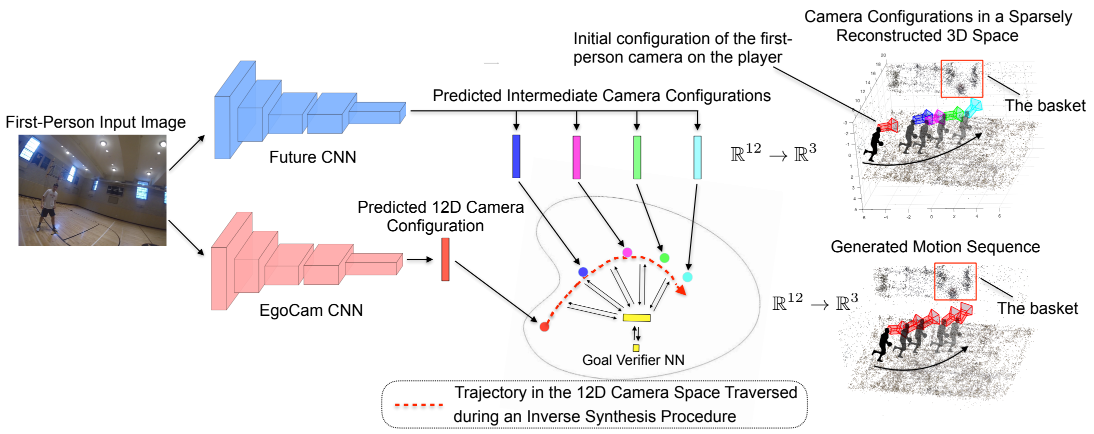

First-Person Input Image
Future CNN
EgoCam CNN
Predicted Intermediate Camera Configurations
Predicted 12D Camera Configuration
Initial configuration of the first-person camera on the player
Camera Configurations in a Sparsely Reconstructed 3D Space
The basket
$\mathbb{R}^{12} \to \mathbb{R}^3$
Generated Motion Sequence
The basket
$\mathbb{R}^{12} \to \mathbb{R}^3$
Goal Verifier NN
Trajectory in the 12D Camera Space Traversed during an Inverse Synthesis Procedure

Figure 59: Our model takes a single first-person image as input and outputs an egocentric basketball motion sequence in the form of a 12D camera configuration trajectory, encoding a player's 3D location and 3D head orientation throughout the sequence. First, we feed the first-person image through our proposed future CNN to predict an initial sequence of future 12D camera configurations. Then, we use our proposed inverse synthesis procedure to synthesize a refined camera configuration sequence that matches the configurations predicted by the future CNN, while also maximizing the output of the goal verifier network. The goal verifier network is a fully-connected network trained to verify that a given 12D camera configuration is consistent with the final goals of real players. Finally, by following the trajectory resulting from the refined camera configuration sequence, we obtain the complete 12D motion sequence.

12D configurations for any first-person image.

Our EgoCam CNN produces similar outputs to the actual SfM algorithm. We also report that we tried retrieving missing 12D configurations using a nearest neighbor algorithm, but observed that our EgoCam CNN produced better results (**2.54** vs. **1.97** L2 error in the normalized 12D space).

4.4.2. Future CNN

Given a first person image x_i from the beginning of a sequence, the future CNN encodes it into multiple configurations $\phi(x_i) \in \mathbb{R}^{k\times 12}$, which represent intermediate states that capture how real players move during a one-on-one game. Here, k is a parameter that controls how many intermediate configurations we want to generate. Our future CNN is implemented using a multi-path ResNet-101 architecture, meaning that after the final convolutional *pool5* layer, the network splits itself into k branches, each denoted as $\phi_j(x_i)$ for $j = 1 \dots k$. Each branch in the future CNN is responsible for generating its own intermediate state.

To train the future CNN, for every real basketball sequence, we first assign to each camera configuration y_i of the sequence a value $s \in [0, 1]$, indicating its order in the sequence. Let y_i be the i^{th} configuration in a sequence of length M. Then, we can compute s as $s(i) = i/M$. The j^{th} branch in the future CNN is then optimized to predict camera configurations, with s values falling in the interval $[\frac{j-1}{k}, \frac{j}{k}]$, where k is the number of branches in the future CNN. For instance, if $k = 4$, then the second branch in the network will be optimized to predict camera configurations with values $s \in [0.25, 0.5]$. This constraint ensures that each branch generates an intermediate state associated with a specific time-point in a sequence, thus generating configurations covering the entire sequence (albeit with wider gaps). We also point out that all input images x_i that are used to train the future CNN have values $s \in [0, 0.1]$ to make sure that the future CNN generates accurate intermediate configurations from a first-person image at the beginning of a sequence. The future CNN is then trained using the following loss function:

$$L_{fut} = \sum_{j=1}^{k} ||\phi_j(x_i) - \hat{y}_{i'}(x_{i'})||_2^2 \tag{4.2}$$

where ϕ_j denotes the output from branch j of a future CNN, and $\hat{y}_{i'}$ is the output of an EgoCam CNN for an input image $x_{i'}$. The constraints on x_i and $x_{i'}$ that we discussed above are expressed as: $s(i) \in [0, 0.1]$ and $s(i') \in [\frac{j-1}{k}, \frac{j}{k}]$.

We note that our training setup requires basketball sequences that are trimmed. In this work, we trim the sequences manually: the sequence starts when a player begins his offensive position and ends when a shot or a turnover occurs. We believe that we could also trim sequences automatically by using a player's head location and pose to assess whether a player is attacking or defending.

4.4.3. Goal Verifier Network

The goal verifier network aims to verify whether a given configuration aligns with the final goals of real players. The goal verifier takes a 12D camera configuration $\hat{y}_i \in \mathbb{R}^{1\times 12}$ as input, then outputs a real value $\psi(\hat{y}_i) \in \mathbb{R}^{1\times 1}$ in the interval $[0, 1]$, indicating how well a given camera configuration captures the final goals of real players.

The key question is: how do we infer the final goals of real basketball players without asking them directly? To do this, we assume that the last few images in a given sequence represent the final goals of that player. Of course, such reasoning may not always be correct because, in some sequences, players may fail to accomplish their goals. However, we conjecture that when looking at many sequences of real players, the pattern of goals should be quite distinctive. In other words, if we use the right learning strategy, the network should be able to learn configurations that are typically associated with the final goals of real players.

We propose to employ discriminative training of the goal verifier, which allows us to effectively address the issues of noisy labels in our setting. To create these noisy labels, we assign a value $g(\hat{y}_i) = 1$ to all configurations with $s(i) > 0.92$ (configurations that appear at the end of sequences) and a value of zero to all other configurations. Such a scheme allows us to highlight the configurations at the end of sequences, which are more likely to represent the goals of the players. Subsequently, the goal verifier, which is a two-layer network, takes a 12D camera configuration as its input and is trained using a cross-entropy loss:

$$L_g = -\big[g(\hat{y}_i) \log \psi(\hat{y}_i) + (1 - g(\hat{y}_i)) \log (1 - \psi(\hat{y}_i))\big] \tag{4.3}$$

where $\hat{y}_i \in \mathbb{R}^{1\times 12}$ is the 12D camera configuration output from an EgoCam CNN (associated with an image x_i), and $\psi(\hat{y}_i) \in \mathbb{R}^{1\times 1}$ is the output of a goal verifier network that indicates whether a given 12D camera configuration accurately represents the final goals of real players.

4.4.4. *Motion Planning using Inverse Synthesis*

We can now put all the pieces of our model together and show how to generate an egocentric basketball motion sequence that captures the goals of real players. Our approach is partially inspired by the idea of synthesizing the preferred stimuli in a CNN [242, 243, 244], which has been mostly used to visualize activations in the CNNs [242, 243, 244]. In this work, we propose the concept of inverse synthesis for generating a realistic basketball motion sequence.

Consider the two problems that we were solving previously: (1) encoding an egocentric basketball motion sequence via a set of intermediate 12D configurations and (2) verifying that a given 12D camera configuration aligns with the final goals of real players. In the previous sections, we solved these two problems using the future CNN and the goal verifier network, respectively. However, we now want to invert these two problems: given (1) a set of intermediate states predicted by the future CNN and (2) a trained goal verifier network, our goal is to generate a smooth basketball motion sequence in the form of 12D camera configurations.

To do this, we frame the inverse synthesis problem as a search problem in the 12D camera space, where our goal is to find a 12D configuration h that (1) matches intermediate states generated by the future CNN $\phi_j(x_i)$ and (2) maximizes the output of a goal verifier network $\psi(h)$. We formulate this problem as a minimization problem in the 12D camera configuration space:

$$h^* = \arg\min_h \left[||\phi_j(x_i) - h||_2^2 - \log\left(\psi(h)\right) \right] \tag{4.4}$$

Here, h is a 12D camera configuration vector that we initialize to $\hat{y}_i(x_i)$, which is the camera configuration of the very first image in the sequence (i.e., $s(i) = 0$, or, equivalently, $i = 0$). The first term in the equation encourages h to reach configurations predicted by the future CNN, whereas the second term encourages h to take values that would maximize the output of the goal verifier network. We minimize this function by computing the appropriate gradients and then running gradient descent in the 12D camera configuration space for N iterations. To select the branch from a future CNN whose output ϕ_j we are trying to match, we compute $j = floor(c/(N/k))$, where j is the index of a selected branch, c is the iteration counter, and k is the number of branches in the future CNN.

During the inverse synthesis procedure, we fix the parameters of all three networks and only adjust h. Since the inverse synthesis is performed in the 12D camera configuration space, at every step we are generating a new 12D camera configuration. Thus, at the end, we can generate a final camera configuration sequence by simply following the 12D camera trajectory traversed during the optimization.

4.4.5. Implementation Details

For all of our experiments, we used the Caffe library [16]. Both the future and EgoCam CNNs were based on the ResNet-101 [241] architecture and were trained for $10,000$ iterations, with a learning rate of 10^{-4}, 0.9 momentum, a weight decay of $5 \cdot 10^{-4}$, and 20 samples per batch. We designed our future CNN to have 4 distinct branches, which we experimentally discovered to work well. Each branch consists of a single fully-connected layer that maps *pool5* features to a 12D camera configuration feature. Next, we designed the goal verifier network as a two layer fully connected network with 100 neurons in the first-hidden layer and a single output neuron at the end. To generate final basketball motion sequences, we ran our inverse synthesis procedure for $6,000$ iterations with a step size of 0.001.

4.5. Egocentric One-on-One Basketball Dataset

We present a first-person basketball dataset consisting of 988 sequences from one-on-one basketball games between nine college-level players. The videos are recorded in 1280×960

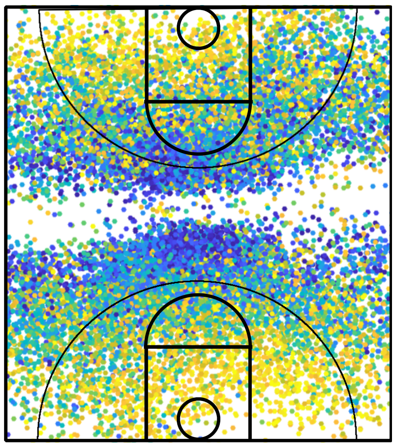


Figure 60: A figure illustrating the 3D locations of every camera configuration from our dataset mapped on a 2D court (best viewed in color). The blue points indicate the configurations from the beginning of sequences, whereas the yellow points depict configurations from the end of sequences. The diversity of real sequences in our dataset allows us to build a powerful basketball motion model.

resolution at 100 fps using GoPro Hero Session cameras, which are mounted on the players' heads. We extract the videos at 5 frames per second. We then randomly split all the sequences into training and testing sets, consisting of 815 and 173 sequences, respectively. Each sequence is about 25 frames long.

To obtain better insight about the distribution of our data, we map the 3D locations of all camera configurations from real sequences onto the 2D court and visualize the results in Figure 60. The blue points indicate the configurations at the beginning of sequences, whereas the yellow points represent configurations at the end of sequences. Based on this figure, we note that our dataset contains a wide array of different beginning and ending configurations, enabling us to learn a diverse basketball motion model.

We chose a one-on-one setting because it is more data-efficient for studying the decision-making process of basketball players (compared to the five-on-five setting). For example, first-person videos of five-on-five games capture numerous instances when players do not have the ball and are idly waiting for their teammate with the ball to perform some action. On the contrary, in our one-on-one dataset, players continuously make decisions and take actions to reach their goals, as there are no teammates to rely on.

4.6. Experimental Results

To the best of our knowledge, we are the first to generate egocentric basketball motion sequences in the 12D camera configuration space from a single first-person image. As a result, there are no prior established baselines for this task. To validate our method's effectiveness, we compare it with other popular deep learning techniques such as RNNs [221], LSTMs [222], and GANs [223]. We construct these baselines using the ResNet-101 [241] architecture. Each baseline takes a single first-person image as input and is then trained to generate 10 future configurations. The RNN and LSTM baselines are optimized with an L2 loss in the 12D camera configuration space, whereas the GAN baseline has the same architecture as RNN, but is trained to fool the discriminator by producing realistic 12D future camera configurations. To maximize the amount of available training data, the training data are constructed by using every feasible training image as a starting point. During testing, each baseline predicts camera configurations recurrently until we observe no significant change in the prediction or until the predicted sequence length exceeds the length of a real sequence. We also include a nearest neighbor baseline, which operates in the 12D camera configuration space.

The goal of our evaluations is to verify that the generated sequences (1) are realistic and (2) that they capture the goals of real players. To do this, we propose several evaluation schemes, which we describe in the next few subsections.

4.6.1. Quantitative Results

Predicting Future Motion. To verify that our generated sequences are realistic, we examine whether our method can predict a player's future motion. In the context of such evaluation, the sequences that we want to compare typically have different lengths, which renders standard Euclidean and L1 distance metrics unusable. Thus, we use the Hausdorff distance [245], which is commonly used for sequential data comparisons. To implement this

	Evaluation Tasks	
	PFM $\downarrow$	CG $\uparrow$
GAN [223]	62.31	0.329
LSTM [222]	5.66	0.678
RNN [221]	5.82	0.612
NN	5.36	-
Ours w/o GV	**4.91**	0.671
Ours w/ GV	4.93	**0.776**

Table 31: We assess each method on two tasks: predicting a player's future motion (PFM), and capturing the goals of real players (CG). The $\uparrow$ and $\downarrow$ symbols next to a task indicate whether a higher or lower number is better, respectively. Our method outperforms the baselines for both tasks, suggesting that we can generate sequences that (1) are more similar to a true player's motion and (2) capture the goals of real players more accurately. We also note that removing a goal verifier ("Ours w/o GV") leads to a sharp decrease in performance, according to the CG evaluation metric.

idea, we first use our EgoCam CNN to extract camera configurations from every frame in every real sequence in the testing dataset. Then, as our evaluation metric, we compute a distance $d = D(y, \hat{y})$, where $\hat{y}$ is generated using the first image from a real sequence y, and D is a Hausdorff distance operator.

In Column 2 of Table 31, we record the average d over all testing sequences for every method (i.e., the lower the distance the better). We observe that our method outperforms all the other baselines, suggesting that we can use it for future behavior prediction.

Capturing the Goals of Real Players. Earlier, we assumed that the last few frames in real sequences approximate the goals of the players. To examine how well our method captures these goals, we first select the last generated configuration $\hat{y}_M$ in the sequence of length M. Then, we use nearest neighbor search to retrieve a real configuration y_{nn} that is most similar to $\hat{y}_M$. Finally, we look up where in the real sequence y_{nn} appears. For instance, if y_{nn} was the i^{th} configuration in a real sequence of length N, then we assign our current prediction a value of $v = i/N$. Intuitively, if the last generated configuration is similar to the last real configuration, we conclude that our method was able to capture the goals of real players.

In Column 3 of Table 31, we record the average v values over all testing sequences for every method (i.e., the higher the better). These results suggest that, out of all baselines, our method is the most accurate at capturing the goals of real players. Also, note that, if we remove the goal verifier, these results drop significantly, indicating the goal verifier's importance to our system. We conjecture that this happens because the future CNN is trained using a non-discriminative L2 loss, while the goal verifier network is trained to discriminate which configurations represent the final goals of the players. Thus, similar to GANs [223], due to the discriminative training, the goal verifier may be more useful for generating the configurations that capture the final goals of the players. We also note that the future CNN is still essential, as using a goal verifier network alone produces short and unrealistic sequences.

4.6.2. Qualitative Results

Future CNN Visualization. To better understand how the future CNN works, we visualize its outputs in Figure 62. To do this, we take the 12D camera configurations generated by the future CNN, transform them into 3D space, and then visualize them in a sparsely reconstructed 3D scene. Note that the future CNN produces a wide array of different configurations, allowing us to generate diverse sequences.

Furthermore, in Columns 2 and 3 of Figure 61, we visualize the activations of a future CNN from the *res4a* and *res5c* layers, respectively. We observe that, in the *res4a* layer, the CNN focuses on different parts of a defender's body, which makes intuitive sense, as basketball players often use such information to decide their next action. Furthermore, we also observe that, in the *res5c* layer, the future CNN learns to recognize open spaces in the court that could be used by a player to navigate the court and get away from a defender.

Visualizing Generated Motion Trajectories. In the last column of Figure 61, we also visualize sequences generated by our inverse synthesis procedure (by projecting them onto a 2D court). Even though it is difficult to validate whether our generated motion sequences represent optimal motion pattern, they do seem reasonable. In all visualized instances,

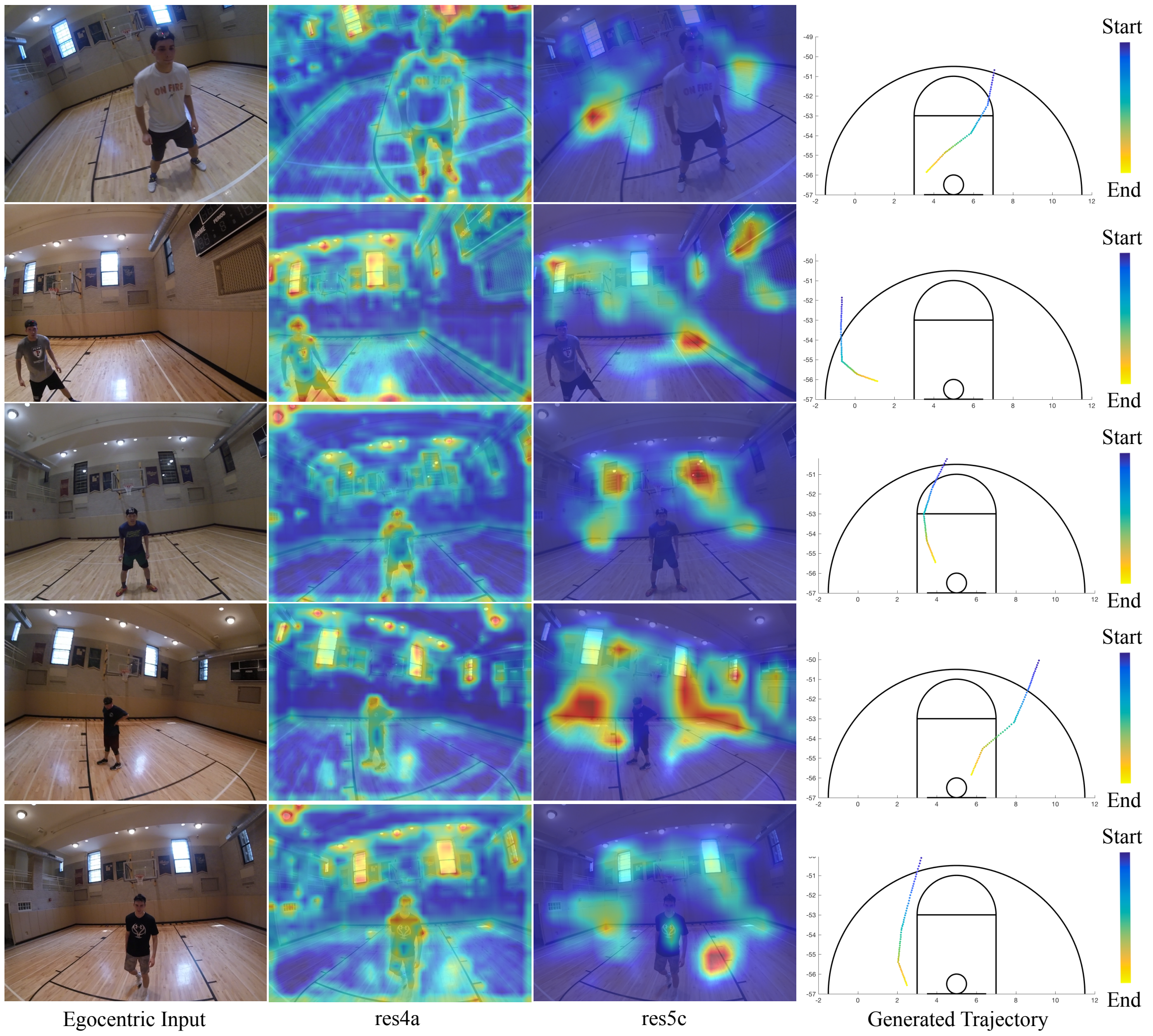


Figure 61: A figure illustrating some of our qualitative results. In the first column, we depict an egocentric input image. In the second and third columns, we visualize the activations of a Future CNN from the res4a and res5c layers. Finally, in the last column, we present our generated 2D motion trajectories. Based on the activations of the Future CNN, we conclude that our CNN recognizes visual cues in an egocentric image, which may be helpful for deciding how to effectively navigate the court and reach the basket, or how to get away from a defender. Furthermore, our generated motion trajectories seem realistic, as they avoid colliding with the defender and typically end near the basket, which reflects how most real players would move.

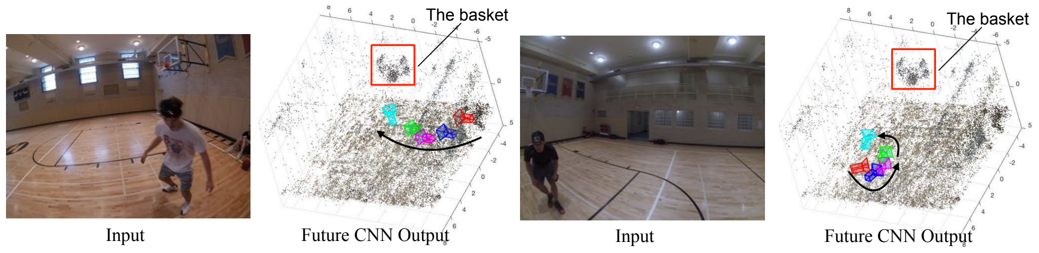


Figure 62: An illustration of the camera configurations generated by our future CNN, which we visualize in a sparsely reconstructed 3D space (best viewed in color). The red camera depicts the initial camera configuration state, while the blue, magenta, green, and cyan cameras correspond to the outputs from the $1^{st}, 2^{nd}, 3^{rd}$, and 4^{th} branches of our future CNN, respectively. We note that our future CNN is able to produce a diverse set of intermediate configurations, which allows us to generate a wide array of different sequences at a later step in our model.

our model produces motion sequences that avoid colliding with the defender. Furthermore, most of our generated sequences end around the basket, which reflects what a real player would likely try to do.

Retrieved Image Sequences. To show what our generated sequences look like in the form of first-person images, we use nearest neighbor search in the 12D camera space to retrieve the most similar first-person images from the training data. We then sample 5 first-person images from the sequence and visualize them in Figure 63, along with a real sequence. The similarity between our generated sequences and the real sequences suggests that our model accurately captures how real players move during a basketball game.

Additionally, to qualitatively validate the need for a goal verifier, in Figure 64, we visualize the very last images in our generated sequences, when the goal verifier is not used (w/o Goal Verifier) and when it is used (w/ Goal Verifier). Note that when we remove the goal verifier from our system, the very last images in our generated sequences look less realistic. That is, they focus on the walls or floor of the basketball court, which is probably not something that real players would do. In contrast, adding a goal verifier to our system makes these images focus on the basket, which makes much more sense, since most players finish their sequences when they are looking directly at the basket. Thus, the goal verifier allows us to

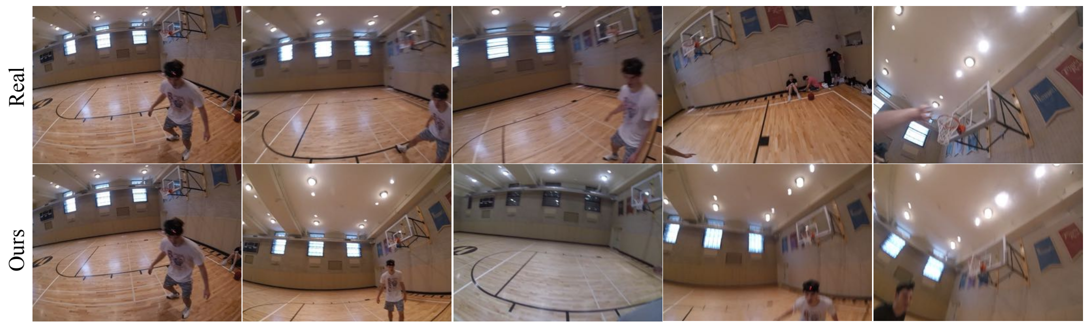


Figure 63: A visual illustration of our generated basketball sequences, in the form of first-person images retrieved using nearest neighbor search. We observe that our generated sequence is similar to a real sequence, suggesting that our model could be used for applications such as future behavior prediction.

generate more realistic sequences.

Video Results. Due to space constraints, we cannot include all of our qualitative results in an image format. Furthermore, because images are static, they cannot capture the full content of our generated sequences. Thus, we include more video results in the following link: `https://www.youtube.com/watch?v=wRRRl4QsUQg`.

4.7. Conclusions

In this work, we introduced a model that uses a single first-person image to generate an egocentric basketball motion sequence in the form of a 12D camera configuration trajectory. We showed that our model generates realistic sequences that capture the goals of real players. Furthermore, we demonstrated that our model can be learned directly from the first-person video data, which is beneficial as obtaining labeled behavioral data is costly.

Our model could be used for a wide array of behavioral applications, such as future behavior prediction or player development, for which we could build models using the data of expert players and then transfer their behavioral patterns to less-skilled players. Furthermore, even though we apply our model on a basketball activity, we do not inject any basketball-specific domain knowledge into the model. Thus, we believe that our model is general enough to also be applied to other activities, which we will explore in our future work. In the future,

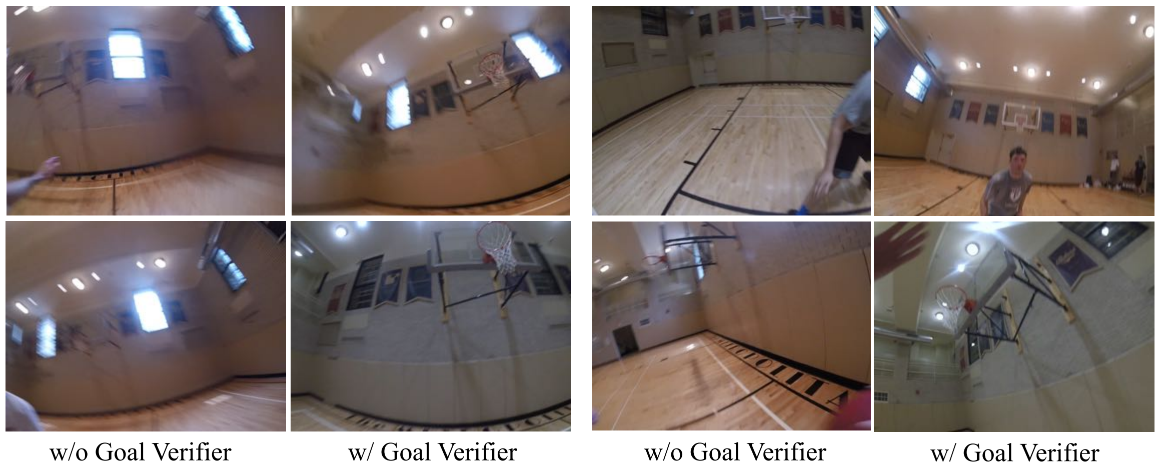


Figure 64: A figure comparing the final images of our generated sequences in two settings: **without** and **with** using the goal verifier network. We retrieve these images via nearest neighbor search. Note that the images produced with a goal verifier "focus" on the basket, just as real players would, thus capturing the goals of real players more accurately.

we would also like to experiment with more complex configurations that would allow us to represent even more complicated behavioral patterns. Finally, we believe that the concept behind our model could be used for various robotic applications, where our model could provide behavioral guidelines for robots.

BIBLIOGRAPHY

[1] Bertasius, G., Shi, J., Torresani, L.: Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2015)

[2] Bertasius, G., Shi, J., Torresani, L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In: The IEEE International Conference on Computer Vision (ICCV). (December 2015)

[3] Bertasius, G., Shi, J., Torresani, L.: Semantic segmentation with boundary neural fields. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[4] Bertasius, G., Torresani, L., Yu, S.X., Shi, J.: Convolutional random walk networks for semantic image segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (July 2017)

[5] Bertasius, G., Torresani, L., Shi, J.: Object detection in video with spatiotemporal sampling networks. In: ECCV (12). Volume 11216 of Lecture Notes in Computer Science., Springer (2018) 342–357

[6] Bertasius, G., Feichtenhofer, C., Tran, D., Shi, J., Torresani, L.: Learning discriminative motion features through detection. CoRR **abs/1812.04172** (2018)

[7] Bertasius, G., Park, H.S., Yu, S.X., Shi, J.: First-person action-object detection with egonet. In: Proceedings of Robotics: Science and Systems. (July 2017)

[8] Bertasius, G., Park, H.S., Yu, S.X., Shi, J.: Unsupervised learning of important objects from first-person videos. In: The IEEE International Conference on Computer Vision (ICCV). (October 2017)

[9] Bertasius, G., Park, H.S., Yu, S.X., Shi, J.: Am i a baller? basketball performance assessment from first-person videos. In: The IEEE International Conference on Computer Vision (ICCV). (October 2017)

[10] Bertasius, G., Chan, A., Shi, J.: Egocentric basketball motion planning from a single first-person image. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2018)

[11] Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. (June 2015)

[12] Ganin, Y., Lempitsky, V.S.: N^4-fields: Neural network nearest neighbor fields for image transforms. ACCV (2014)

[13] Kivinen, J.J., Williams, C.K., Heess, N., Technologies, D.: Visual boundary prediction: A deep neural prediction network and quality dissection. AISTATS **1**(2) (2014) 9

[14] Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV). (2011)

[15] Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. PAMI (2015)

[16] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)

[17] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3) (2015) 211–252

[18] Zhu, X., Wang, Y., Dai, J., Yuan, L., Wei, Y.: Flow-guided feature aggregation for video object detection. In: International Conference on Computer Vision (ICCV). (2017)

[19] Feichtenhofer, C., Pinz, A., Zisserman, A.: Detect to track and track to detect. In: International Conference on Computer Vision (ICCV). (2017)

[20] Han, W., Khorrami, P., Paine, T.L., Ramachandran, P., Babaeizadeh, M., Shi, H., Li, J., Yan, S., Huang, T.S.: Seq-nms for video object detection. CoRR **abs/1602.08465** (2016)

[21] Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M., Tran, D.: Detect-and-Track: Efficient Pose Estimation in Videos. In: CVPR. (2018)

[22] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Val Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV. (2016)

[23] Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. European Conference on Computer Vision (2018)

[24] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018. (2018)

[25] Park, H.S., Shi, J.: Social saliency prediction. In: CVPR. (2015)

[26] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., Weinberger, K.,

eds.: Advances in Neural Information Processing Systems 25. Curran Associates, Inc. (2012) 1097–1105

[27] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CVPR (to appear) (November 2015)

[28] Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., eds.: Advances in Neural Information Processing Systems 24. Curran Associates, Inc. (2011) 109–117

[29] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. CoRR **abs/1412.7062** (2014)

[30] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5) (May 2011) 898–916

[31] Borenstein, E.: Combining top-down and bottom-up segmentation. In: In Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR. (2004) 46

[32] Shotton, J.: Contour-based learning for object detection. In: In Proc. ICCV. (2005) 503–510

[33] Opelt, A., Zisserman, A.: A boundary-fragment-model for object detection. In: In ECCV. (2006) 575–588

[34] Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2014)

[35] Pinheiro, P.H.O., Collobert, R.: Recurrent convolutional neural networks for scene parsing. CoRR **abs/1306.2795** (2013)

[36] Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. CoRR **abs/1312.4659** (2013)

[37] Lim, J., Zitnick, C.L., Dollár, P.: Sketch tokens: A learned mid-level representation for contour and object detection. In: CVPR. (2013)

[38] Arbelaez, P., Pont-Tuset, J., Barron, J., Marqués, F., Malik, J.: Multiscale combinatorial grouping. In: Computer Vision and Pattern Recognition (CVPR). (2014)

[39] Ren, X., Bo, L.: Discriminatively Trained Sparse Code Gradients for Contour Detection. In: Advances in Neural Information Processing Systems. (December 2012)

[40] Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2** (1998) 121–167

[41] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (1997) 888–905

[42] Isola, P., Zoran, D., Krishnan, D., Adelson, E.H.: Crisp boundary detection using pointwise mutual information. In: ECCV. (2014)

[43] Maire, M., Yu, S.X., Perona, P.: Reconstructive sparse code transfer for contour detection and semantic labeling. In: Asian Conference on Computer Vision (ACCV). (2014)

[44] Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6) (June 1986) 679–698

[45] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge (2014)

[46] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014)

[47] Zhang, N., Paluri, M., Ranzato, M., Darrell, T., Bourdev, L.: Panda: Pose aligned networks for deep attribute modeling. CoRR **abs/1311.5591** (2013)

[48] Karayev, S., Hertzmann, A., Winnemoeller, H., Agarwala, A., Darrell, T.: Recognizing image style. CoRR **abs/1311.3715** (2013)

[49] Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-svms for object detection and beyond. In: ICCV. (2011)

[50] Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision. Volume 2. (July 2001) 416–423

[51] Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. Int. J. Comput. Vision **43**(1) (June 2001) 7–27

[52] Ren, X., Fowlkes, C.C., Malik, J.: Scale-invariant contour completion using condition random fields. Technical report

[53] Iandola, F.N., Moskewicz, M.W., Karayev, S., Girshick, R.B., Darrell, T., Keutzer, K.: Densenet: Implementing efficient convnet descriptor pyramids. CoRR **abs/1404.1869** (2014)

[54] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR **abs/1312.6229** (2013)

[55] Hariharan, B., Arbeláez, P.A., Girshick, R.B., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. CoRR **abs/1411.5752** (2014)

[56] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR **abs/1411.4038** (2014)

[57] Hsieh, P.J., Vul, E., Kanwisher, N.: Recognition alters the spatial pattern of fmri activation in early retinotopic cortex. Journal of Neurophysiology **103**(3) (2010) 1501–1507

[58] Sanguinetti, J.L., Allen, J.J., Peterson, M.A.: The ground side of an object perceived as shapeless yet processed for semantics. Psychological science (2013) 0956797613502814

[59] Kourtzi, Z., Kanwisher, N.: Representation of perceived object shape by the human lateral occipital complex. Science **293** (2001) 1506–1509

[60] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)

[61] Zitnick, C.L., Dollár, P.: Edge boxes: Locating object proposals from edges. In: ECCV. (2014)

[62] Sironi, A., Lepetit, V., Fua, P.: Multiscale centerline detection by learning a scale-space distance transform. (June 2014)

[63] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. CoRR **abs/1310.1531** (2013)

[64] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

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

[66] Xie, S., Tu, Z.: Holistically-nested edge detection. In: The IEEE International Conference on Computer Vision (ICCV). (December 2015)

[67] Mostajabi, M., Yadollahpour, P., Shakhnarovich, G.: Feedforward semantic segmentation with zoom-out features. CoRR **abs/1412.0774** (2014)

[68] Carreira, J.a., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: Proceedings of the 12th European Conference on Computer Vision - Volume Part VII. ECCV'12, Berlin, Heidelberg, Springer-Verlag (2012) 430–443

[69] Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2014)

[70] Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Simultaneous detection and segmentation. In: European Conference on Computer Vision (ECCV). (2014)

[71] Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Transactions on Pattern Analysis and Machine Intelligence (August 2013)

[72] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. In: International Conference on Computer Vision (ICCV). (2015)

[73] Dai, J., He, K., Sun, J.: Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In: The IEEE International Conference on Computer Vision (ICCV). (December 2015)

[74] Hong, S., Noh, H., Han, B.: Decoupled deep neural network for semi-supervised semantic segmentation. In: NIPS). (December 2015)

[75] Lin, G., Shen, C., Reid, I.D., van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. CoRR **abs/1504.01013** (2015)

[76] Tappen, M.F., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In: Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2. ICCV '03, Washington, DC, USA, IEEE Computer Society (2003) 900–

[77] Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11) (November 2001) 1222–1239

[78] Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16. MIT Press (2004) 321–328

[79] Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA (1994)

[80] Wainwright, M., Jaakkola, T., Willsky, A.: Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. IEEE Transactions on Information Theory **51** (2002) 3697–3717

[81] Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary mrfs via extended roof duality. In: Proc Comp. Vision Pattern Recogn. (CVPR). (June 2007)

[82] Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: ICCV. (2015)

[83] Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Computer Vision (ICCV), 2015 IEEE International Conference on. (2015)

[84] Chen, L., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. CVPR (2016)

[85] Lovasz, L.: Random walks on graphs: A survey (1993)

[86] Bahmani, B., Chowdhury, A., Goel, A.: Fast incremental and personalized pagerank. Proc. VLDB Endow. **4**(3) (December 2010) 173–184

[87] Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (November 1999) Previous number = SIDL-WP-1999-0120.

[88] Chen, L., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. CVPR (2016)

[89] Qi, X., Shi, J., Liu, S., Liao, R., Jia, J.: Semantic segmentation with object clique potential. In: The IEEE International Conference on Computer Vision (ICCV). (December 2015)

[90] Kokkinos, I.: Surpassing humans in boundary detection using deep learning. CoRR **abs/1511.07386** (2015)

[91] Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2001) 282–289

[92] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[93] Gould, S., Fulton, R., Koller, D.: Decomposing a scene into geometric and semantically consistent regions. In: Proceedings of the International Conference on Computer Vision (ICCV). (2009)

[94] Liu, C., Yuen, J., Torralba, A. In: Nonparametric Scene Parsing via Label Transfer. Springer International Publishing, Cham (2016) 207–236

[95] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR). (2015)

[96] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 770–778

[97] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. (2017)

[98] He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., eds.: Computer Vision – ECCV 2014. (2014)

[99] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV). (2017)

[100] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal Loss for Dense Object Detection. In: Proceedings of the International Conference on Computer Vision (ICCV). (2017)

[101] Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Neural Information Processing Systems (NIPS). (2015)

[102] Girshick, R.: Fast R-CNN. In: Proceedings of the International Conference on Computer Vision (ICCV). (2015)

[103] Gupta, S., Girshick, R., Arbelaez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: ECCV. (2014)

[104] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV. (2016)

[105] Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc. (2016) 379–387

[106] Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR (2015)

[107] Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. (2017) 6517–6525

[108] Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., Ouyang, W.: T-CNN: tubelets with convolutional neural networks for object detection from videos. IEEE TCSVT 2017 (2017)

[109] Kang, K., Ouyang, W., Li, H., Wang, X.: Object detection from video tubelets with convolutional neural networks. CoRR **abs/1604.04053** (2016)

[110] Lee, B., Erdenee, E., Jin, S., Rhee, P.: Multi-class multi-object tracking using changing point detection. CoRR **abs/1608.08434** (2016)

[111] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). Volume 00. (Oct. 2017) 764–773

[112] Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. CoRR **abs/1512.01274** (2015)

[113] Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition. In: CVPR. (2017)

[114] Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. (2016) 779–788

[115] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: CVPR. (2017)

[116] Wei, S.E., Ramakrishna, V., Kanade, T., Sheikh, Y.: Convolutional pose machines. In: CVPR. (2016)

[117] Peng, X., Schmid, C.: Multi-region two-stream R-CNN for action detection. In: ECCV 2016 - European Conference on Computer Vision, Amsterdam, Netherlands (October 2016)

[118] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 27. Curran Associates, Inc. (2014) 568–576

[119] Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)

[120] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the 2015 IEEE Interna-

tional Conference on Computer Vision (ICCV). ICCV '15, Washington, DC, USA, IEEE Computer Society (2015) 4489–4497

[121] Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the kinetics dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. (2017) 4724–4733

[122] Huang, D.A., Ramanathan, V., Mahajan, D., Torresani, L., Paluri, M., Fei-Fei, L., Niebles, J.C.: What makes a video a video : Analyzing temporal information in video understanding models and datasets. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, USA, 2018. (2018)

[123] Xiao, F., Lee, Y.J.: Video object detection with an aligned spatial-temporal memory. In: European Conference on Computer Vision (ECCV). (2018)

[124] Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. (2016)

[125] Bilen, H., Fernando, B., Gavves, E., Vedaldi, A.: Action recognition with dynamic image networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(12) (2017) 2799–2813

[126] Girdhar, R., Ramanan, D.: Attentional pooling for action recognition. In: NIPS. (2017)

[127] Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: Advances in Neural Information Processing Systems (NIPS). (2016) 3468–3476

[128] Feichtenhofer, C., Pinz, A., Wildes, R.P.: Spatiotemporal multiplier networks for video action recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2017)

[129] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Gool, L.V.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV (8). Volume 9912 of Lecture Notes in Computer Science., Springer (2016) 20–36

[130] Ng, J.Y.H., Hausknecht, M.J., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR, IEEE Computer Society (2015) 4694–4702

[131] Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV. (2018) 318–335

[132] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 1800–1807

[133] Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: ICCV. (2017)

[134] Soomro, K., Zamir, A.R., Shah, M., Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. CoRR (2012)

[135] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: Proceedings of the International Conference on Computer Vision (ICCV). (2011)

[136] Wang, X., Farhadi, A., Gupta, A.: Actions ˜ transformations. In: CVPR. (2016)

[137] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. (2017)

[138] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR **abs/1311.2901** (2013)

[139] Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T.J., Lipson, H.: Understanding neural networks through deep visualization. CoRR **abs/1506.06579** (2015)

[140] Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV. (2018) 139–156

[141] Iqbal, U., Milan, A., Gall, J.: Posetrack: Joint multi-person pose estimation and tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)

[142] Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, IEEE (December 2013) 3192–3199

[143] Gu, C., Sun, C., Vijayanarasimhan, S., Pantofaru, C., Ross, D.A., Toderici, G., Li, Y., Ricco, S., Sukthankar, R., Schmid, C., Malik, J.: AVA: A video dataset of spatio-temporally localized atomic visual actions. arXiv preprint arXiv:1705.08421 (2017)

[144] Sun, C., Shrivastava, A., Vondrick, C., Murphy, K., Sukthankar, R., Schmid, C.: Actor-centric relation network. In: ECCV (11). Volume 11215 of Lecture Notes in Computer Science., Springer (2018) 335–351

[145] Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: A better baseline for AVA. CoRR **abs/1807.10066** (2018)

[146] Li, Y., Li, Y., Vasconcelos, N.: Resound: Towards action recognition without representation bias. In: The European Conference on Computer Vision (ECCV). (September 2018)

[147] Johansson, R.S., Westling1, G., Bäckström, A., Flanagan, J.R.: Eye hand coordination in object manipulation. Journal of Neuroscience (2001)

[148] Perone, S., Madole, K.L., Ross-Sheehy, S., Carey, M., Oakes, L.M.: The relation between infants' activity with objects and attention to object appearance. Developmental Psychology (2008)

[149] Vidoni, E.D., McCarley, J.S., Edwards, J.D., Boyd, L.A.: Manual and oculomotor performance develop contemporaneously but independently during continuous tracking. Experimental Brain Research (2009)

[150] Bowman, M.C., Johannson, R.S., Flanagan, J.R.: Eye hand coordination in a sequential target contact task. Experimental Brain Research (2009)

[151] Lazzari, S., Mottet, D., Vercher, J.L.: Eye hand coordination in rhythmical pointing. Journal of Motor Behavior (2009)

[152] Li, Y., Ye, Z., Rehg, J.M.: Delving into egocentric actions. In: CVPR

[153] Ren, X., Gu, C.: Figure-ground segmentation improves handled object recognition in egocentric video. In: CVPR. (2010)

[154] Minghuang Ma, K.K.: Going deeper into first-person activity recognition. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2016)

[155] Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., Rehg, J.M.: A scalable approach to activity recognition based on object use. In: ICCV. (2007)

[156] Yao, B., Fei-Fei, L.: Grouplet: A structured image representation for recognizing human and object interactions. In: CVPR. (2010)

[157] Yao, B., Fei-Fei, L.: Modeling mutual context of object and human pose in human object interaction activities. In: CVPR. (2010)

[158] Delaitre, V., Sivic, J., Laptev, I.: Learning person-object interactions for action recognition in still images. In: NIPS. (2011)

[159] Turek, M.W., Hoogs, A., Collins, R.: Unsupervised learning of functional categories in video scenes. In: ECCV. (2010)

[160] Gall, J., Fossati, A., Van Gool, L.: Functional categorization of objects using real-time markerless motion. In: CVPR. (2011)

[161] Kjellstrom, H., Romero, J., Kragic, D.: Visual object action recognition: Inferring object affordances from human demonstration. CVIU (2011)

[162] Gupta, A., Davis, L.S.: Objects in action: An approach for combining action understanding and object perception. In: CVPR. (2007)

[163] Gupta, A., Satkin, S., Efros, A.A., Hebert, M.: From 3d scene geometry to human workspace. In: CVPR. (2011)

[164] Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: ICCV. (2007)

[165] Fouhey, D.F., Delaitre, V., Gupta, A., Efros, A.A., Laptev, I., Sivic, J.: People watching: Human actions as a cue for single-view geometry. In: ECCV. (2012)

[166] Yu, L.F., Duncan, N., Yeung, S.K.: Fill and transfer: A simple physics-based approach for containability reasoning. In: ICCV. (2015)

[167] Gori, I., Aggarwal, J.K., Ryoo, M.S.: Building unified human descriptors for multi-type activity recognition. CoRR **abs/1507.02558** (2015)

[168] Fathi, A., Ren, X., Rehg, J.M.: Learning to recognize objects in egocentric activities. In: CVPR, IEEE Computer Society (2011) 3281–3288

[169] Pirsiavash, H., Ramanan, D.: Detecting activities of daily living in first-person camera views. In: CVPR. (2012)

[170] Fathi, A., Farhadi, A., Rehg, J.M.: Understanding egocentric activities. In: ICCV

[171] Ryoo, M.S., Fuchs, T.J., Xia, L., Aggarwal, J., Matthies, L.: Robot-centric activity prediction from first-person videos: What will they do to me? In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction. HRI '15, New York, NY, USA, ACM (2015) 295–302

[172] Lee, Y.J., Grauman, K.: Predicting important objects for egocentric video summarization. IJCV (2015)

[173] Lu, Z., Grauman, K.: Story-driven summarization for egocentric video. In: CVPR. (2013)

[174] Li, Y., Fathi, A., Rehg, J.M.: Learning to predict gaze in egocentric video. In: ICCV

[175] Cai, M., Kitani, K.M., Sato, Y.: Understanding hand-object manipulation with grasp types and object attributes. In: Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016. (2016)

[176] Krüger, N., Geib, C.W., Piater, J.H., Petrick, R.P.A., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrcen, D., Agostini, A., Dillmann, R.: Object-

action complexes: Grounded abstractions of sensory-motor processes. Robotics and Autonomous Systems **59**(10) (2011) 740–757

[177] Bertasius, G., Park, H.S., Shi, J.: Exploiting egocentric object prior for 3d saliency detection. arXiv:1511.02682 (2015)

[178] Rother, C., Kolmogorov, V., Blake, A.: "grabcut": Interactive foreground extraction using iterated graph cuts. ToG (SIGGRAPH) (2004)

[179] Judd, T., Ehinger, K., Durand, F., Torralba, A.: Learning to predict where humans look. In: IEEE International Conference on Computer Vision (ICCV). (2009)

[180] Harel, J., Koch, C., Perona, P.: Graph-based visual saliency. In: NIPS. (2007)

[181] Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR. (2014)

[182] Li, Y., Hou, X., Koch, C., Rehg, J., Yuille, A.: The secrets of salient object segmentation. In: CVPR. (2014)

[183] Pinheiro, P.H.O., Collobert, R.: From image-level to pixel-level labeling with convolutional networks. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2015)

[184] Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2016)

[185] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. (2009)

[186] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision (ECCV), Zürich (September 2014)

[187] Bertasius, G., Park, H.S., Yu, S.X., Shi, J.: First-person action-object detection with egonet. In: Proceedings of Robotics: Science and Systems. (July 2017)

[188] Li, Y., Paluri, M., Rehg, J.M., Dollar, P.: Unsupervised learning of edges. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[189] Bearman, A., Russakovsky, O., Ferrari, V., Fei-Fei, L.: What's the Point: Semantic Segmentation with Point Supervision. ECCV (2016)

[190] Papandreou, G., Chen, L.C., Murphy, K., Yuille, A.L.: Weakly- and semi-supervised learning of a dcnn for semantic image segmentation. arxiv:1502.02734 (2015)

[191] Xu, J., Schwing, A.G., Urtasun, R.: Learning to segment under various forms of weak supervision. In: Proc. CVPR. (2015)

[192] Lin, D., Dai, J., Jia, J., He, K., Sun, J.: Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[193] Pathak, D., Shelhamer, E., Long, J., Darrell, T.: Fully convolutional multi-class multiple instance learning. In: ICLR Workshop. (2015)

[194] Pathak, D., Krähenbühl, P., Darrell, T.: Constrained convolutional neural networks for weakly supervised segmentation. In: ICCV. (2015)

[195] Li, G., Yu, Y.: Deep contrast learning for salient object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016) 478–487

[196] : https://www.stats.com.

[197] Yue, Y., Lucey, P., Carr, P., Bialkowski, A., Matthews, I.: Learning fine-grained spatial models for dynamic sports play prediction. 2014 IEEE International Conference on Data Mining (ICDM) **00** (2014) 670–679

[198] Lucey, P., Bialkowski, A., Carr, P., Yue, Y., Matthews, I.: How to get an open shot: Analyzing team movement in basketball using tracking data. In: MITSSAC. (2014)

[199] : http://firstvision.tv/.

[200] Wei, X., Lucey, P., Morgan, S., Reid, M., Sridharan, S.: The thin edge of the wedge: Accurately predicting shot outcomes in tennis using style and context priors. In: MITSSAC. (2016)

[201] Le, H., Carr, P., Yue, Y., Matthews, I.: Data-driven ghosting using deep imitation learning. In: MITSSAC. (2014)

[202] Miller, A., Bornn, L.: Possession sketches: Mapping nba strategies. In: MITSSAC. (2016)

[203] Su, S., Hong, J.P., Shi, J., Park, H.S.: Predicting behaviors of basketball players from first person videos. In: CVPR. (2017)

[204] Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, ISBN: 0521540518 (2004)

[205] Lucey, P., Bialkowski, A., Carr, P., Morgan, S., Matthews, I., Sheikh, Y.: Representing and discovering adversarial team behaviors using player roles. 2013 IEEE Conference on Computer Vision and Pattern Recognition **00** (2013) 2706–2713

[206] Maksai, A., Wang, X., Fua, P.: What players do with the ball: A physically constrained interaction modeling. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

[207] Kim, K., Lee, D., Essa, I.: Detecting regions of interest in dynamic scenes with camera motions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society (June 2012)

[208] Swears, E., Hoogs, A., Ji, Q., Boyer, K.: Complex activity recognition using granger constrained dbn (gcdbn) in sports and surveillance video. (06 2014)

[209] Ibrahim, M.S., Muralidharan, S., Deng, Z., Vahdat, A., Mori, G.: A hierarchical deep temporal model for group activity recognition. In: Computer Vision and Pattern Recognition (CVPR). (2016)

[210] Ramanathan, V., Huang, J., Abu-El-Haija, S., Gorban, A., Murphy, K., Fei-Fei, L.: Detecting events and key actors in multi-person videos. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA (June 2016)

[211] Damen, D., Leelasawassuk, T., Haines, O., Calway, A., Mayol-Cuevas, W.: You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In: Proceedings of the British Machine Vision Conference, BMVA Press (2014)

[212] Park, H.S., Hwang, J.J., Niu, Y., Shi, J.: Egocentric future localization. In: CVPR. (2016)

[213] Fathi, A., Hodgins, J.K., Rehg, J.M.: Social interactions: A first-person perspective

[214] Park, H.S., Jain, E., Sheikh, Y.: 3d gaze concurrences from head-mounted cameras. In: NIPS. (2012)

[215] Kitani, K.M., Okabe, T., Sato, Y., Sugimoto, A.: Fast unsupervised ego-action learning for first-person sports videos. In: CVPR. (2011)

[216] Park, H.S., Hwang, J.J., Shi, J.: Force from motion: Decoding physical sensation from a first person video. In: CVPR. (2016)

[217] : http://nbasavant.com.

[218] : https://www.sloansportsconference.com.

[219] Singh, S., Arora, C., Jawahar, C.V.: First person action recognition using deep learned descriptors. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)

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

[221] Jain, L.C., Medsker, L.R.: Recurrent Neural Networks: Design and Applications. 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1999)

[222] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (November 1997) 1735–1780

[223] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27. (2014) 2672–2680

[224] Pentland, A., Liu, A.: Modeling and prediction of human behavior. Neural Comput. **11**(1) (January 1999) 229–242

[225] Liu, Y., Gupta, A., Abbeel, P., Levine, S.: Imitation from observation: Learning to imitate behaviors from raw video via context translation. CoRR **abs/1707.03374** (2017)

[226] Rahmatizadeh, R., Abolghasemi, P., Bölöni, L., Levine, S.: Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. CoRR **abs/1707.02920** (2017)

[227] Wu, C., Zhang, J., Selman, B., Savarese, S., Saxena, A.: Watch-bot: Unsupervised learning for reminding humans of forgotten actions. (2016)

[228] Ali, S., Shah, M. In: Floor Fields for Tracking in High Density Crowd Scenes. Springer Berlin Heidelberg, Berlin, Heidelberg (2008) 1–14

[229] Pellegrini, S., Ess, A., Schindler, K., Gool, L.J.V.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: ICCV, IEEE Computer Society (2009) 261–268

[230] Ma, W.C., Huang, D.A., Lee, N., Kitani, K.M.: Forecasting interactive dynamics of pedestrians with fictitious play. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (July 2017)

[231] Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51** (May 1995) 4282–4286

[232] Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: CVPR, IEEE Computer Society (2009) 935–942

[233] Alahi, A., Ramanathan, V., Li, F.F.: Socially-aware large-scale crowd forecasting. In: CVPR, IEEE Computer Society (2014) 2211–2218

[234] Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: CVPR. (2016)

[235] Lee, N., Kitani, K.M.: Predicting wide receiver trajectories in american football. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE (2016) 1–9

[236] Le, H., Carr, P., Yue, Y., Lucey, P.: Data-driven ghosting using deep imitation learning. In: MITSSAC. (2017)

[237] Zheng, S., Yue, Y., Hobbs, J.: Generating long-term trajectories using deep hierarchical networks. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29. Curran Associates, Inc. (2016) 1543–1551

[238] Soran, B., Farhadi, A., Shapiro, L. In: Action Recognition in the Presence of One Egocentric and Multiple Static Cameras. (2015)

[239] Su, Y.C., Grauman, K.: Detecting engagement in egocentric video. In: European Conference on Computer Vision (ECCV). (2016)

[240] Rhinehart, N., Kitani, K.M.: First-person activity forecasting with online inverse reinforcement learning. In: The IEEE International Conference on Computer Vision (ICCV). (Oct 2017)

[241] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR (2015)

[242] Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T.J., Lipson, H.: Understanding neural networks through deep visualization. CoRR **abs/1506.06579** (2015)

[243] Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: IEEE Conference on Computer Vision and Pattern Recognition. (2015)

[244] Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision (2016) 1–23

[245] Huttenlocher, D.P., Klanderman, G.A., Rucklidge, W.A.: Comparing images using the hausdorff distance. IEEE Trans. Pattern Anal. Mach. Intell. **15**(9) (September 1993) 850–863