Multimodal Learning with Minimal Human Supervision from Videos and Natural Language

By

FANYI XIAO

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Professor Yong Jae Lee, Chair

Professor Premkumar Devanbu

Professor Zhou Yu

Committee in Charge

2020

ABSTRACT

Multimodal Learning with Minimal Human Supervision from Videos and Natural Language

By

Fanyi Xiao

Doctor of Philosophy in Computer Science

University of California, Davis

Professor Yong Jae Lee, Chair

Humans perceive and interact with the surrounding world by processing information from many different sensory modalities (e.g., visual inputs, auditory signals, self-motion, haptics, smell, taste and language, etc). In this thesis, we believe it is promising to mimic humans to perform multimodal learning with our AI agents, in order to enable human-level visual perception capability. Specifically, we will present algorithms that learn from multimodal data like videos and natural language for visual understanding. Meanwhile, as multimodal data offers abundant opportunities to serve as supervision for training visual models, we will also present algorithms that can learn with either weak supervision or no supervision at all from multimodal data. We believe these are the first steps towards a more general and capable visual perception system.

To my family

CONTENTS

ACKNOWLEDGMENTS

Chapter 1

Introduction

Figure 1.1: **A baby learns to interact with an object via multiple sensory modalities.** Please visit `https://youtu.be/HCYnOFXX0RE` for the full video clip.

Humans perceive and interact with the surrounding world by processing information from many different sensory modalities – e.g., visual/auditory inputs, self-motion, haptics, smell, taste and language, etc. An prominent example of this is on how human babies learn to interact with objects in the world (shown in Fig. 1.1, which is one frame taken out from a video clip). First, the baby is looking [*visual*] at the object to localize it. Meanwhile, he is hearing [*auditory*] the sound of the bubbles bursting whenever he hits [*haptics*] them. Furthermore, he is using his arms to move the object around to get different views of the object [*self-motion*]. Finally, the baby puts the object into his mouth [*taste*] to get a taste of it (shown in the video clip). Therefore, to enable human-level visual perception capability, one plausible way is to mimic what a human

Figure 1.2: **Videos consist of multiple data modalities.** One can identify the action "blowing nose" from either the RGB frame, the sound or the motion presented in this video clip.

baby is doing here to perform multimodal learning with our AI agents.

As a prevalent multimodal (i.e., RGB, sound, motion illustrated in Fig. 1.2) data source, videos play an important role in the visual processing of human brains – we perceive the world by processing an input "video stream" captured by our eyes. In addition to the RGB pixels presented in each "frame", our brain also effectively utilizes rich information presented in other modalities like motion and audio. Therefore, in order to enable human-like visual perception capability (such that we can enable applications like robot navigation in a fresh environment), it is vital to develop algorithms that can effectively understand videos. Moreover, according to Cisco Visual Networking Index, videos constitute the largest chuck of Internet traffic – 75% of all Internet traffic in 2017 and predicted to grow to 82% by 2022. Therefore, models that can understand rich semantics (e.g., recognizing actions/activities, localizing humans/objects, inferring intents and predicting future, etc.) in videos will have huge practical impact.

Despite the great advances on visual perception in the past few years, the progress is in large propelled by the advent of large-scale *static-image* datasets like ImageNet for image classification, MS-COCO for instance segmentation and ADE20K for scene understanding. This steers significant research efforts towards visual understanding on images. However, as mentioned before, humans do not perceive the visual world by parsing a set of independent snapshots, and this is true in almost all scenarios in which we would like an AI agent navigate and interact with the environment using its visual capability. Thus, an image-based algorithm

would most likely deliver suboptimal results as 1) they ignore rich information like motion and audio completely; and 2) they suffer from a domain gap if trained on images and tested on videos (e.g., videos have specific challenges like motion blur which does not occur in images). With this motivation, one topic that will be heavily discussed in this thesis lies on the research direction of video understanding. Specifically, this thesis will present several state-of-the-art approaches for various tasks ranging from video segmentation [255] (utilizing the idea of *progressive learning* to iteratively expand/refine results spatially and temporally), video object detection [256] (temporal propagation via an aligned spatial-temporal memory) and audiovisual video recognition [258] (hierarchical audiovisual fusion with training dynamics matching).

While pushing the boundary of video understanding, a key issue arises with the current state-of-the-art video understanding paradigm – it requires a large amount of labeled data to train models. As the task becomes more finer-grained (i.e., classification to detection, which further requires localization), the cost of labeling such dataset increases tremendously (image-level class labels to bounding box labels). Moreover, the problem gets even worse as we'd like to get into the regime of multimodal learning and therefore might need to annotate data for various modalities. To mitigate this problem, as demonstrated in previous work [52, 64, 86, 166], we don't need tons of labeled data and could instead resort to weakly- or self-supervised learning to train our visual models. Fortunately, multimodal data (e.g. videos) offers abundant opportunities to provide supervisory signals for this. One prominent way is to exploit the "correspondence" between different modalities and treat each modality as one particular "view" of the same data[1]. For example, one can use the synchronization among RGB, audio and motion channels in videos to train a model to solve "pretext tasks" (e.g., predicting audio-visual [175] or RGB-motion [157] correspondences) – the underlying belief is that, in order to perform well on these tasks, the model has to understand the semantics of the data and therefore induce a useful representation. Unlike previous work which usually resort to ad-hoc fusion of different modalities using off-the-shelf network architectures, we developed an integrated audiovisual model with hierarchical fusion to learn self-supervised video representation by exploiting the synchronization of audio and visual signals.

[1]In fact, the term "self-supervised learning" is proposed in cognitive science literatures to specifically refer to this learning paradigm.

Of course, the cross-modal correspondence is not the only thing that one can utilize for self-supervision. For example, we discover that we can learn a disentangled identity/pose representation for objects, with which we can recombine them in arbitrary ways to generate novel images (i.e., given object A and B, we can generate a new object with A's identity and B's pose) [259]. Unlike previous approaches that tackle this problem using complicated but often brittle cyclic-constraints [49, 116], we propose to make use of *unlabeled videos* to automatically generate training input/output pairs, which then allows us to train our model without any identity/pose labels. The key observation is that videos link together different views of the same object and thus provide pairs of images sharing the same object identity but with different poses. Our results show that by harvesting supervision from unlabeled videos, our method is much more robust compared to previous cyclic-constraints based methods that only rely on static images for training.

In addition to videos, it is also beneficial to learn from various other data modalities in combination with visual inputs. Among them, natural language is an important modality for many distinct characteristics. In contrast to other modalities like audio and motion, natural language is a pure human construct – for example, a sentence is composed of a hierarchy of words and phrases, which are artificially created objects of high-level semantics and rich structures. Therefore, we propose to make use of the linguistic structure presented in image captions to visually ground (i.e., localize) arbitrary linguistic phrases (in the form of spatial attention masks) in a weakly-supervised manner.

Next, I will briefly summarize the algorithms we developed for each of these problems and the main contributions.

1.1 Unsupervised Video Object Proposals

We present an unsupervised approach that generates a diverse, ranked set of bounding boxes and segmentation video object proposals (spatio-temporal tubes that localize the foreground objects) in an unannotated video. In contrast to previous unsupervised methods that either track regions initialized in an arbitrary frame or train a fixed model over a cluster of regions, we instead discover a set of easy-to-group instances of an object and then iteratively update its appearance

model to gradually detect harder instances in temporally-adjacent frames. Our method first generates a set of spatio-temporal bounding box proposals, and then refines them to obtain pixel-wise segmentation proposals.

Main contributions: We propose the first video object proposal algorithm that exploits the idea of iterative growing from easy frames to harder ones in a self-paced way. We demonstrate state-of-the-art (at the time of publication) segmentation results on the SegTrack v2 dataset, and bounding box tracking results that perform competitively to the supervised tracking methods.

1.2 Aligned Memory for Video Object Detection

We introduce Spatial-Temporal Memory Networks (STMN) for video object detection. At its core, a novel Spatial-Temporal Memory module (STMM) serves as the recurrent computation unit to model long-term temporal appearance and motion dynamics. The STMM's design enables full integration of pretrained backbone CNN weights, which we find to be critical for accurate detection. Furthermore, in order to tackle object motion in videos, we propose a novel MatchTrans module to align the spatial-temporal memory from frame to frame.

Main contributions: We propose a novel spatial-temporal memory network (STMN) for video object detection. Our main contributions include a carefully-designed recurrent computation unit that integrates pre-trained image classification weights into the memory and an in-network alignment module that spatially-aligns the memory across time. Together, these lead to the state-of-the-art video object detection results on the ImageNet VID benchmark.

1.3 Integrated Audiovisual Learning for Video Recognition

We present Audiovisual SlowFast Networks, an architecture for integrated audiovisual perception. Based on SlowFast Networks [62], AVSlowFast has Slow and Fast visual pathways that are deeply integrated with a Faster Audio pathway to model vision and sound in a unified representation. We fuse audio and visual features at multiple layers, enabling audio to contribute to the formation of hierarchical audiovisual concepts. To overcome training difficulties that arise from different learning dynamics for audio and visual modalities, we introduce DropPathway, which randomly drops the Audio pathway during training as an effective regularization technique. Inspired by

prior studies in neuroscience, we perform hierarchical audiovisual synchronization to learn joint audiovisual features. We report state-of-the-art results on multiple video action classification and detection datasets. Furthermore, we generalize AVSlowFast to self-supervised learning and demonstrate that our model can learn state-of-the-art video representations without human labels.

Main contributions: We propose the first integrated audiovisual modeling approach with hierarchical fusion between the audio and visual modalities. We propose several training techniques like DropPathway and hierarchical audiovisual synchronization to improve our results. These lead to state-of-the-art results for video recognition and action detection tasks. Finally, we demonstrate that AVSlowFast can be further extended to learn state-of-the-art self-supervised video representations.

1.4 Disentangle Visual Factors using Unlabeled Videos

We propose a novel approach that disentangles the identity and pose of objects for image generation. Our model takes as input an ID image and a pose image, and generates an output image with identity of the ID image and pose of the pose image. Unlike previous unsupervised work which only makes use of images, we instead leverage unlabeled videos to automatically construct pseudo ground-truth targets. This allows our network to train with direct supervision without any annotations. To further encourage disentanglement, we propose a novel disentanglement loss, and to improve realism, we propose a pixel-verification loss in which the generated image's pixels must trace back to the ID input.

Main contributions: To our knowledge, this is the first work exploiting unlabeled videos to learn to disentangle the identity and pose of input images. Compared to previous unsupervised approaches that typically employ cyclic-constraints, our approach demonstrates better disentanglement and higher generation quality on datasets of car, bus, and truck, thanks to the free supervision obtained from videos.

1.5 Transferring Linguistic Structures into Visual Domains

We propose a weakly-supervised approach that takes image-sentence pairs as input and learns to visually ground (i.e., localize) arbitrary linguistic phrases, in the form of spatial attention

masks. Specifically, the model is trained with images and their associated image-level captions, without any explicit region-to-phrase correspondence annotations. To this end, we introduce an end-to-end model which learns visual groundings of phrases with two types of carefully designed loss functions. In addition to the standard discriminative loss, which enforces that attended image regions and phrases are consistently encoded, we propose a novel *structural loss* which makes use of the parse tree structures induced by the sentences. In particular, we ensure complementarity among the attention masks that correspond to sibling noun phrases, and compositionality of attention masks among the children and parent phrases, as defined by the sentence parse tree.

Main contributions: We demonstrate that it is feasible and beneficial to transfer the rich structures in language into the visual domain. To achieve this, we propose two novel structural losses for the task of grounding arbitrary phrases/sentences in images and demonstrate its effectiveness through careful experimental studies.

Chapter 2

An Iterative Approach for Unsupervised Video Object Proposals

Generating object proposals – a set of candidate object-like regions in an image that may contain the object-of-interest – from *static images* has been extensively studied in recent years [3, 8, 134, 184, 236, 283]. The success of these methods now serves as a keystone to many state-of-the-art object detection [77, 90, 226] and semantic segmentation [41, 89] algorithms. Object proposals are beneficial in two main aspects: (1) *Computation*: compared to sliding window detection, they greatly reduce the number of regions in an image that must be considered (from potentially millions to thousands); and (2) *Recognition accuracy*: they tend to reduce non-object regions that would otherwise result in false-positive detections [98].

Compared with static images, video provides rich spatio-temporal information that can greatly benefit learning algorithms. Object proposals are equally, if not more, needed in the video domain since the number of regions in a video is much larger than that of a single image. Furthermore, many video applications including summarization, activity recognition, and retrieval would benefit tremendously from a robust video object proposals method that can reduce the complexity of a video by focusing on its main objects. For example, by reducing a long video down to a small set of spatio-temporal tubes consisting of the main objects, more accurate video summaries could be produced.

Existing approaches for video object proposals [65, 147, 172, 180, 203] (and more generally, video object segmentation [22, 23, 47, 83, 204, 237]) employ bottom-up or learned top-down

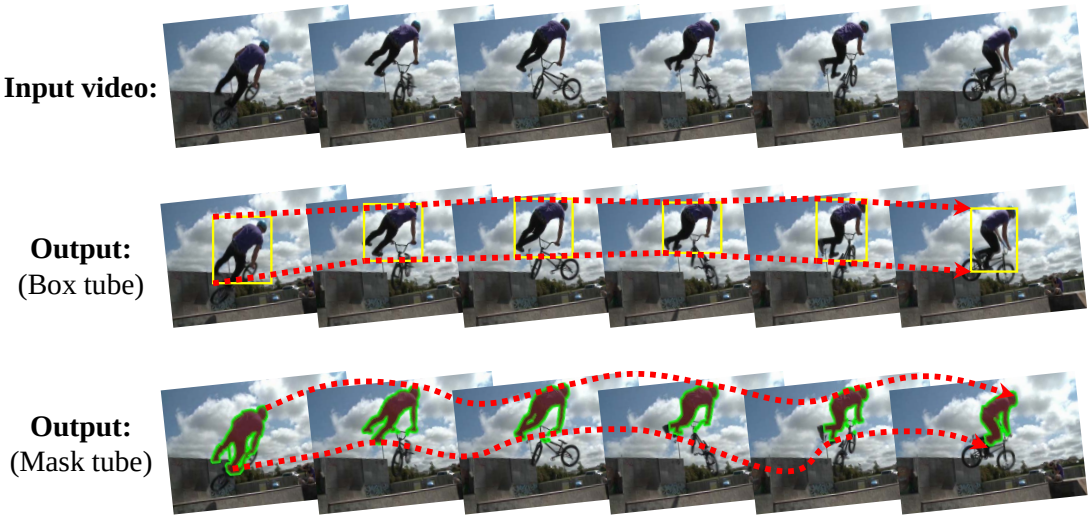

Figure 2.1: Given an unannotated video, our algorithm produces a set of spatio-temporal bounding box proposals and segmentation proposals that localize the foreground objects. Here we show one proposal (out of multiple) for each type.

appearance and motion cues to group pixels into spatio-temporal tubes that may belong to the same object. However, most methods either attempt to group pixels from all frames [47, 83, 262] or track regions that are initialized from arbitrary frames (e.g., the 1st frame) [22, 24, 148]. Consequently, these methods are susceptible to well-known challenges associated with clustering and tracking (model selection and computational complexity for the former, and initialization, drifting, and occlusion for the latter).

Inspired by the *key-segments* approach of [147], we instead sample a group of *easy* instances for each candidate object in the video to initialize an appearance model, and use that model to detect other "harder" instances of the object in the remaining frames. An object's easy instances are the regions that are object-like in appearance and have distinct motion against their immediate surrounding background. These regions are likely to span a single object, and therefore exhibit more appearance regularity, making them easier to group.

However, unlike [147], our key idea is to *iteratively* discover the harder instances in adjacent frames and update the object's appearance model in a self-paced manner, which allows the learned model to adapt and be more robust to (potentially) large appearance variations of the object. Furthermore, we work with bounding boxes, and only generate pixel-level segmentations conditioned on the detected boxes once all frames have been covered. Operating on bounding

boxes substantially reduces computational complexity, since individual pixel predictions can be avoided. Finally, we show how to explicitly enforce the initial easy instances to be temporally spread-out across the video. This helps to reduce drifting, since any new frame in which the object needs to be detected is likely to be temporally-close to at least one of the initial easy instances. It also has the additional benefit to help focus on the main foreground objects that consistently appear throughout the video, and give less emphasis to background objects that appear over only a short period of time.

Contributions. Our main contribution is a novel unsupervised algorithm that generates a set of spatio-temporal video object proposal boxes and segmentations (see Fig. 2.1). By discovering an object's easy instances first, and gradually detecting harder instances in temporally-adjacent frames, our algorithm effectively adapts to the object's changing appearance over time. We conduct experiments to evaluate our spatio-temporal bounding box and segmentation proposals. To evaluate our bounding box proposals, we compare with existing tracking algorithms, which require human annotation in the first frame. We demonstrate competitive results on the Visual Tracker benchmark [253], even though we do not use any human supervision. To evaluate our segmentation proposals, we compare with existing video segmentation algorithms and show state-of-the-art results on the SegTrack-v2 [148] dataset.

2.1 Related Work

Video segmentation. Previous video segmentation algorithms can be roughly categorized into three types. The first includes methods that cluster pixels using appearance and optical flow-based motion information across all frames [47, 83, 263]. The second clusters long-range point trajectories [23, 24, 167, 168, 178, 190], which tend to handle rigid objects better. The main limitation of these approaches is their lack of an explicit notion of *object appearance*; i.e., with only low-level bottom-up information, they tend to oversegment objects. We instead discover a small but diverse set of easy instances that likely belong to the same object, and use them as top-down supervision to detect the harder instances in the remaining frames.

The third type of methods, which is closest to our approach, compute segments on each frame and link them together through bottom-up or learned appearance matching and optical

flow [14, 22, 65, 69, 147, 172, 180, 246]. The key difference is that we first discover a set of easy instances of an object to build an initial model, and then we *iteratively* refine the model while simultaneously discovering harder instances in temporally-adjacent frames. For this, we leverage the fact that the object's appearance will be smoothly-varying in time. As more and more instances are discovered, the model becomes more robust to the object's changing appearance, and allows us to link together the object's instances that might otherwise be difficult to group due to large appearance variations of the object. While the method in [148] also iteratively refines a model to track an object over the video, the model is initialized with regions from the first frame, so can be more susceptible to drifting. Finally, unlike [65, 134, 172, 184], which *learn* to propose objects either in videos or static images, our approach does not require any labeled video/image to train and instead adapts to each unknown object on a per-video basis.

Tracking. Tracking algorithms (e.g., [71, 88, 225, 253, 275]) share our goal of localizing the same object over the video, but require human-annotation as initialization in the first-frame. In particular, the tracking approaches of [103, 225] use the framework of self-paced learning [17, 139] to carefully choose which frames to learn and update a tracking model. We also iteratively update our model and detections after initializing with the easy instances. However, unlike [103, 225], we do not require any human supervision. Moreover, in addition to bounding boxes, we output pixel-level segmentation masks.

2.2 Approach

We are given an unlabeled video $V=\{f_1, \ldots, f_N\}$ with N frames, and our goal is to discover the main objects in every frame that they appear, without knowing their categories a priori. To this end, we propose to generate a set of video object proposals – spatio-temporal tubes that track objects that have salient appearance and motion, and appear frequently throughout the video.

Our approach consists of three main steps: *initialization*, *iterative growing*, and *pixel-wise segmentation*. In the initialization step, we discover and rank a set of clusters; each cluster contains easy bounding box instances of an object in the video that have salient *object-like appearance* and *motion*. During iterative growing, we iteratively grow each cluster to detect harder instances of the corresponding object throughout the entire video. Finally, conditioned on

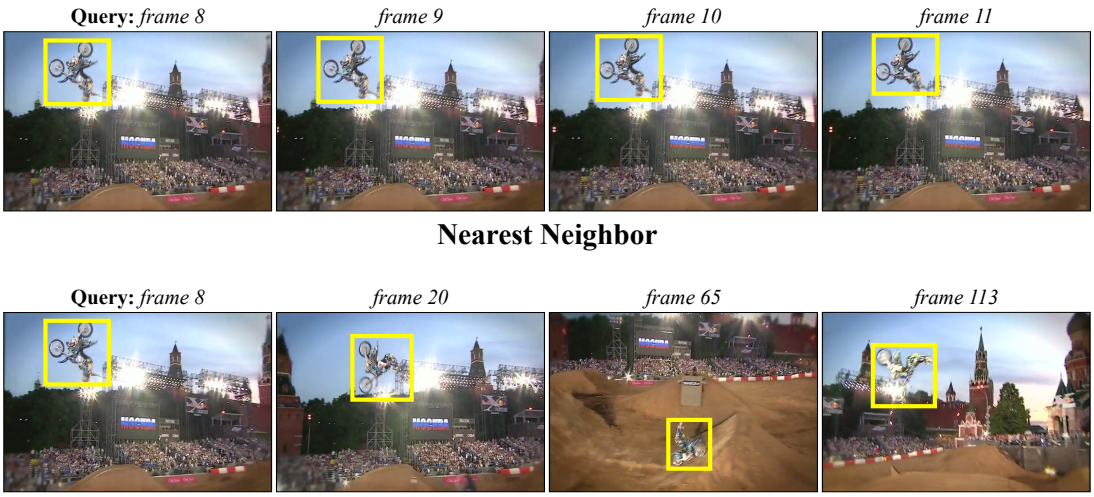

Figure 2.2: Directly searching for nearest neighbors in all frames tends to return patches from consecutive frames, which results in homogeneous and non-informative clusters (first row). Our approach partitions the video into uniform-length temporal segments, and takes one nearest neighbor from each segment. This produces more diverse and informative clusters (second row).

the discovered object bounding boxes, we apply a pixel-wise segmentation algorithm to obtain fine-grained object segmentation masks in each frame.

2.2.1 Initialization

Our initialization step aims to discover a set of clusters, each comprised of the *easy* instances of a candidate object in the video that are spread-out in time. We define as easy instances those that have salient appearance and motion with respect to their surrounding background, since they will be easier for a clustering algorithm to group. By finding instances of an object that are spread-out in time, we can focus on the foreground objects that consistently appear throughout the video and give less emphasis to background objects that appear over only a short period of time.

To identify the easy instances, we begin with a construction similar to that of [147]. To ensure good coverage of the foreground objects, we first generate a large set of *static* object proposals in each frame. Since there can be many frames in the video, we need a fast object proposals method to reduce runtime complexity. To this end, we use Edge Boxes [283], which produces $\sim$1000 box proposals in an image in 0.25 seconds. The algorithm scores each proposal based on the edge contours that are wholly-contained in it, which is indicative of the likelihood

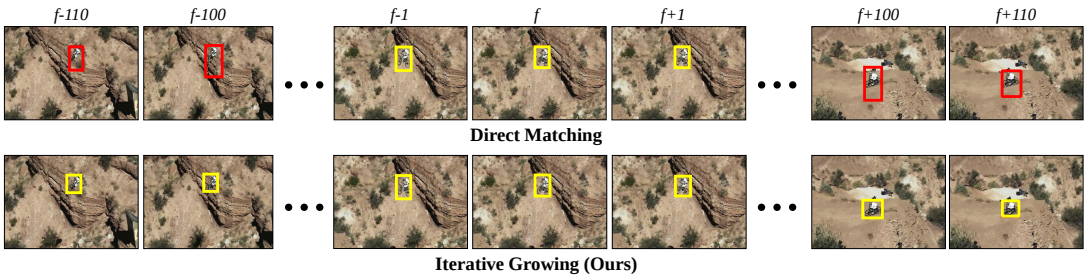

Figure 2.3: Starting from frame f, both direct matching and iterative growing obtain correct detections in adjacent frames $f+1$ and $f-1$. However, for frames that are very far away from f in time, direct matching can drift due to large appearance changes in the object. By iteratively growing and simultaneously updating the detector, our approach can *adapt* to the object's large appearance variation to obtain a correct detection even in far-away frames.

that the proposal contains a whole object [283]. We use this score to measure the object-like appearance s_a and motion distinctiveness s_m of each proposal. Specifically, we extract 1000 proposals each from the RGB frame and the frame's optical flow magnitude map (for a total of 2000 proposals), and then for each proposal, compute its s_a and s_m on the edgemaps [53] computed on the RGB frame and flow map, respectively. Finally, we compute a single combined objectness score for each proposal: $s = s_a * s_m$. Taking the product gives high score to the proposals that have *both* high appearance and high motion scores. In each frame, we retain the top 25 proposals with the highest objectness scores.

Let R be the set of retained high-scoring proposals across the video. We next identify in R those that belong to the same object *and* are spread-out in time, since we would like to focus on the foreground objects that consistently appear throughout the video and ignore background objects that only appear for a short time. Because we do not know how many foreground objects are in the video, we generate a large number of candidate clusters via a soft-clustering approach. The idea is to take each proposal in R as the query patch and retrieve its k-nearest neighbors to form a cluster. However, directly taking nearest neighbors for a query patch tends to produce clusters of patches from consecutive frames since they will be very similar in appearance (see Fig. 2.2). Thus, we instead *force* the nearest neighbors to be spread-out in time.

Specifically, we first split the video into N_s=10 uniform-length contiguous segments in time (i.e., each segment has N/N_s frames), and treat each proposal in R as a seed. We then compute a seed's best matching proposal (nearest neighbor) in each of the N_s segments; we

compute the matching by taking the inner-product between proposals in L2-normalized fc_7 feature space (fully-connected 7th layer activation feature of AlexNet [136], pre-trained on ImageNet classification). Thus, each seed produces a cluster with N_s instances that are spread-out over the video. To account for any foreground objects that may be missing in some of the N_s segments or have widely-varying appearance throughout the video (e.g., a person who is initially facing the camera and later faces away from the camera), we can further create variable-sized clusters by retaining only the k most similar among the N_s nearest neighbors.

Finally, we compute a score for each cluster by summing the objectness score of its instances multiplied with the instances' appearance-similarity to the seed, which rewards large clusters whose instances likely belong to the same object: $s(c) = \sum_j s(p^j) * \left(\phi(p^j)^T \phi(p^{seed})\right)$, where c is a cluster, j indexes c's instances, p^{seed} is the seed of c, and $\phi(\cdot)$ denotes the L2-normalized fc_7 feature. Since we generate clusters using all proposals in R as a seed, there will be many redundant clusters. We therefore perform cluster-level non-maximum suppression: We start with an empty set $\mathbb{S} = \emptyset$, and rank all the clusters in descending order of their cluster score. We then greedily add a cluster c_i to $\mathbb{S}$=$\{\mathbb{S} \cup c_i\}$ if it is significantly different in appearance to any higher-scoring cluster already in $\mathbb{S}$, as measured by the appearance similarity between the clusters' seed proposals.

2.2.2 Iterative growing

The top-K ranked clusters $\{c_1, \ldots, c_K\}$ in $\mathbb{S}$ comprise a diverse set of candidate objects in the video. However, each cluster only covers a small number of easy instances (at most N_s) of an object, and some of them could be noisy. We thus need to detect the harder instances of the object in the remaining frames and correct any existing noisy ones. A natural way to proceed would be to train a detector using the cluster's instances as positives and any non-overlapping proposals in their same frames as negatives, and fire that detector on all frames (as done in [147]). However, the object's appearance can change drastically across the video, so such an approach can be prone to drift.

To tackle this, we instead propose to *iteratively* update the detector by starting with the frames that are temporally close to the initial set of cluster instances, and then gradually grow out to cover all N frames. See Fig. 2.3. When growing out (i.e., detecting new instances), we

initially do not leverage any motion-based tracking cues from existing detections in neighboring frames. This is because we do not want to commit to any detection (especially early on) since there could be noisy detections that lead to drifting. We instead iteratively update the detector until all frames are covered, and then combine the final detector's outputs with motion cues to obtain the final detections.

For each cluster c,[1] we start by training an initial linear SVM detector $\mathbf{w}$ using the cluster's instances $P = \{p^j \mid j \in S\}$ as positives and any proposal with intersection-over-union ratio (IOU) less than 0.4 to any instance in P as negatives. Denote S as the initial set of frames that c already covers (i.e., has an instance in). We next exploit the property that an object's appearance will change slowly over time, in order to make our detection problem easier.

Specifically, we first identify the set of frames S', which are the $n = 3$ temporal neighbors of S that are not yet covered. We then fire $\mathbf{w}$ on all 1000 proposals in each frame in both S' *and* S. By firing the detector even on the frames in S, we can update any mis-detections in those existing frames. We take the proposal with the highest detection score in each frame, and add them to the set of cluster instances P. We then retrain $\mathbf{w}$ with the new and updated instances in P as positives, and any proposal with IOU less than 0.4 to any instance in P as negatives. We update the set of frames that cluster c covers as $S = \{S \cup S'\}$. We repeat this process of detecting/updating instances in neighboring/existing frames and retraining the detector until all frames are covered, i.e., $S = V$.

Finally, we take the final trained detector $\mathbf{w}$, and fire it back on all proposals in all frames in V. We combine its per-frame detections with optical flow based motion cues to encourage smooth detections in time. Formally, we solve the following optimization problem to get a final set of detections $P = \{p^1, \ldots, p^N\}$ using dynamic programming:

$$\max_P J(P) = \sum_{j=1}^{N} \mathbf{w}^T \phi(p^j) + \sum_{j=1}^{N-1} \text{IOU}(p^{j+1}, q^{j+1}), \tag{2.1}$$

where $\phi(p^j)$ is the fc_7 feature of p^j, q^{j+1} is the bounding box location in frame j+1 obtained by shifting p^j in frame j to frame j+1 according to its average optical flow displacement vector, and IOU is the intersection-over-union ratio.

[1] We drop the cluster subscript for simplicity.

The final set of detections P for cluster c forms a spatio-temporal bounding box tube. We repeat the above for each of the top-K clusters.

2.2.3 Pixel-wise segmentation

Thus far, our approach produces a set of spatio-temporal bounding box proposals given an unannotated video. For some applications however (e.g., semantic video segmentation), the ensuing algorithm may require the video object proposals to be pixel segmentations instead of bounding boxes. Hence, we next show how to output pixel-wise segmentation masks, given the bounding boxes we obtained in the previous section as initialization. Similar to GrabCut [198], the main idea is to use the bounding boxes as weak supervision, and iteratively refine a pixel-level appearance model of the object and its corresponding foreground object segmentation in each frame of the video.

Since our bounding box detections have already provided a rough localization of the object-of-interest, we can ignore any regions that are spatially far from each bounding box when computing their segmentations. To this end, we define an *operating region*, with size $[3\times w, 3\times h]$, for each bounding box proposal p (where w and h are the width and height of p, respectively) and is centered on the center pixel of p. We then initialize a pixel-level appearance model by taking all the pixels inside each bounding box as positives, and all pixels outside the *safe region*, which is a 32 pixel-wide boundary that immediately surrounds each bounding box, but within the operating region as negatives. The safe region accounts for any mis-localizations of the input bounding boxes. See Fig. 2.4.

To represent each pixel, we adapt the *hypercolumn* representation [89], which models both low-level appearance and high-level learned semantics. Specifically, we combine the activation features of the *pool2* and *pool5* layers of AlexNet [136]. In order to make sure we have a large enough spatial resolution to model small objects, we resize each operating region to 2400×2400 pixels. This produces a corresponding activation feature map of size 147×147 and 72×72 for *pool2* and *pool5*, respectively. We bi-linearly interpolate the *pool2* feature map to 72×72 to match the size of the *pool5* map. We then train a logistic regression classifier with the positive (inside the bounding box) and negative (outside the safe region) pixels.

To compute the segmentation for a frame f, we define a graph over its operating region

Figure 2.4: The yellow box shows the spatio-temporal bounding box proposal produced from our iterative growing procedure in one frame. To compute its pixel-level segmentation, we first define an *operating region* (orange box) in order to ignore any pixels that are spatially far from the proposal. The blue dotted box is a *safe region*, outside of which we sample negative data to train our pixel-level appearance model, in order to account for any mis-localizations of the initial box proposal.

where a node corresponds to a pixel, and an edge between two nodes corresponds to the cost of a cut. The cost function we minimize is:

$$C(l, f) = \sum_{i \in \mathcal{O}} D_i(l_i) + \sum_{i,j \in \mathcal{N}} V_{i,j}(l_i, l_j), \tag{2.2}$$

where l is a labeling of the pixels, $\mathcal{O} = \{o_1, \ldots, o_m\}$ is the set of m pixels in the operating region, $\mathcal{N}$ consists of the four spatially neighboring pixels, and i and j index the pixels. Each pixel o_i is assigned to $l_i \in \{1, 0\}$, where 1 and 0 correspond to foreground and background, respectively.

We use the pixel-level logistic regression classifier's probability output to compute the data term D_i, which defines the cost of labeling pixel o_i with label l_i. The neighborhood term $V_{i,j}$ encourages label smoothness in space. We compute an edge map using [53] and assign $V_{i,j}$ between o_i and o_j with their edge confidence, which favors assigning the same label to neighboring pixels that do not have a strong edge between them. We then minimize Eqn. 2.2 using graph-cuts [21] to obtain the pixel-wise segmentation for frame f. Finally, we update the pixel appearance model with the newly obtained segmentations from all frames. We take the new model and use it to update the segmentations; we repeat this process until convergence (i.e., the pixel assignments in all frames do not change). See Fig. 2.5.

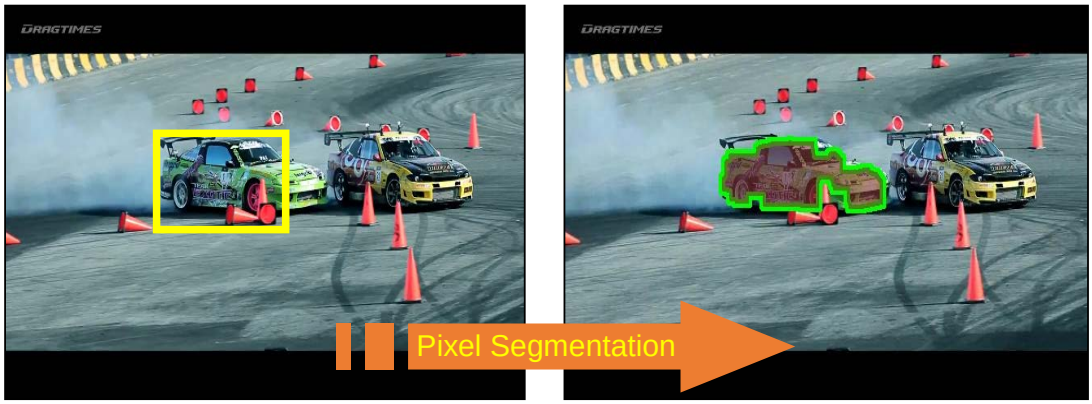

Figure 2.5: Given a bounding box proposal output in a frame (left, yellow box), our algorithm trains a pixel-level appearance model to produce a segmentation mask (right, green boundary). Note that this not only produces a segmentation proposal, but it can also correct mis-localizations from the bounding box proposal.

Due to the large receptive fields of *pool2* and *pool5* features, it is difficult to obtain accurate *pixel-level* predictions. Therefore, we refine the foreground mask with a simple post-processing step: we represent each pixel with an RGB feature vector to train a GMM with 5 components each for the predicted foreground/background pixels. We then apply Eqn. 2.2 again on the pixels to get the final foreground mask.

Bounding box proposal refinement. Finally, our pixel segmentation can in turn be used to improve the bounding box localization that it was initialized on. Among all connected components labeled as foreground that are overlapping with the initial bounding box, we simply keep those whose area is larger than $0.6\times$ the area of the largest component. We then take the bounding box that tightly encloses all the selected components. The refined boxes are taken as our final spatio-temporal box proposals, together with the corresponding segmentation proposals.

2.3 Experiments

We evaluate our bounding box and pixel segmentation proposals against state-of-the-art tracking and video segmentation methods, and conduct ablation studies to analyze the contribution of our iterative growing procedure and the synergy of our bounding box and segmentation tubes.

Implementation details. We set the number of initial proposal clusters to $K = 85$ for the Visual Tracker Benchmark [253] and $K = 150$ for the SegTrack-v2 dataset [148]. For very

	SCM [275]	Struck [88]	TLD [117]	ASLA [110]	CXT [51]	VTD [140]	VTS [141]	TGPR [71]	RPT [149]	[241]	[156]	Ours
Supervised?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
Overall	0.499	0.473	0.437	0.434	0.424	0.416	0.416	0.539	0.576	0.599	0.612	0.437
IV	0.472	0.427	0.399	0.429	0.365	0.42	0.428	N/A	0.555	0.598	0.577	0.453
SV	0.518	0.425	0.421	0.452	0.389	0.405	0.4	N/A	0.535	0.558	0.558	0.451
OCC	0.487	0.412	0.402	0.376	0.369	0.404	0.398	N/A	N/A	0.571	0.615	0.434
DEF	0.448	0.393	0.378	0.372	0.324	0.377	0.368	N/A	N/A	0.644	0.615	0.469
MB	0.298	0.433	0.404	0.258	0.369	0.309	0.304	N/A	0.559	0.580	N/A	0.460
FM	0.296	0.461	0.417	0.248	0.384	0.303	0.299	N/A	0.549	0.565	0.54	0.507
IPR	0.457	0.443	0.416	0.425	0.449	0.43	0.415	N/A	0.569	0.555	N/A	0.423
OPR	0.47	0.431	0.42	0.422	0.416	0.435	0.425	N/A	0.553	0.581	0.605	0.443
OV	0.361	0.459	0.457	0.312	0.427	0.446	0.443	N/A	N/A	0.592	0.596	0.543
BC	0.45	0.458	0.345	0.408	0.338	0.425	0.428	N/A	0.606	0.564	0.58	0.352
LR	0.279	0.372	0.309	0.157	0.312	0.177	0.168	N/A	N/A	0.514	N/A	0.372

Table 2.1: We compare our *unsupervised* video object box proposals to state-of-the-art *supervised* tracking methods on the Visual Tracker Benchmark [253]. The supervised methods require human annotation in the first frame to initialize their tracker. Even without this requirement, our unsupervised approach is able to outperform many of the baselines. We measure accuracy in terms of the Area Under Curve (AUC) of the One Pass Evaluation (OPE) success plot as defined in [253]. Higher is better. *IV-Illumination Variation, SV-Scale Variation, OCC-Occlusion, DEF-Deformation, MB-Motion Blur, FM-Fast Motion, IPR-In-Plane Rotation, OPR-Out-of-Plane Rotation, OV-Out-of-View, BC-Background Clutter, LR-Low Resolution.* (Baseline results are taken from [71, 149, 156, 241, 253].)

difficult videos (e.g. "Penguin" in SegTrack-v2) that do not have well-defined foreground objects, we find that more clusters are needed in order to get an initialization that corresponds to the human-annotated object. For generating clusters, we set the number of neighbors of a seed to $k = \{1, 3, 5, 7, 9\}$ for SegTrack-v2 and fix it to $k = 9$ for Visual Tracker Benchmark. Empirically, we find that a smaller k produces better cluster initializations for videos that have many confusing appearance patterns.

2.3.1 Evaluation of box proposals

We first evaluate the quality of our spatio-temporal box proposals against tracking algorithms on the Visual Tracker Benchmark [253]. This dataset contains 50 test sequences (with frame lengths that range from 71 to 3872) with various challenging properties like illumination/scale variation, occlusion, deformation, etc., which make it an excellent testbed for evaluating the robustness of our box proposals.

We use the standard *success plot* performance metric defined in [253], which measures the ratio of frames that have an intersection-over-union overlap (IOU) score above a threshold, and draw a curve by varying that threshold from 0 to 1 in 0.05 increments. The overall performance of an algorithm is then the area under the success plot curve (AUC). We evaluate our approach

with the box proposal that corresponds to the object with ground-truth annotation, and report the ranking of that proposal based on its cluster score, as described in Sec. 2.2.1.

First, we show the performance of our proposals compared with several supervised tracking methods. We compare against the baselines using the One Pass Evaluation (OPE) [253], which initializes the tracker with a human-annotated box in the first-frame. Note however, that *we do not use this annotation*, as ours is unsupervised. We measure performance both in terms of the average OPE success plot AUC across *all* videos, as well as the AUC for different sub-categories of videos defined by various challenging factors, e.g., illumination variation, scale variation, occlusion, etc. As shown in Table 2.1, our algorithm performs competitively with existing state-of-the-art supervised tracking methods—even outperforming several of them [51, 110, 117, 140, 141]—despite the fact that our method does not require *any human annotation*. In contrast, all of the baselines require human annotation on the first frame to initialize their tracker.

On average, we need 123.7 proposals to achieve the results in Table 2.1, which indicates that our ranking is able to focus on the foreground objects despite the various challenging factors present in these video.

2.3.2 Evaluation of segmentation proposals

We next evaluate our spatio-temporal segmentation proposals. We use the SegTrack-v2 [148] dataset, which contains 14 video sequences with frame lengths varying from 21 to 279 across the sequences. Every frame is annotated with a binary (pixel-level) foreground/background mask. We compare with previous state-of-the-art unsupervised methods [83, 147, 148, 185] and also to a recent supervised method [250] that requires human annotation of the object's boundary in the first frame.

To evaluate segmentation accuracy, we use the intersection-over-union ratio (IOU) measure: $IOU = \frac{|A \cap GT|}{|A \cup GT|}$, where A is the binary segmentation produced by the algorithm and GT is ground-truth binary mask. We evaluate our approach with the segmentation tube that corresponds to the object with ground-truth annotation, and report the ranking of that tube based on its cluster score.

Table 2.2 shows the results. Our method produces state-of-the-art results compared with previous unsupervised methods [83, 147, 148] with a moderate number of proposals (122 on

Sequence/Object	[148]+CSI	[147]	[83]	Ours	[250]
Supervised?	N	N	N	N	Y
Mean per object	65.9	45.3	51.8	**69.1**	71.8
Mean per sequence	71.2	57.3	50.8	**73.9**	72.2
Girl	**89.2**	87.7	31.9	86.4	84.6
Birdfall	62.5	49.0	57.4	**72.5**	78.7
Parachute	93.4	**96.3**	69.1	95.9	94.4
CheetahDeer	37.3	44.5	18.8	**61.2**	66.1
CheetahCheetah	**40.9**	11.7	24.4	39.4	35.3
MonkeydogMonkey	71.3	**74.3**	68.3	74.0	82.2
MonkeydogDog	18.9	4.9	18.8	**39.6**	21.1
Penguin#1	51.5	12.6	**72.0**	53.2	94.2
Penguin#2	76.5	11.3	**80.7**	72.9	91.8
Penguin#3	**75.2**	11.3	75.2	74.4	91.9
Penguin#4	57.8	7.7	**80.6**	57.2	90.3
Penguin#5	**66.7**	4.2	62.7	63.5	76.3
Penguin#6	50.2	8.5	**75.5**	65.7	88.7
Drifting Car#1	**74.8**	63.7	55.2	70.7	67.3
Drifting Car#2	60.6	30.1	27.2	**70.7**	63.7
Hummingbird#1	**54.4**	46.3	13.7	53.0	58.3
Hummingbird#2	72.3	**74.0**	25.2	70.5	50.7
Frog	72.3	0	67.1	**80.2**	56.3
Worm	82.8	**84.4**	34.7	82.4	79.3
Soldier	**83.8**	66.6	66.5	76.3	81.1
Monkey	**84.8**	79.0	61.9	83.1	86.0
Bird of Paradise	**94.0**	92.2	86.8	90.0	93.0
BMXPerson	85.4	87.4	39.2	**86.1**	88.9
BMXBike	24.9	38.6	32.5	**40.3**	5.70
Avg. # of Proposals	60.0	10.6	336.6	121.9	N/A

Table 2.2: Segmentation results on SegTrack-v2 in terms of mean segmentation IOU with ground-truth. Higher is better. For both the "mean per object" and the "mean per sequence" metric, we outperform state-of-the-art unsupervised methods [83, 147, 148] using a moderate number of proposals. We also perform competitively to the supervised method of [250], which requires human annotation on the first frame, whereas we require none. (Baseline results are taken from [148, 250].)

average across the videos). The approach of [83] clusters pixels using bottom-up motion and appearance cues, and thus needs to generate many proposals to obtain good performance. The state-of-the-art approaches of [147, 148] perform better with fewer proposals, by leveraging top-down cues from learned static object proposals [26, 57]. However, [148] tracks multiple static proposals initialized in the first frame, and so inherits challenges associated with tracking (e.g., drifting). This is likely the reason for their low accuracy on videos with fast moving objects like CheetahDeer and BMXBike. In contrast, our approach discovers the easy instances *throughout* the video, and iteratively updates the model to grow-out from those instances. Thus, we find our approach to be more robust to drifting. For this same reason, we outperform the key-segments approach of [147], which like ours, discovers easy instances and builds a model to detect the object in new frames, but unlike ours, does not iteratively refine the model.

Our algorithm even performs competitively to the recent state-of-the-art *supervised* method of [250] that requires manual initialization in the first frame. Although our average under the "mean per object" metric is lower (69.1% vs. 71.8%), we perform better in terms of the "mean per sequence" metric (73.9% vs. 72.2%), *without any supervision*. The main reason our average under the "mean per object" metric is lower is due to the sequences in the Penguin video, which has a group of penguins that are spatially very close to each other, and are similar in appearance and motion. This makes it very difficult for an unsupervised method to keep track of a single penguin. If we remove the penguin sequences from the evaluation, our average improves to 70.7%, while the average of [250] drops to 66.3%, under the "mean per object" metric.

The average number of proposals our approach produces also compares favorably to that of previous methods (see last row of Table 2.2, the lower the better). Unfortunately, the penguin sequences again hurt our average ranking. We need to generate hundreds of proposals to obtain the one corresponding to a specific penguin since many proposals end up with similar rank due to the penguins' similarity in appearance and motion. If we remove the penguin sequences, the average number of proposals needed to localize the ground-truth object drops to 79.6.

Finally, we also compare with other unsupervised methods [23, 180, 246, 269] that report results in terms of average pixel error, which is the average number of incorrectly labeled pixels across frames. Table 2.3 shows the result. We outperform previous methods under this metric by

	[246]	[269]	[180]	[23]	Ours
Average Pixel Error	4766	25289	5859	16074	**2464.5**

Table 2.3: Segmentation results on SegTrack-v2 in terms of average pixel error with ground-truth. Lower is better. We outperform all previous unsupervised methods by a large margin (∼48% error reduction compared with [246]) under this metric. (Baseline results are taken from [246].)

Figure 2.6: (**rows 1-3**) Qualitative results of our spatial-temporal box proposals on the Visual Tracker Benchmark. Our proposals localize the object well, despite challenging factors in the videos, e.g., scale variation (first row), clutter (second row), in-plane rotation (third row), etc. (**rows 4-6**) Qualitative results of our spatial-temporal segmentation proposals on SegTrack-v2. Our approach produces fine details of the object boundary (fourth row), and handles occlusion well (fifth row). Lastly, for the very difficult case in which objects have thin structures, e.g., the bicycle in the last row, our algorithm leaks out qualitatively, even though quantitatively we outperform previous methods.

a large margin. For example, compared with the results of [246], we reduce the error by ∼48%.

2.3.3 Ablation studies

We next study the different components of our algorithm. We use the Visual Tracker Benchmark [253] and measure the performance of our spatio-temporal box proposals using the AUC of the overall success plot.

We first study the effect of our iterative growing procedure by comparing to a baseline that does not iteratively update a model when growing out. Specifically, the baseline takes the detector $\mathbf{w}$ trained on an initially discovered cluster, and fires it on all frames in the video (instead of

firing only on the temporally-adjacent frames and iteratively updating the model). This baseline produces an AUC of 0.319, which is significantly lower than the 0.365 produced with iterative growing. This demonstrates that iterative growing can help avoid drift, as shown in Fig. 2.3.

The next aspect we investigate is the synergy of our box output and segmentation output. Specifically, we measure how much our pixel segmentation helps in producing better localized box proposals (recall from Sec. 2.2.3 that we update the bounding box proposals given the segmentation masks). We compare the bounding boxes produced before and after segmentation refinement. The quality of the bounding boxes further increases from 0.365 to 0.437 after refinement, which shows that our segmentations indeed lead to better object localizations.

2.3.4 Runtime speed analysis

Since object proposals can be building blocks to many vision applications, it is essential that they be fast to compute, so that they are not the computational bottleneck. Thus, we finally perform a detailed run-time speed analysis of our approach. For each of the three steps in our algorithm (initialization, iterative growing, and pixel-level segmentation), we profile the average computation time spent to process each frame. We conduct our analysis on the "Woman" sequence in the Visual Tracker Benchmark, which has 597 frames and is closest to the average frame length (584 frames) of that dataset. We measure the timings on a machine with an Intel i7 3.40GHz CPU and an NVIDIA Tesla K40 GPU.

It takes a total of 3.0 seconds per frame for the initialization stage, which includes computing and scoring the static image proposals (Edge Boxes on RGB image and optical flow magnitude map), optical flow computation (we use the fast GPU-based method of [224]), and obtaining the initial ranked set of clusters $\mathbb{S}$. For iterative growing, extracting fc_7 features for all 2000 proposals for each frame takes 1.25 seconds on a GPU. The remaining spatio-temporal bounding box tube processing—iteratively training an SVM detector and firing it to detect new instances and refine existing ones—takes 0.24 seconds per frame per spatio-temporal tube. For the pixel-wise segmentation stage, the time spent on extracting a hypercolumn feature is 0.85 seconds per frame on a GPU. Generating pixel-segmentations takes 0.46 seconds per frame per spatio-temporal tube.

Note that the feature computation for generating the spatio-temporal bounding box is shared

and thus only needs to be done once (i.e., independent of the number of proposals that we generate). Whereas the hypercolumn feature used to generate the segmentation tubes needs to be computed separately for each proposal. While the computational cost in generating our segmentation proposals is comparable to that of previous approaches [147, 148], generating our box proposals is much faster and thus can be applied in a more practical setting.

2.4 Remarks

We presented an unsupervised approach for spatio-temporal video object proposals. It identifies and groups easy instances of an object in the video to initialize an appearance model, and iteratively updates the model while detecting new instances in temporally-adjacent frames. We demonstrated our method's effectiveness on several datasets, showing state-of-the-art unsupervised video segmentation results, and competitive bounding box tracking results compared to supervised baselines.

Chapter 3

Video Object Detection with an Aligned Spatial-Temporal Memory

As mentioned in Chapter 1, object detection is a fundamental problem in computer vision. While there has been a long history of detecting objects in *static images*, there has been much less research in detecting objects in *videos*. However, cameras on robots, surveillance systems, vehicles, wearable devices, etc., receive videos instead of static images. Thus, for these systems to recognize the key objects and their interactions, it is critical that they be equipped with accurate *video* object detectors.

The simplest way to detect objects in video is to run a static image-based detector independently on each frame. However, due to the different biases and challenges of video (e.g., motion blur, low-resolution, compression artifacts), an image detector usually does not generalize well. More importantly, videos provide *rich temporal and motion information* that should be utilized by the detector during both training and testing. For example, in Fig. 3.1, since the hamster's profile view (frames 1-2) is much easier to detect than the challenging viewpoint/pose in later frames, the image detector only succeeds in detecting the leading frame of the sequence. On the other hand, by learning to aggregate useful information over time, a video object detector can robustly detect the object under extreme viewpoint/pose.

Therefore, in recent years, there has been a growing interest in the community on designing video object detectors [61, 85, 87, 119, 144, 280, 281]. However, many existing methods exploit temporal information in an ad-hoc, post-processing manner – static object detections

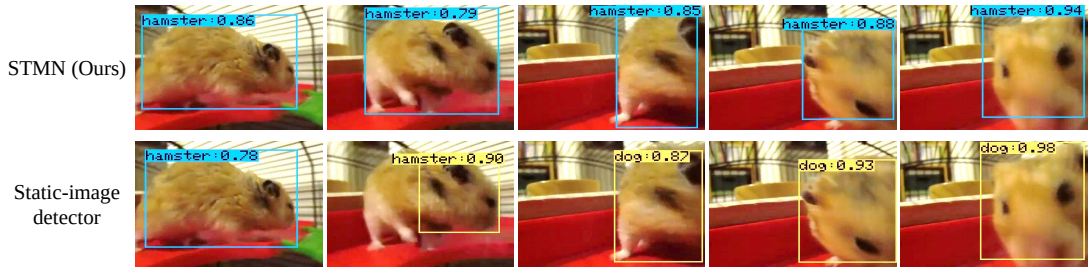

Figure 3.1: Static image detectors (such as Fast-RCNN [77] or R-FCN [42]), tend to fail under occlusion or extreme pose (false detections shown in yellow). By learning to aggregate information across time, our STMN video object detector can produce correct detections in frames with challenging pose/viewpoints. In this example, it aggregates information from the easier profile views of the hamster (first two frames) to aid detection in occluded or extreme views of the hamster (third-fifth frames).

returned by an image detector like R-FCN [42] or Faster R-CNN [196] are linked across frames [87, 119, 144], or video segmentation is performed to refine the detection results [85]. Although these methods show improvement over a static image detector, exploiting temporal information as post-processing is sub-optimal since temporal and motion information are ignored during detector training. As such, they have difficulty overcoming consecutive failures of the static detector e.g., when the object-of-interest has large occlusion or unusual appearance for a long time.

More recent works [61, 280, 281] learn to exploit temporal information *during training* by either learning to combine features across neighboring frames or by predicting the displacement of detection boxes across frames. However, these methods operate on fixed-length temporal windows and thus have difficulty modeling *variable and long-term* temporal information. While the Tubelet Proposal Network [118] does model long-term dependencies, it uses *vectors* to represent the memory of the recurrent unit, and hence loses spatial information. To compensate, it computes the memory vectors at the region-level for each tube (sequence of proposals), but this can be very slow and depends strongly on having accurate initial tubes.

To address these limitations, we introduce the *Spatial-Temporal Memory Network* (STMN), which jointly learns to model and align an object's long-term appearance and motion dynamics in an end-to-end fashion for video object detection. At its core is the Spatial-Temporal Memory Module (STMM), which is a convolutional recurrent computation unit that fully integrates

pre-trained weights learned from static images (e.g., ImageNet [48]). This design choice is critical in addressing the practical challenge of learning from contemporary video datasets, which largely lack intra-category object diversity; i.e., since video frames are highly redundant, a video dataset of e.g., 1 million frames has much lower diversity than an image dataset with 1 million images. By designing our memory unit to be compatible with pre-trained weights from both its preceding and succeeding layers, we show that it outperforms the standard ConvGRU [13] recurrent module for video object detection.

Furthermore, in order to account for the 2D spatial nature of visual data, the STMM preserves the spatial information of each frame in its memory. In particular, to achieve accurate pixel-level spatial alignment over time, the STMM uses a novel MatchTrans module to explicitly model the displacement introduced by motion across frames. Since the convolutional features for each frame are aligned and aggregated in the spatial memory, the feature for any particular object region is well-localized and contains information across multiple frames. Furthermore, each region feature can be extracted trivially via ROI pooling from the memory.

In summary, our main contribution is a novel spatial-temporal memory network for video object detection. Our ablative studies show the benefits provided by the STMM and MatchTrans modules – integrating pre-trained static image weights and providing spatial alignment across time. These design choices lead to state-of-the-art results on the ImageNet video object detection dataset (VID) [1] across different base detectors and backbone networks.

3.1 Related work

Static image object detection. Recent work that adopt deep neural networks have significantly advanced the state-of-the-art in static image object detection [33, 42, 76, 77, 151, 192, 196, 202, 205]. Our work also builds on the success of deep networks to learn the features, classifier, and bounding box localizer in an end-to-end framework. However, in contrast to most existing work that focus on detecting objects in static images, our work aims to detect objects in *videos*.

Video object detection. Compared to static image-based object detection, there has been less research in detecting objects in videos. Early work processed videos captured from a static camera or made strong assumptions about the type of scene (e.g., highway traffic camera for

detecting cars or an indoor room for detecting persons) [40, 251]. Later work used hand-designed features by aggregating simple motion cues (based on optical flow, temporal differences, or tracking), and focused mostly on pedestrian detection [43, 115, 181, 239].

With the introduction of ImageNet VID [1] in 2015, researchers have focused on more generic categories and realistic videos. However, many existing approaches combine per-frame detections from a static image detector via tracking in a two-stage pipeline [87, 119, 235]. Since the motion and temporal cues are used as a post-processing step only during testing, many heuristic choices are required, which can lead to sub-optimal results. In contrast, our approach directly *learns* to integrate the motion and temporal dependencies during training. Our end-to-end architecture also leads to a clean and fast runtime.

Sharing our goal of leveraging temporal information *during training*, the recent works of Zhu et al. [280, 281] learn to combine features of different frames with a feed-forward network for improved detection accuracy. Our method differs in that it produces a *spatial-temporal memory* that can carry on information across long and variable number of frames, whereas the methods in [280, 281] can only aggregate information over a small and fixed number of frames. In Sec. 3.3.3, we demonstrate the benefits gained from this flexibility. Although the approach of Kang et al. [118] uses memory to aggregate temporal information, it uses a vector representation. Since spatial information is lost, it computes a separate memory vector for each region tube (sequence of proposals) which can make the approach very slow. In contrast, our approach only needs to compute a single *frame-level* spatial memory, whose computation is independent of the number of proposals.

Finally, Detect and Track [61] aims to unify detection and tracking, where the correlation between consecutive *two* frames are used to predict the movement of the detection boxes. Unlike [61], our spatial-temporal memory aggregates information across $t > 2$ frames. Furthermore, while our approach also computes the correlation between neighboring frames with the proposed MatchTrans module, we use it to warp the entire feature map for alignment (i.e., at the coarse pixel-level), rather than use it to predict the displacement of the boxes. Overall, these choices lead to state-of-the-art detection accuracy on ImageNet VID.

Learning with videos. Apart from video object detection, other recent work use convolutional and/or recurrent networks for video classification [13, 121, 231]. These methods tend to model entire video frames instead of pixels, which means the fine-grained details required for localizing objects are often lost. Object tracking (e.g., [149, 164]), which requires accurate localization, is also closely-related. The key difference is that in tracking, the bounding box of the first frame is given and the tracker does not necessarily need to know the semantic category of the object being tracked.

Modeling sequence data with RNNs. In computer vision, RNNs have been used for image captioning [54, 130, 238], visual attention [11, 162, 265], action/object recognition [13, 54], human pose estimation [28, 66], and semantic segmentation [274]. Recently, Tripathi et al. [235] adopted RNNs for video object detection. However, in their pipeline, the CNN-based detector is first trained, then an RNN is trained to refine the detection outputs of the CNN.

Despite the wide adoption of RNNs in various vision tasks, most approaches work with vector-form memory units (as in standard LSTM/GRU). To take spatial locality into account, Ballas et al. [13] proposed convolutional gated recurrent units (ConvGRU) and applied it to the task of action recognition. Built upon [13], Tokmakov et al. [230] used ConvGRUs for the task of video object segmentation. Our work differs in three ways: (1) we classify bounding boxes rather than frames or pixels; (2) we propose a new recurrent computation unit called STMM that makes better use of static image detector weights pre-trained on a large-scale image dataset like ImageNet; and (3) our spatial-temporal memory is aligned frame-to-frame through our MatchTrans module. We show that these properties lead to better results than ConvGRU for video object detection.

3.2 Approach

We propose a novel RNN architecture called the Spatial-Temporal Memory Network (STMN) to model an object's changing appearance and motion over time for video object detection.

3.2.1 Overview

The overall architecture is shown in Fig. 3.2. Assuming a video sequence of length T, each frame is first forwarded through a convnet to obtain convolutional feature maps $F_1, F_2, ..., F_T$ as

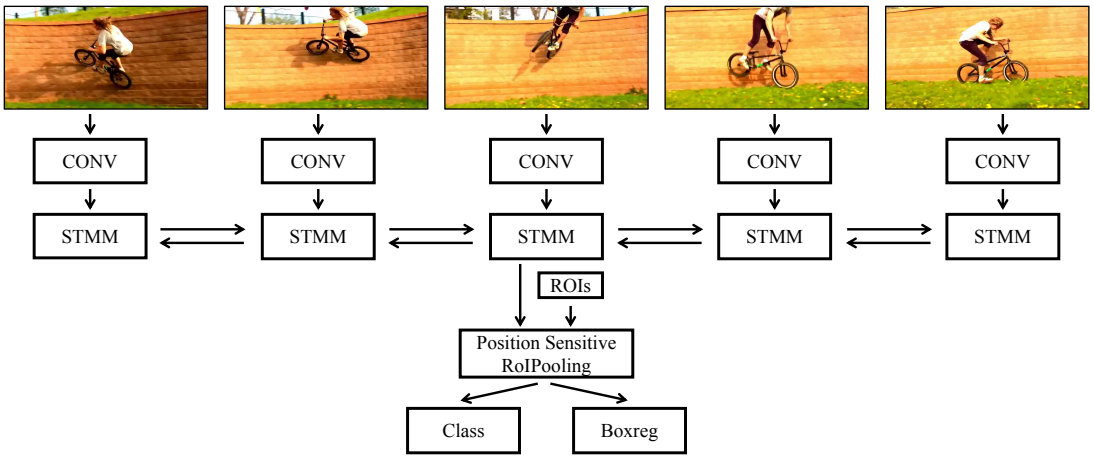

Figure 3.2: Our STMN architecture. Consecutive frames are forwarded through the convolutional stacks to obtain spatial-preserving convolutional feature maps, which are then fed into the spatial-temporal memory module (STMM). In this example, in order to detect an object on the center frame, information flows into the center STMM from all five frames. The STMM output from the center frame is then fed into a classification and box regression sub-network.

appearance features. To aggregate information along the temporal axis, the appearance feature of each frame is fed into the Spatial-Temporal Memory Module (STMM). The STMM at time step t receives the appearance feature for the current frame F_t, as well as a spatial-temporal memory $M^{\rightarrow}_{t-1}$, which carries the information of all previous frames up through timestep $t-1$. The STMM then updates the spatial-temporal memory for the current time step $M^{\rightarrow}_t$ conditioned on both F_t and $M^{\rightarrow}_{t-1}$. In order to capture information from both previous and later frames, we use two STMMs, one for each direction, to obtain both $M^{\rightarrow}$ and $M^{\leftarrow}$. These are then concatenated to produce the temporally modulated memory M for each frame.

The concatenated memory M, which also preserves spatial information, is then fed into subsequent convolution/fully-connected layers for both category classification and bounding box regression. This way, our approach combines information from both the current frame as well as temporally-neighboring frames when making its detections. This helps, for instance, in the case of detecting a frontal-view bicycle in the center frame of Fig. 3.2 (which is hard), if we have seen its side-view (which is easier) from nearby frames. In contrast, a static image detector would only see the frontal-view bicycle when making its detection.

Finally, to train the detector, we use the same loss function used in R-FCN [42]. Specifically, for each frame in a training sequence, we enforce a cross-entropy loss between the predicted

class label and the ground-truth label, and enforce a smooth $L1$ loss on the predicted bounding box regression coefficients. During testing, we slide the testing window and detect on all frames within each sliding window, to be consistent with our training procedure.

3.2.2 Spatial-temporal memory module

We next explain how the STMM models the temporal correlation of an object across frames. At each time step, the STMM takes as input F_t and M_{t-1} and computes the following:

$$z_t = \texttt{BN}^*(\texttt{ReLU}(W_z * F_t + U_z * M_{t-1})), \tag{3.1}$$

$$r_t = \texttt{BN}^*(\texttt{ReLU}(W_r * F_t + U_r * M_{t-1})), \tag{3.2}$$

$$\tilde{M}_t = \texttt{ReLU}(W * F_t + U * (M_{t-1} \odot r_t)), \tag{3.3}$$

$$M_t = (1 - z_t) \odot M_{t-1} + z_t \odot \tilde{M}_t, \tag{3.4}$$

where $\odot$ is element-wise multiplication, $*$ is convolution, and U, W, U_r, W_r, U_z, W_z are the 2D convolutional kernels, whose parameters are optimized end-to-end. Gate r_t masks elements of M_{t-1} (i.e., it allows the previous state to be forgotten) to generate candidate memory $\tilde{M}_t$. And gate z_t determines how to weight and combine the memory from the previous step M_{t-1} with the candidate memory $\tilde{M}_t$, to generate the new memory M_t.

To generate r_t and z_t, the STMM first computes an affine transformation of M_{t-1} and F_t, and then ReLU [136] is applied to the outputs. Since r_t and z_t are gates, their values need to be in the range of $[0, 1]$. Therefore, we make two changes to the standard BatchNorm [108] (and denote it as $\texttt{BN}^*$) such that it normalizes its input to $[0, 1]$, instead of zero mean and unit standard deviation.

First, our variant of BatchNorm computes the mean $\mu(X)$ and standard deviation $\sigma(X)$ for an input batch X, and then normalizes values in X with the linear squashing function $S(X; \mu, \sigma)$ shown in Fig. 3.3. Second, we compute the mean and standard deviation for each batch independently instead of keeping running averages across training batches. In this way, we do not need to store different statistics for different time-steps, which allows us to generate test results for sequence lengths not seen during training (e.g., we can compute detections on longer sequences than those seen during training as demonstrated in Sec. 3.3.3). Note that a key difference between $\texttt{BN}^*$ and instance/layer normalization [12, 105] is that $\texttt{BN}^*$ guarantees

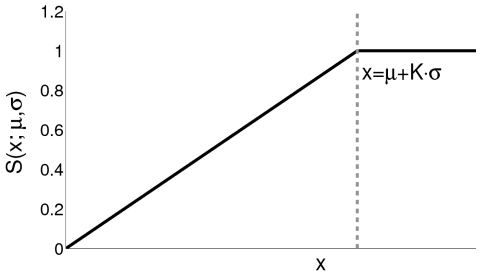

Figure 3.3: $S(x; \mu, \sigma)$ squashes any value in $[0, +\inf)$ into range $[0, 1]$, with a linear scaling function thresholded at $\mu + K \cdot \sigma$. We set $K = 3$.

that *each and every* value in its output is normalized within $[0, 1]$ (which is necessary for gating variables), whereas neither instance nor layer normalization ensures this property. Although simple, we find $\texttt{BN}^*$ works well for our purpose.

Differences with ConvGRU [13] A key practical challenge of learning video object detectors is the lack of intra-category object diversity in contemporary video datasets; i.e., since video frames are highly redundant, a video dataset of e.g., 1 million frames has much lower diversity than an image dataset with 1 million images. The cost of annotation is much higher in video, which makes it difficult to have the same level of diversity as an image dataset. Therefore, transferring useful information from large-scale *image* datasets like ImageNet [48]—into the memory processing unit itself—would benefit our video object detector by providing additional diversity.

Specifically, we would like to initialize our STMN detector with the weights of the state-of-the-art static image-based RFCN detector [42] which has been pretrained on ImageNet DET images, and continue to fine-tune it on ImageNet VID videos. In practice, this would entail converting the last convolution layer before the Position-Sensitive ROI pooling in RFCN into our STMM memory unit (see Fig. 3.2). However, this conversion is non-trivial with standard recurrent units like LSTM/GRU that employ `Sigmoid`/`Tanh` nonlinearities, since they are different from the `ReLU` nonlinearity employed in the R-FCN convolutional layers.

Thus, to transfer the weights of the pre-trained RFCN static image detector into our STMN video object detector, we make two changes to the ConvGRU [13]. First, in order to make full use of the pre-trained weights, we need to make sure the output of the recurrent computation unit is compatible with the pre-trained weights before and after it. As an illustrative example, since

the output of the standard ConvGRU is in $[-1, 1]$ (due to `Tanh` non-linearity), there would be a mismatch with the input range that is expected by the ensuing pre-trained convolutional layer (the expected values should all be positive due to `ReLU`). To solve this incompatibility, we change the non-linearities in standard ConvGRU from `Sigmoid` and `Tanh` to `ReLU`. Second, we initialize W_z, W_r and W in Eqs. 3.1-3.3 with the weights of the convolution layer that is swapped out, rather than initializing them with random weights. Conceptually, this can be thought of as a way to initialize the memory with the pre-trained static convolutional feature maps. In Sec. 3.3.3, we show that these modifications allow us to make better use of pre-trained weights and achieve better detection performance.

3.2.3 Spatial-temporal memory alignment

Next, we explain how to align the memory across frames. Since objects *move* in videos, their spatial features can be mis-aligned across frames. For example, the position of a bicycle in frame $t-1$ might not be aligned to the position of the bicycle in frame t (as in Fig. 3.2). In our case, this means that the spatial-temporal memory M_{t-1} may not be spatially aligned to the feature map for current frame F_t. This can be problematic, for example in the case of Fig. 3.4; without proper alignment, the spatial-temporal memory can have a hard time forgetting an object after it has moved to a different spatial position. This is manifested by a trail of saliency, in the fourth row of Fig. 3.4, due to the effect of overlaying multiple unaligned feature maps. Such hallucinated features can lead to false positive detections and inaccurate localizations, as shown in the third row of Fig. 3.4.

To alleviate this problem, we propose the MatchTrans module to align the spatial-temporal memory across frames. For a feature cell $F_t(x, y) \in 1 \times 1 \times D$ at location (x, y) in F_t, MatchTrans computes the affinity between $F_t(x, y)$ and feature cells in a small vicinity around location (x, y) in F_{t-1}, in order to transform the spatial-temporal memory M_{t-1} to align with frame t. More formally, the transformation coefficients Γ are computed as:

$$\Gamma_{x,y}(i, j) = \frac{F_t(x, y) \cdot F_{t-1}(x + i, y + j)}{\sum_{i,j \in \{-k,\dots,k\}} F_t(x, y) \cdot F_{t-1}(x + i, y + j)},$$

where both i and j are in the range of $[-k, k]$, which controls the size of the matching vicinity.

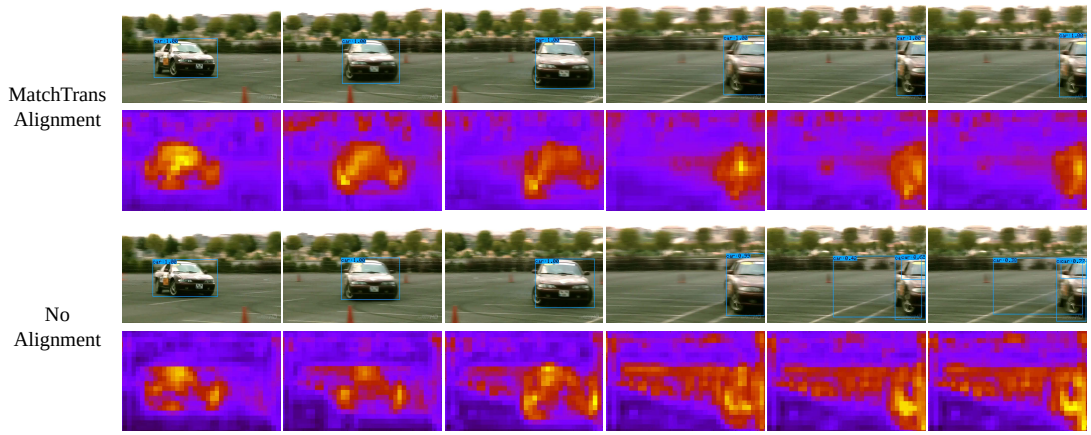

Figure 3.4: Effect of alignment on spatial-temporal memory. In the first and second rows, we show the detection and the visualization of the spatial-temporal memory (by computing the $L2$ norm across feature channels at each spatial location to get a saliency map), respectively, with MatchTrans alignment. The detection and memory without alignment are shown in rows 3 and 4, respectively. Without proper alignment, the memory has a hard time forgetting an object after it has moved to a different spatial position (third row), which is manifested by a trail of saliency on the memory map due to overlaying multiple unaligned maps (fourth row). Alignment with MatchTrans helps generate a much cleaner memory (second row), which also results in better detections (first row). Best viewed in pdf.

With Γ, we transform the unaligned memory M_{t-1} to the aligned M'_{t-1} as follows:

$$M'_{t-1}(x,y) = \sum_{i,j \in \{-k,\dots,k\}} \Gamma_{x,y}(i,j) \cdot M_{t-1}(x+i, y+j).$$

The intuition here is that given transformation Γ, we reconstruct the spatial memory $M'_{t-1}(x,y)$ as a weighted average of the spatial memory cells that are within the $(2k+1) \times (2k+1)$ vicinity around (x,y) on M_{t-1}; see Fig. 3.5. At this point, we can thus simply replace all occurrences of M_{t-1} with the spatially aligned memory M'_{t-1} in Eqs. 3.1-3.4. With proper alignment, our generated memory is much cleaner (second row of Fig. 3.4) and leads to more accurate detections (first row of Fig. 3.4). Since the computational cost is quadratic in k, we set $k = 2$ for all our experiments as this choice provides a good trade-off between performance and computation.

Our MatchTrans is related to the alignment module used in recent video object detection work by [280, 281]. However, [280, 281] use optical flow, which needs to be computed either externally e.g., using [24], or in-network through another large CNN e.g., FlowNet [55]. In contrast, our MatchTrans is much more efficient, saving computation time and/or space for storing optical flow. For example, it is nearly an order of magnitude faster to compute (on

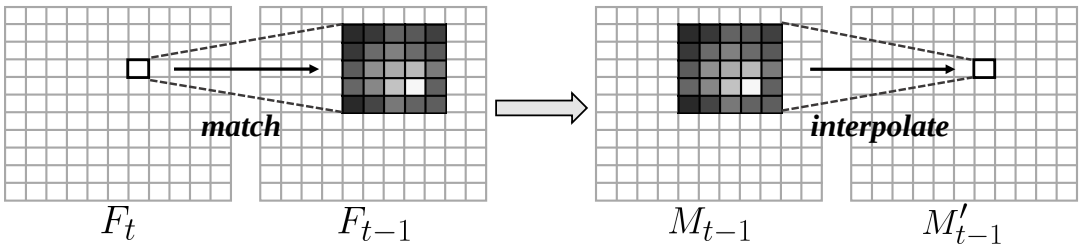

Figure 3.5: The transformation coefficients Γ for position (x, y) are computed by matching $F_t(x, y)$ to $F_{t-1}(i, j)$, where i, j indexes a spatial neighborhood surrounding (x, y). The transformation coefficients are then used to synthesize $M'_{t-1}(x, y)$ by interpolating the corresponding $M_{t-1}(i, j)$ feature vectors.

average, 2.9ms vs. 24.3ms for an 337x600 frame) than FlowNet [55], which is one of the fastest optical flow methods. Also, a similar procedure for computing transformation coefficients was used in [61]. However, in [61], the coefficients are fed as input to another network to regress the displacement of bounding boxes for tracking, whereas we use it to warp the entire feature map for aligning the memory. In other words, rather than use the transformation coefficients to track and connect detections, we instead use them to align the memory over time to produce better features for each candidate object region. We show in Sec. 3.3.1 that this leads to better performance on ImageNet VID.

3.2.4 Temporal linkage during testing

Finally, even though we enforce temporal smoothness in our spatial-temporal memory (i.e., at the feature level), we do not have an explicit smoothness constraint in the output space to ensure that detections in adjacent frames are spatially smooth. We therefore apply standard Seq-NMS [87] over our per-frame detections, following [61, 280].

3.2.5 Approach summary

Through the specially designed Spatial-Temporal Memory and MatchTrans modules, our STMN detector aggregates and aligns useful information from temporally nearby frames for video object detection.

3.3 Results

We show quantitative and qualitative results of our STMN video object detector, and compare to both state-of-the-art static image and video detectors. We also conduct ablation studies to

analyze the different components of our model.

Dataset. We use ImageNet VID [1], which has 3862/555/937 videos for training / validation / testing for 30 categories. Bounding box annotation is provided for all frames. We choose ImageNet VID for its relatively large size as well as for ease of comparison to existing state-of-the-art methods [1, 42, 61, 118, 119, 280, 281].

Implementation details. For object proposals, we use DeepMask [184] for our method and our own baselines. We use the R-FCN detector [42] with ResNet-101 [91] as the backbone network. Following [61], we first train R-FCN on ImageNet DET, and then transfer its weights (using the method described in Sec. 3.2.2) to initialize our STMN detector and continue fine-tuning it on ImageNet VID. We set sequence length $T = 7$ during training. For testing, we observe better performance when using a longer sequence length; specifically, $T = 11$ frames provides a good balance between performance and GPU memory/computation (we later show the relationship between performance and test sequence length). We set the number of channels of the spatial memory to 512. To reduce redundancy within sequences, we form a sequence by sampling 1 in every 10 video frames with uniform stride. For training, we start with a learning rate of 1e-3 with SGD and lower it to 1e-4 when training loss plateaus. During testing we ensemble the detection results of the STMN detector with the initial R-FCN detector from which it started since it comes for free as a byproduct of the training procedure. We employ standard left-right flipping augmentation.

3.3.1 Comparison to state-of-the-art

Table 3.1 shows the comparison to existing state-of-the-art image and video detectors. First, our STMN detector outperforms the static-image based R-FCN detector with a large margin (+7.1%). This demonstrates the effectiveness of our proposed spatial-temporal memory. Our STMN detector also achieves the best performance compared to all existing video object detection methods with ResNet-101 as the base network. Furthermore, in order to enable a fairer comparison to older methods that use Fast/Faster-RCNN + VGG-16 as the base detector and backbone network, we also train an STMN model with the Fast-RCNN as the base detector and VGG-16 as the backbone feature network. Specifically, we first train a static-image Fast-RCNN detector and initialize the weights of STMN using a similar procedure as described in Sec. 3.2.2.[1] With this

	Base network	Base detector	Test	Val
STMN (Ours)	ResNet-101	R-FCN	-	**80.5**
D&T [61]	ResNet-101	R-FCN	-	79.8
Zhu et al. [281]	ResNet-101+DCN	R-FCN	-	78.6
FGFA [280]	ResNet-101	R-FCN	-	78.4
T-CNN [119]	DeepID+Craft [174, 266]	RCNN	67.8	73.8
R-FCN [42]	ResNet-101	R-FCN	-	73.4
TPN [118]	GoogLeNet	TPN	-	68.4
STMN (Ours)	VGG-16	Fast-RCNN	**56.5**	**61.7**
Faster-RCNN [1, 87]	VGG-16	Faster-RCNN	48.2	52.2
ITLab VID - Inha [1]	VGG-16	Fast-RCNN	51.5	-

Table 3.1: mAP comparison to the state-of-the-art on ImageNet VID. For both the "R-FCN+ResNet-101" and the "Fast-RCNN+VGG-16" settings, our STMN detector outperforms all existing methods with the same base detector and backbone network. Furthermore, in both cases, our STMN outperforms the corresponding static-image detector by a large margin.

setting, our STMN achieves 61.7% val mAP, which is much higher than its static-image based counterpart (52.2%). This result shows that our method can be generalized across different base detectors and backbone networks.

When examining per-category results, our method shows the largest improvement on categories like "sheep", "rabbit", and "domestic cat" compared to methods like [61]. In these cases, we see a clear advantage of aggregating information across multiple frames (vs. 2 frames as in [61]), as there can be consecutive "hard" frames spanning multiple (> 2) frames (e.g., a cat turning away from the camera for several frames). On the other hand, we find that the three categories on which we perform the worst are "monkey", "snake", and "squirrel". These are categories with large deformation and strong motion blur. When the per-frame appearance features fail to accurately model these objects due to such challenges, aggregating those features over time with our STMM does not help. Still, overall, we find that our model produces robust detection results across a wide range of challenges as demonstrated next in the qualitative results.

[1]Specifically, we convert the `conv5` layer in VGG-16 to an STMM module by initializing W_z, W_r and W in Eqs. 3.1-3.3 with the weights of `conv5`.

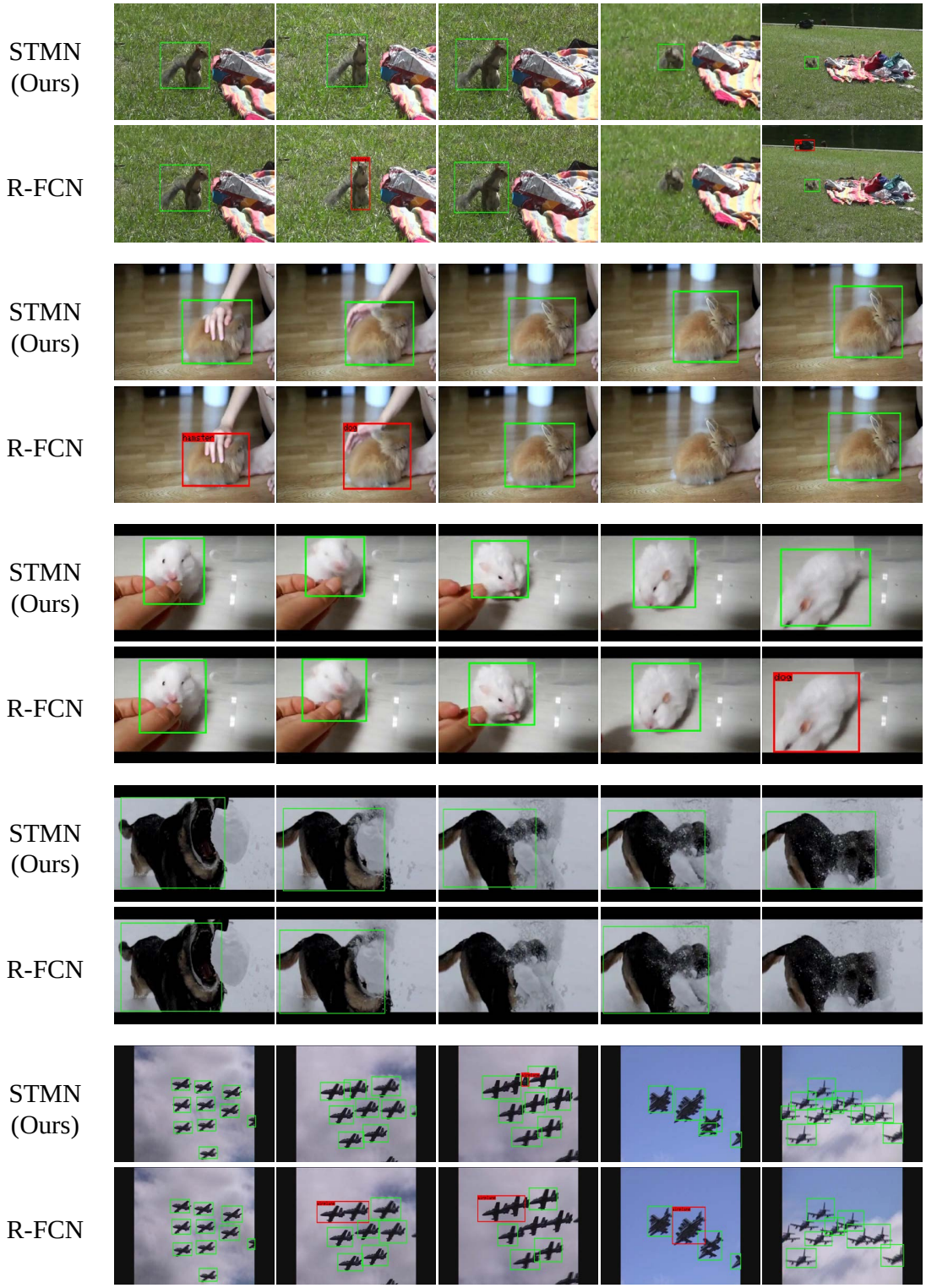

Figure 3.6: Example detections produced by our STMN video object detector vs. R-FCN image detector. Green and red boxes indicate correct and incorrect detections, respectively. For any false positive detection due to misclassification or mislocalization, the predicted category label is shown at the top-left corner of the box. The ground-truth object in each sequence is: "squirrel", "rabbit", "hamster", "dog," and "airplane". Best viewed in pdf, zoomed-in.

3.3.2 Qualitative results

Fig. 3.6 shows qualitative comparisons between our STMN detections and the static image R-FCN detections. Our STMN detections are more robust to motion blur; e.g., in the last frame of the "hamster" sequence, R-FCN gets confused about the class label of the object due to large motion blur, whereas our STMN detector correctly detects the object. In the case of difficult viewpoint and occlusion ("dog" and "rabbit", respectively), our STMN produces robust detections by leveraging the information from neighboring easier frames (i.e., center frame in the "rabbit" sequence and the first frame in the "dog" sequence). Also, our model outputs detections that are more consistent across frames, compared with the static image detector, as can be seen in the case of "squirrel" and "rabbit". Finally, our STMN detector is also more robust in crowded scenes as shown in the "airplane" sequence.

3.3.3 Ablation studies

We next conduct ablation studies to analyze the impact of each component in our model by comparing it to a number of baselines that lack one or more components. For this, we use Fast-RCNN as the base detector and VGG-16 as the backbone network since it is much faster to train compared to RFCN + ResNet-101. To ensure a clean analysis, we purposely do not employ any data augmentation during training for this ablative study.

Contribution of STMN components The first baseline, compared with our model, lacks the MatchTrans module and thus does not align the memory from frame to frame (STMN-No-MatchTrans). The second baseline computes the memory using ConvGRU [13], instead of our proposed STMM. Like ours, this baseline (ConvGRU-Pretrain) also uses pre-trained ImageNet weights for both the feature stack and prediction layers. Our final baseline is ConvGRU without pre-trained weights for the ensuing prediction FCs (ConvGRU-FreshFC).

Table 3.2 shows the results. First, comparing our STMN to the STMN-No-MatchTrans baseline, we observe a 1.7% test mAP improvement brought by the spatial alignment across frames. This result shows the value of our MatchTrans module. To compare our STMM with ConvGRU, we first replace STMM with ConvGRU and as with standard practice, randomly initialize the weights for the FC layers after the ConvGRU. With this setting (ConvGRU-FreshFC), we obtain a relatively low test mAP of 44.8%, due to the lack of data to train the large amount of

	STMN	STMN No-MatchTrans	ConvGRU Pretrain	ConvGRU FreshFC
Test mAP	50.7	49.0	48.0	44.8

Table 3.2: Ablation studies on ImageNet VID. Our improvements over the baselines show the importance of memory alignment across frames with MatchTrans (vs. STMN-No-MatchTrans), and the effectiveness of using pre-trained weights with STMM over standard ConvGRU (vs. ConvGRU-Pretrain and ConvGRU-FreshFC).

weights in the FCs. This result shows that initializing the memory by only partially transferring the pre-trained ImageNet weights is suboptimal. If we instead initialize the weights of the FCs after the ConvGRU with pre-trained weights (ConvGRU-Pretrain), we improve the test mAP from 44.8% to 48.0%. Finally, by replacing `Sigmoid` and `Tanh` with `ReLU`, which is our full model (STMN), we boost the performance even further to 50.7%. This shows the importance of utilizing pre-trained weights in both the feature stacks and prediction head, and the necessity of an appropriate form of recurrent computation that best matches its output to the input expected by the pre-trained weights.

Length of test window size We next analyze the relationship between detection performance and length of test window size. Specifically, we test our model's performance with test window size 3, 7, 11, and 15, on ImageNet VID validation set (the training window size is always 7). The corresponding mAP differences, with respect to that of window size 7, are -1.9%, 0.0%, +0.7%, +1.0%, respectively; as we increase the window size, the performance tends to keep increasing. This suggests the effectiveness of our memory: the longer the sequence, the more longer-range useful information is stored in the memory, which leads to better detection performance. However, increasing the test window size also increases computation cost and GPU memory consumption. Therefore, we find that setting the test window size to 11 provides a good balance.

3.3.4 Computational overhead of STMN

Finally, we sketch the computational overhead of our memory module. To forward a batch of 11 frames of size 337x600, it takes 0.52 and 0.83 seconds for R-FCN and STMN respectively, on a Titan X GPU. The added 0.028 (=0.31/11) secs/frame is spent on STMM computation including

MatchTrans alignment.

3.4 Remarks

We proposed a novel spatial-temporal memory network (STMN) for video object detection. Our main contributions are a carefully-designed recurrent computation unit that integrates pre-trained image classification weights into the memory and an in-network alignment module that spatially-aligns the memory across time. Together, these lead to state-of-the-art results on ImageNet VID. Finally, we believe that our STMN could also be useful for other video understanding tasks that require accurate spatial information like action detection and keypoint detection.

Chapter 4

Integrated Audiovisual Learning for Video Recognition

As discussed in Chapter 1, videos are intrinsically multimodal (i.e., RGB, audio, motion) and therefore it is natural to develop video understanding algorithms that could effectively learn from these modalities in a synergistic manner. To this end, we propose an integrated audiovisual video modeling approach that could be applied in various video understanding tasks including video recognition, action detection and self-supervised video representation learning.

Joint audiovisual learning is core to human perception. However, most contemporary models for video analysis exploit only the visual signal and ignore the audio signal. For many video understanding tasks, audio could be very helpful. Audio has the potential to influence action recognition not only in obvious cases where sound dominates, like "playing saxophone", but also visually subtle cases where the action itself is is difficult to see in the video frames, like "whistling", or closely related actions where sound helps disambiguate, like "closing" vs. "slamming" the door.

This line of thinking is supported by perceptual and neuroscience studies suggesting interesting ways in which visual and audio signals are combined in the brain. A classic example is the McGurk effect [160][1] – when one is listening to an audio clip (e.g., sounding "ba-ba"), alongside watching a video of fabricated lip movements (indicating "va-va"), the sound one perceives *changes* (in this case from "ba-ba" to "va-va").

[1] https://www.youtube.com/watch?v=G-lN8vWm3m0

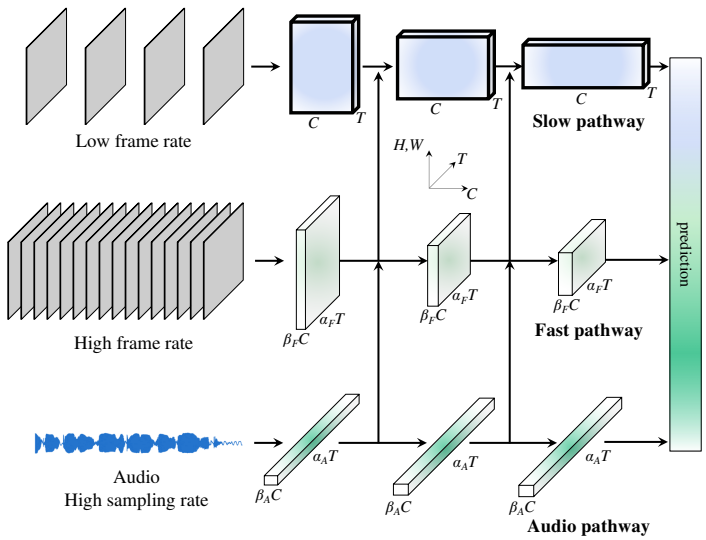

Figure 4.1: **Audiovisual SlowFast Networks** have Slow and Fast visual pathways that are deeply integrated with a Faster Audio pathway to model vision and sound in a unified representation. The network performs hierarchical fusion enabling audio to contribute to the formation of hierarchical audiovisual concepts.

This effect demonstrates that there is tight entanglement between audio and visual signals (known as the *multisensory integration process*) [125, 159, 216, 218]. Importantly, research has suggested this fusion between audio and visual signals happens at a fairly early stage [171, 200].

Given its high potential in facilitating video understanding, researchers have attempted to utilize audio in videos [6, 7, 10, 72, 73, 124, 175, 176, 201]. However, there are a few challenges in making effective use of audio. First, audio does not always correspond to the visual frames (e.g., in a "dunking basketball" video, there can be class-unrelated background music playing). Conversely, audio does not always contain information that can help understand the video (e.g., "shaking hands" does not have a particular sound signature). There are also challenges from a technical perspective. Specifically, we identify the incompatibility of "learning dynamics" between the visual and audio pathways – audio pathways generally train much faster than visual ones, which can lead to generalization issues during joint audiovisual training. Due in part to these various difficulties, a principled approach for audiovisual modeling is currently lacking. Many previous methods adopt an ad-hoc scheme that consists of a separate audio network that is integrated with the visual pathway via "late-fusion" [6, 73, 175].

The objective of this work is to build an architecture for *integrated audiovisual percep-*

tion. We aim to go beyond previous work that performs "late-fusion" of independent audio and visual pathways, to instead learn *hierarchies* of integrated audiovisual features, enabling unified audiovisual perception. We propose a new architecture, *Audiovisual SlowFast* Networks (AVSlowFast), to perform fusion at multiple levels (Fig. 4.1). AVSlowFast Networks build on SlowFast [62], a class of architectures that has two pathways, of which one (Slow) is designed to capture more static but semantic-rich information whereas the other (Fast) is tasked to capture motion. AVSlowFast hierarchically intertwines a Faster Audio pathway with the Slow and Fast pathways, as audio has higher sampling rate, that learns end-to-end from vision and sound. We show that the Audio path can be lightweight ($<$20% of computation), but requires a careful design and training strategies to be useful in practice.

We evaluate our approach on standard datasets in the human action recognition community and find consistent improvement for integrating audio. The improvement in accuracy varies for datasets and classes but comes with a relatively small increase in computational cost. For example, on the leading dataset for egocentric video, EPIC-kitchens [44], audio boosts by +2.9/+4.3/+2.3 the top-1 accuracy for verb/noun/action recognition at 20% of overall compute, on Kinetics [123] action classification by +1.4 top-1 accuracy at 11% of compute, and on AVA [84] action detection by +1.2 mAP at only 2% of the overall compute.

Our key contributions are:

(i) We present AVSlowFast, which fuses audio and visual information *at multiple levels* in the network hierarchy (*i.e.*, hierarchical fusion) so that audio can contribute to the formation of visual concepts at different levels of abstraction. In contrast to late-fusion, this enables the audio signal to participate in the process of forming visual features.

(ii) To overcome the incompatibility of learning dynamics between the visual and audio pathways, we propose *DropPathway*, which randomly drops the Audio pathway during training as a simple and effective regularization technique to tune the pace of the learning process. This enables us to train our joint audiovisual model with hierarchical fusion connections across modalities.

(iii) Inspired by the multisensory integration process mentioned above and prior work in neuroscience [125], which suggests that there exist *audiovisual mirror neurons* in monkey brains

that respond to "any evidence of the action, be it auditory or visual", we propose to perform audio visual synchronization (AVS) [6, 10, 132, 175] at multiple network layers to learn features that generalize across modalities.

(iv) We conduct extensive experiments on six video recognition datasets for human action classification and detection. We report state-of-the-art results and provide ablation studies to understand the trade-offs of various design choices. In addition to evaluating the performance of AVSlowFast for established supervised video classification and detection tasks, we validate the generalization of the audiovisual representation to self-supervised learning, revealing that strong video features can be learned with AVSlowFast using standard pretraining objectives.

4.1 Related Work

Video recognition. Significant progress has been made in video recognition in recent years. Some notable directions are two-stream networks in which one stream processes RGB frames and the other processes optical flow [59, 208, 242], 3D ConvNets as an extension of 2D networks to the spatiotemporal domain [189, 231, 260], and recent SlowFast Networks that have two pathways to process videos at different temporal frequencies [62]. Despite all these efforts on harnessing temporal information in videos, research is relatively lacking when it comes to another important information source – audio in video.

Audiovisual activity recognition. Joint modeling of audio and visual signals has been largely conducted in a "late-fusion" manner in video recognition literature [73, 124, 153]. For example, all the entries that utilize audio in the 2018 ActivityNet challenge report [73] have adopted this paradigm – meaning that there are networks processing visual and audio inputs separately, and then they either concatenate the output features or average the final class scores across modalities. Recently, an interesting audiovisual fusion approach has been proposed [124] using flexible binding windows when fusing audio and visual features. With three similar network streams, this approach fuses audio features with the features from RGB and optical flow at the final stage before classification. In contrast, AVSlowFast is building a hierarchically integrated audiovisual representation.

Multi-modal learning. Researchers have long been interested in developing models that can learn from multiple modalities (e.g., audio, vision, language). Beyond audio and visual modalities, extensive research has been conducted in other instantiations of multi-modal learning, including vision and language [9, 67, 257], vision and locomotion [70, 282], and learning from physiological data [143].

Other audiovisual tasks. Audio has also been extensively utilized outside of video recognition, *e.g.* for learning audiovisual representations in a self-supervised manner [6, 10, 99, 132, 175, 176] by exploiting audio-visual correspondence. Other audiovisual tasks include audio-visual speech recognition [165, 187], lip reading [37], biometric matching [163], sound-source localization [7, 29, 58, 99, 201, 273], audio-visual source separation [72, 175], and audiovisual question answering [2].

4.2 Audiovisual SlowFast Networks

Inspired by research in neuroscience [19], which suggests that audio and visual signals fuse at multiple levels, we propose to fuse audio and visual features at multiple stages, from intermediate-level features to high-level semantic concepts. This way, audio can participate in the formation of visual concepts at different levels. AVSlowFast Networks are conceptually simple: SlowFast has Slow and Fast pathways to process visual input (§4.2.1), and AVSlowFast extends this with an Audio pathway (§4.2.2).

4.2.1 SlowFast pathways

We begin by briefly reviewing the SlowFast architecture [62]. The Slow pathway (Fig. 4.1, top row) is a convolutional network that processes videos with a large temporal stride (*i.e.*, it samples one frame out of τ frames). The primary goal of the Slow pathway is to produce features that capture semantic contents of the video, which has a low refresh rate (semantics do not change all of a sudden). The Fast pathway (Fig. 4.1, middle row) is another convolutional model with three key properties. First, it has an α_F times higher frame rate (*i.e.*, with temporal stride τ/α_F, $\alpha_F > 1$) so that it can capture fast motion information. Second, it preserves fine temporal resolution by avoiding any temporal downsampling. Third, it has a lower channel capacity (β_F times the Slow pathway channels, where $\beta_F < 1$) as it is demonstrated to be a

desired trade-off [62]. We refer readers to [62] for more details.

4.2.2 Audio pathway

A key property of the Audio pathway is that it has an even finer temporal structure than the Slow and Fast pathways (with waveform sampling rate on the order of kHz). As standard processing, we take a log-mel-spectrogram (2-D representation in time and frequency of audio) as input and set the temporal stride to τ/α_A frames, where α_A can be much larger than α_F (*e.g.*, 32 *vs.* 8). In a sense, it serves as a "Faster" pathway with respect to Slow and Fast pathways. Another notable property of the Audio pathway is its low computation cost, as audio signals, due to their lower-dimensional nature, are cheaper to process than visual signals. To control this, we set the channels of the Audio pathway to $\beta_A \times$ Slow pathway channels. By default, we set β_A to $1/2$. Depending on the specific instantiation, the Audio pathway typically only requires 10% to 20% of the overall computation of AVSlowFast.

4.2.3 Lateral connections

In addition to the lateral connections between the Slow and Fast pathways in [62], we add lateral connections between the Audio, Slow, and Fast pathways to fuse audio and visual features. Following [62], lateral connections are added after ResNet "stages" (*e.g.*, pool_1, res_2, res_3, res_4 and pool_5). However, unlike [62], which has lateral connections after each stage, we found that it is most beneficial to have lateral connections between audio and visual features starting from intermediate levels (we ablate this in Sec. 4.3.2). This is conceptually intuitive as very low-level visual features such as edges and corners might not have a particular sound signature. Next, we discuss several concrete AVSlowFast instantiations.

4.2.4 Instantiations

AVSlowFast Networks define a generic class of models that follow the same design principles. In this section, we exemplify a specific instantiation in Table 4.1. We denote spatiotemporal size by $T\times S^2$ for Slow/Fast pathways and $F\times T$ for the Audio pathway, where T is the temporal length, S is the height and width of a square spatial crop, and F is the number of frequency bins for audio.

stage	*Slow* pathway	*Fast* pathway	*Audio* pathway
raw clip	$3\times64\times224^2$	$3\times64\times224^2$	80×128 (*freq.*$\times$*time*)
data layer	stride 16, 1^2	stride 2, 1^2	-
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	[9×1, 1×9], 32 stride 1, 1
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	-
res_2	$\begin{bmatrix} 1\times1^2, 64 \\ 1\times3^2, 64 \\ 1\times1^2, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times1^2, 8 \\ 1\times3^2, 8 \\ 1\times1^2, 32 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 32 \\ [3\times1, 1\times3], 32 \\ 1\times1, 128 \end{bmatrix}\times3$
res_3	$\begin{bmatrix} 1\times1^2, 128 \\ 1\times3^2, 128 \\ 1\times1^2, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 3\times1^2, 16 \\ 1\times3^2, 16 \\ 1\times1^2, 64 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 64 \\ [3\times1, 1\times3], 64 \\ 1\times1, 256 \end{bmatrix}\times4$
res_4	$\begin{bmatrix} 3\times1^2, 256 \\ 1\times3^2, 256 \\ 1\times1^2, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 3\times1^2, 32 \\ 1\times3^2, 32 \\ 1\times1^2, 128 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times6$
res_5	$\begin{bmatrix} 3\times1^2, 512 \\ 1\times3^2, 512 \\ 1\times1^2, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 3\times1^2, 64 \\ 1\times3^2, 64 \\ 1\times1^2, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times3$
global average pool, concat, fc			

Table 4.1: **An instantiation of the AVSlowFast network**. For Slow & Fast pathways, the dimensions of kernels are denoted by $\{T\times S^2, C\}$ for temporal, spatial, and channel sizes. For the Audio pathway, kernels are denoted with $\{F\times T, C\}$, where F and T are frequency and time. Strides are denoted with {temporal stride, spatial stride^2} and {frequency stride, time stride} for SlowFast and Audio pathways, respectively. The speed ratios are $\alpha_F = 8$, $\alpha_A = 32$ and the channel ratios are $\beta_F = 1/8$, $\beta_A = 1/2$ and $\tau = 16$. The backbone is ResNet-50.

SlowFast pathways. For Slow and Fast pathways, we follow the basic instantiation of SlowFast 4×16, R50 model defined in [62]. It has a Slow pathway that samples $T = 4$ frames out from a 64-frame raw clip with a temporal stride $\tau = 16$. There is no temporal downsampling in the Slow pathway, since input stride is large. Also, it only applies non-degenerate temporal convolutions (temporal stride > 1) in res_4 and res_5 (see Table 4.1), as this is more effective.

For the Fast pathway, it has a higher frame rate ($\alpha_F = 8$) and a lower channel capacity ($\beta_F = 1/8$), such that it can better capture motion while trading off model capacity. To preserve fine temporal resolution, the Fast pathway has non-degenerate temporal convolutions in every residual block. Spatial downsampling is performed with stride 2^2 convolution in the center ("bottleneck") filter of the first residual block in each stage of both the Slow and Fast pathways.

Audio pathway. The Audio pathway takes as input the log-mel-spectrogram representation, which is a 2-D representation with one axis being time and the other one denoting frequency bins. In the instantiation in Table 4.1, we use 128 spectrogram frames (corresponding to 2 seconds of audio) with 80 frequency bins.

Similar to the Slow and Fast pathways, the Audio pathway is also based on a ResNet, but with specific design to better fit the audio inputs. First, we do not pool after the initial convolutional filter (*i.e.* there is no downsampling layer at stage pool_1) to preserve information along both temporal and frequency axis. Downsampling in time-frequency space is performed by stride 2^2 convolution in the center ("bottleneck") filter of the first residual block in each stage from res_2 to res_5. Second, we decompose the 3×3 convolution filters in res_2 and res_3 into 1×3 filters for frequency and 3×1 filters for time. This increases accuracy slightly (by 0.2% on Kinetics) but also reduces computation. Conceptually, it allows the network to treat time and frequency separately (as opposed to 3×3 filters which imply both axes are uniform) in early stages. While for spatial filters it is reasonable to perform filtering in x and y dimensions symmetrically, this might not be optimal for early filtering in time and frequency dimensions, as the statistics of spectrograms are different from natural images, which instead are approximately isotropic and shift-invariant [104, 199].

Lateral connections. There are many options for how to fuse audio features into the visual pathways. Here, we describe several instantiations and the motivation behind them. Note that this section discusses the lateral connections between the Audio and SlowFast pathways. For the fusion connection between the two visual pathways (Slow and Fast), we adopt the temporal strided convolution as it is demonstrated to be most effective in [62].

(i) *A*→*F*→*S*: In this approach (Fig. 4.2 left), the Audio pathway (A) is first fused to the Fast pathway (F), and then fused to the Slow pathway (S). Specifically, audio features are subsampled to the temporal length of the Fast pathway and then fused into the Fast pathway with a *sum* operation. After that, the resulting features are further subsampled by α_F (e.g., $4\times$ subsample) and fused with the Slow pathway (as is done in SlowFast). The key property of this approach is that it enforces *strong temporal alignment* between audio and visual features, as audio features are fused into the Fast pathway which preserves fine temporal resolution.

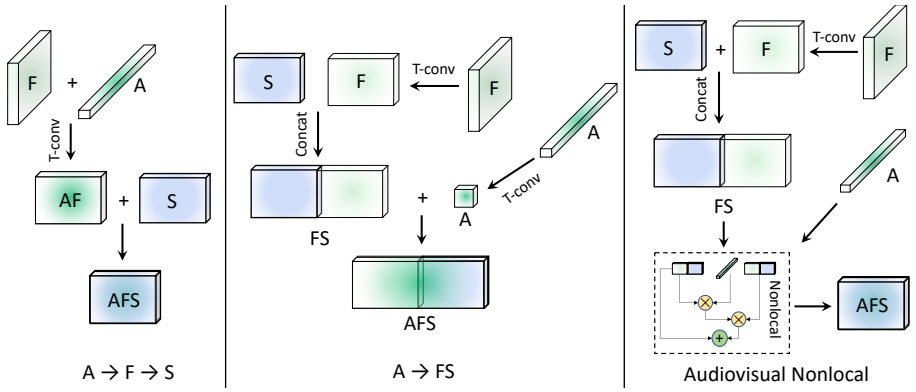

Figure 4.2: **Fusion connections for AVSlowFast**. *Left*: A→F→S enforces strong temporal alignment between audio and RGB frames, as audio is fused into the Fast pathway with fine temporal resolution. *Center*: A→FS has higher tolerance on temporal misalignment as audio is fused into the temporally downsampled output of SlowFast fusion. *Right*: Audiovisual Nonlocal fuses through a Nonlocal block [248], such that audio features are used to select visual features that are deemed important by audio.

(ii) *A→FS*: An alternative way is to fuse the Audio pathway into the output of the SlowFast fusion (Fig. 4.2 center), which is coarser in temporal resolution. We adopt this design as our default choice as it imposes a less stringent requirement on temporal alignment between audio and visual features, which we found to be important in our experiments. Similar ideas of relaxing the alignment requirement are also explored in [124], in the context of combining RGB, flow, and audio streams.

(iii) *Audiovisual Nonlocal*: One might also be interested in using audio as a *modulating signal* to visual features. Specifically, instead of directly summing or concatenating audio features into the visual stream, one might expect audio to play a more subtle role of modulating the visual concepts, through attention mechanisms such as Non-Local (NL) blocks [248]. One example would be audio serving as a probing signal indicating where the interesting event is happening in the video, both spatially and temporally, and then focusing the attention of visual pathways on those locations. To materialize this, we adapt NL blocks to take both audio and visual features as inputs (Fig. 4.2 right). Audio features are then matched to different locations within visual features (along H, W and T axis), and the affinity is used to generate a new visual feature that combines information from locations deemed important by audio features.

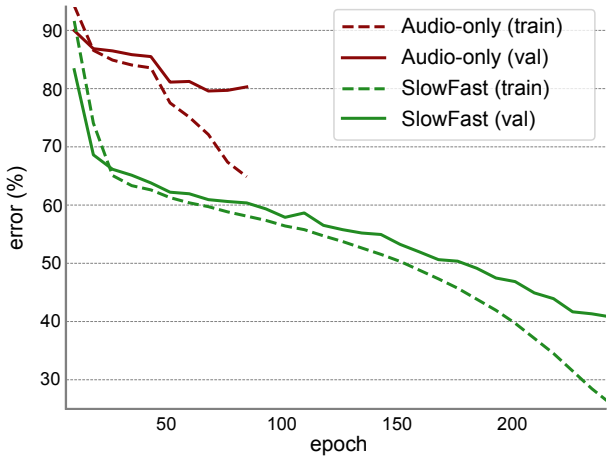

Figure 4.3: Training procedure on Kinetics for Audio-only (red) *vs.* SlowFast (green) networks. We show the top-1 training error (dash) and validation error (solid). The curves show single-crop *errors*; the video *accuracy* is 24.8% *vs.* 75.6%. The audio network converges after around $3\times$ fewer iterations compared to the visual.

4.2.5 Joint audiovisual training

Unlike SlowFast, AVSlowFast trains with multiple modalities. As discussed previously, this leads to challenging training dynamics (*i.e.*, different training speed of audio and visual pathways). To tackle this, we propose two training strategies that enable joint training.

DropPathway. We discuss a possible reason for why previous methods employ audio in a late fusion approach. By analyzing the model training dynamics we observe the following. Audio and visual pathways are very different in terms of their "learning speed".

Taking the curves in Fig. 4.3 as an example, the green curve is for training a visual-only SlowFast model, whereas the red curve is for training an Audio-only model. It shows that the Audio-only model requires fewer training iterations before it starts to overfit (at ~70 epochs, which is $\sim 1/3$ of the visual model's training epochs). One modality dominating multi-modal training has also been observed for lip-reading applications [37] and optical flow streams in action recognition [60] and video object segmentation [261].

The discrepancy on learning pace leads to overfitting if we naively train both modalities jointly. To unlock the potential of joint training, we propose a simple strategy of randomly dropping the Audio pathway during training (*DropPathway*). Specifically, at each training iteration, we drop the Audio pathway altogether with probability P_d. This way, we *slow down*

the learning of the Audio pathway and make its learning dynamics more compatible with its visual counterpart. When dropping the audio pathway, we sum zero tensors with the visual pathways (we also explored feeding the running average of audio features, and found similar results, possibly due to BN).

Our ablation studies in the next section will show the effect of DropPathway, showing that this simple strategy provides good generalization and is essential for jointly training AVSlowFast. Note that DropPathway is different from simply setting different learning rates for the audio/visual pathways in that it 1) ensures the Audio pathway has fewer parameter updates, 2) hinders the visual pathway to 'shortcut' training by memorizing audio information, and 3) provides extra regularization as different audio clips are dropped in each epoch.

Hierarchical audiovisual synchronization. As noted in Sec. 4.1, temporal synchronization (that comes for free) between audio and visual sources has been explored as a self-supervisory signal to learn feature representations [6, 10, 36, 86, 132, 175, 176]. In this work, we use audiovisual synchronization to encourage the network to produce feature representations that are generalizable across modalities (inspired by the *audiovisual mirror neurons* in primate vision [125]). Specifically, we add an auxiliary task to classify whether a pair of audio and visual frames are *in-sync* or not [132, 175] and adopt a curriculum schedule used in [132] that starts with easy negatives (audio and visual frames come from different clips), and transition into a mix of easy and hard (audio and visual frames are from the same clip, but with a temporal shift) after 50% of training epochs. In our experiments, we study the effect of audiovisual synchronization for both supervised and self-supervised audiovisual feature learning.

4.3 Experiments: Action Classification

We evaluate our approach on six video recognition datasets using standard evaluation protocols. For the action classification experiments in this section we use EPIC-Kitchens [44], Kinetics-400 [123], and Charades [207]. For action detection, we use the AVA dataset [84] covered in §4.4, and the AVSlowFast self-supervised representation is evaluated on UCF101 [215] & HMDB51 [137] in §4.5.

Datasets. The EPIC-Kitchens dataset [44] consists of daily activities captured in various kitchen

environments with egocentric video and sound recordings. It has 39k segments in 432 videos. For each segment, the task is to predict a verb (*e.g.*, "turn-on"), a noun (*e.g.*, "switch"), and an action by combining the two ("turn on switch"). Performance is measured as top-1 and top-5 accuracy. We use the train/val split following [16]. Test set results are obtained by submitting to the evaluation server.

Kinetics-400 [123] (abbreviated as K400) is a large-scale video dataset of ~240k training videos and 20k validation videos in 400 action categories. Results on Kinetics are reported as top-1 and top-5 classification accuracy (%).

Charades [207] is a dataset of ~9.8k training videos and 1.8k validation videos in 157 classes. Each video has multiple labels of activities spanning ~30 seconds. Performance is measured in mean Average Precision (mAP).

Audio pathway. Following previous work [6, 7, 132, 133], we extract log-mel-spectrograms from the raw audio waveform to serve as the input to Audio pathway. Specifically, we sample audio data with 16 kHz sampling rate, then compute a spectrogram with window size of 32ms and step size of 16ms. The length of the audio input is exactly matched to the duration spanned by the RGB frames. For example, under 30 FPS, for AVSlowFast with $T\times\tau = 8\times8$ frames (2 secs) input, we sample 128 frames (2 secs) in log-mel space.

Training. We train our AVSlowFast models on Kinetics from scratch without any pre-training. We use synchronous SGD and follow the training recipe (learning rate, weight decay, warm-up, etc) used in [62]. Given a training video, we randomly sample T frames with stride τ and extract the corresponding log-mel-spectrogram. For video frames, we randomly crop 224×224 pixels from a video, randomly flip horizontally, and resize it to a shorter side sampled in [256, 320] pixels [209, 248].

Inference. Following previous work [62, 248], we uniformly sample 10 clips from a video along its temporal axis. For each clip, we resize the shorter spatial side to 256 pixels and take 3 crops of 256×256 along the longer side to cover the spatial dimensions. Video-level predictions are computed by averaging softmax scores. We report the actual *inference-time* computation as in [62], by listing the FLOPs per spacetime "view" of spatial size 256^2 (temporal clip with spatial crop) at inference *and* the number of views (*i.e.* 30 for 10 temporal clips each with 3 spatial

	verbs		nouns		actions	
model	top-1	top-5	top-1	top-5	top-1	top-5
validation						
3D CNN [252]	49.8	80.6	26.1	51.3	19.0	37.8
LFB [252]	52.6	81.2	**31.8**	56.8	22.8	41.1
SlowFast [62]	55.8	83.1	27.4	52.1	21.9	39.7
AVSlowFast	**58.7**	**83.6**	31.7	**58.4**	**24.2**	**43.6**
Δ	*+2.9*	*+0.5*	*+4.3*	*+6.3*	*+2.3*	*+3.9*
test s1 (seen)						
HF-TSN [220]	57.6	87.8	39.9	65.4	28.1	48.6
RU-LSTM [68]	56.9	85.7	43.1	67.1	33.1	55.3
FBK-HUPBA [221]	63.3	89.0	44.8	69.9	35.5	57.2
LFB [252]	60.0	88.4	45.0	**71.8**	32.7	55.3
EPIC-Fusion [124]	64.8	**90.7**	46.0	71.3	34.8	56.7
AVSlowFast	**65.7**	89.5	**46.4**	71.7	**35.9**	**57.8**
test s2 (unseen)						
HF-TSN [220]	42.4	75.8	25.2	49.0	16.9	33.3
RU-LSTM [68]	43.7	73.3	26.8	48.3	19.5	37.2
FBK-HUPBA [221]	49.4	77.5	27.1	52.0	20.3	37.6
LFB [252]	50.9	77.6	31.5	57.8	21.2	39.4
EPIC-Fusion [124]	52.7	79.9	27.9	53.8	19.1	36.5
AVSlowFast	**55.8**	**81.7**	**32.7**	**58.9**	**24.0**	**43.2**
Δ	*+3.1*	*+1.8*	*+4.8*	*+4.9*	*+2.3*	*+6.7*

Table 4.2: **EPIC-Kitchens validation and test results.** Models pretrain on Kinetics [62, 124, 252] or ImageNet [68, 220, 221]. SlowFast backbones: $T\times\tau = 8\times8$, R101. AVSlowFast shows strong margins (Δ) over SlowFast and previous state-of-the-art [124].

crops). Details on training & inference are provided in §4.7.

4.3.1 Main Results

EPIC-Kitchens. We compare to state-of-the-art methods on EPIC-Kitchens in Table 4.2. First, AVSlowFast improves SlowFast with gains of **+2.9** / **+4.3** / **+2.3** top-1 accuracy for verb / noun / action, which highlights the benefits of audio in egocentric video recognition.

Second, as a system-level comparison, AVSlowFast exhibits higher performance in all three categories (verb/noun/action) and two test sets (seen/unseen) vs. state-of-the-art [124] under

Kinetics-400 pretraining.

Comparing to LFB [252], which uses an object detector to localize objects, AVSlowFast achieves similar performance for nouns (objects) on both the seen and unseen test sets, whereas SlowFast *without audio* is largely lagging behind (-4.4% *vs.* LFB on val noun), which is intuitive as sound can be beneficial for recognizing objects.

We observe large performance gains over previous best [124] (which utilizes rgb, audio and flow) on the unseen split (*i.e.*, novel scenes) of the test set (**+3.1 / +4.8/ +2.3** for verb / noun / action), which shows AVSlowFasts' strength on test data.

Kinetics. Table 4.3 shows a comparison on the well-established Kinetics dataset. Comparing AVSlowFast with SlowFast shows a margin of 1.4% top-1 for R50 and 0.9% top-1 accuracy for R101, given the same network backbone and input size. This demonstrates the effectiveness of the audio stream despite its modest cost of only $\approx$10%$-$20% of the overall computation. Comparatively, going deeper from R50 to R101 increases computation by 194%.

On a system-level, AVSlowFast compares favorably to existing methods that utilize various modalities, *i.e.*, audio (A), visual frames (V) and optical flow (F). Adding optical flow streams brings roughly similar gains as audio but *doubles* computation (TS in Table 4.3), not counting optical flow computation; by contrast, audio processing is lightweight (*e.g.* 11% computation overhead for AVSlowFast, R50). Further, AVSlowFast does not rely on pretraining and is competitive with multi-modal approaches that pretrain individual modality streams (✓).

Furthermore, as Kinetics is a visual-heavy dataset (for many classes *e.g.* "*writing*" audio is not useful), to better study audiovisual learning, [6] proposes "Kinetics-Sounds" as a subset of 34 Kinetics classes that are potentially manifested both visually and aurally (example classes include "*blowing nose*" and "*playing drums*"). We test both SlowFast and AVSlowFast on Kinetics-Sounds in the "KS" column of Table 4.3. As expected, the gain from SlowFast to AVSlowFast is stronger on Kinetics-Sounds – for R50/R101, gains doubled to +3.2%/+2.3%, showing the potential of audio on relevant data.

Charades. We test the effectiveness of AVSlowFast on videos of longer range activities on Charades in Table 4.4. We observe that audio can facilitate recognition (+1.2% over a strong SlowFast baseline) and we achieve state-of-the-art performance under Kinetics-400 pre-training.

model	inputs	pretrain	top-1	top-5	KS	GFLOPs×views
R(2+1)D [232]	V	-	72.0	90.0	N/A	152 × 115
TS R(2+1)D [232]	V+F	-	73.9	90.9		304 × 115
ECO [284]	V	-	70.0	89.4		N/A × N/A
ip-CSN-152 [233]	V	-	77.8	92.8		109 × 30
S3D [260]	V	-	69.4	89.1		66.4 × N/A
I3D [27]	V	✓	72.1	90.3		108 × N/A
TS I3D [27]	V+F	✓	75.7	92.0		216 × N/A
TS I3D [27]	V+F	-	71.6	90.0		216 × N/A
Nonlocal [248], R101	V	✓	77.7	93.3		359 × 30
S3D-G [260]	V	✓	74.9	92.0		71.4 × N/A
TS S3D-G [260]	V+F	✓	77.2	93.0		143 × N/A
3-stream SATT [20]	A+V+F	✓	77.7	93.2		N/A × N/A
SlowFast, R50 [62]	V	-	75.6	92.0	80.5	36 × 30
AVSlowFast, R50	A+V	-	77.0	92.7	83.7	40 × 30
SlowFast, R101 [62]	V	-	77.9	93.2	82.7	106 × 30
AVSlowFast, R101	A+V	-	**78.8**	**93.6**	**85.0**	129 × 30

Table 4.3: **AVSlowFast results on Kinetics.** AVSlowFast and SlowFast instantiations are with $T\times\tau = 4\times16$ and $T\times\tau = 8\times8$ inputs for R50/R101 backbones, without NL blocks. "TS" indicates Two-Stream. "KS" refers to top-1 accuracy on Kinetics-Sounds dataset [6]. "pretrain" refers to ImageNet pretraining.

model	pretrain	mAP	GFLOPs×views
Nonlocal, R101 [248]	ImageNet+Kinetics	37.5	544 × 30
STRG, R101+NL [247]	ImageNet+Kinetics	39.7	630 × 30
Timeception [107]	Kinetics-400	41.1	N/A×N/A
LFB, +NL [252]	Kinetics-400	42.5	529 × 30
SlowFast	Kinetics-400	42.5	234 × 30
AVSlowFast	Kinetics-400	**43.7**	278 × 30

Table 4.4: **Comparison with the state-of-the-art on Charades**. SlowFast and AVSlowFast are with R101+NL backbone and 16×8 sampling.

Discussion. Overall, our experiments on action classification indicate that, on standard, visually created datasets for classification, a consistent improvement over very strong visual baselines can be achieved by modeling audio with AVSlowFast. For some cases improvements are

fusion stage	top-1	top-5	GFLOPs
SlowFast	75.6	92.0	36.1
SlowFast+Audio	76.1	92.0	-
$pool_5$	75.4	92.0	38.4
res_4 + $pool_5$	76.5	92.6	39.1
$res_{3,4}$ + $pool_5$	**77.0**	**92.7**	39.8
$res_{2,3,4}$ + $pool_5$	75.8	92.4	40.2

(a) **Fusion stages**.

fusion design	top-1	top-5	GFLOPs
A→F→S	75.3	91.8	51.4
A→FS	77.0	92.7	39.8
AV Nonlocal	**77.2**	92.9	39.9

(b) **Fusion designs**.

Table 4.5: **Effects of hierarchical fusion.** All models are based on R50 and input size 4×16.

exceptionally high (*e.g.* EPIC) and in some lower (*e.g.* Charades), and all results suggest that with AVSlowFast, audio can serve as an *economical* modality that supplements visual input.

4.3.2 Ablation Studies

We ablate of our approach on Kinetics as it represents the largest unconstrained dataset for human action recognition.

Hierarchical fusion. We first study the effectiveness of fusion in Table 4.5a. The first interesting phenomenon is that direct ensembling (late-fusion) of audio/visual models produces only modest gains (76.1% vs 75.6%), whereas joint training with late-fusion ("$pool_5$") does not help (75.6% → 75.4%).

For hierarchical, multi-level fusion, Table 4.5a shows it is beneficial to fuse audio and visual features at multiple levels. Specifically, we found that recognition accuracy steadily increases from 75.4% to 77.0% when we increase the number of fusion connections from one (*i.e.*, only concatenating pool5 outputs) to three ($res_{3,4}$ + $pool_5$) where it peaks. Adding another lateral connection at res_2 decreases accuracy. This suggests that it is beneficial to start fusing audio and visual features from intermediate levels (res_3) all the way to the top of the network. We hypothesize that this is because audio facilitates the formation of visual concepts, but only when features mature to intermediate concepts that are generalizable across modalities (*e.g.* local edges do not have a general sound pattern).

Lateral connections. We ablate the the effect of different types of lateral connections between audio and visual pathways in Table 4.5b. First, A→F→S, which enforces strong temporal

β_A	top-1	top-5	GFLOPs
1/8	76.0	92.5	36.0
1/4	76.6	92.7	36.8
1/2	**77.0**	92.7	39.8
1	75.9	92.4	51.9

(a) **Audio channels** β_A.

P_d	top-1	top-5
-	75.2	91.8
0.2	76.0	92.5
0.5	76.7	92.7
0.8	**77.0**	92.7

(b) **DropPathway rate** P_d.

AVS	top-1	top-5
-	76.4	92.5
res_5	76.7	92.8
$\text{res}_{4,5}$	76.9	92.9
$\text{res}_{3,4,5}$	**77.0**	92.7

(c) **AV synchronization**.

Table 4.6: **Ablations on AVSlowFast design** on Kinetics-400. We show top-1/5 classification accuracy (%), and computational complexity measured in GFLOPs for a single clip input of spatial size 256^2. Backbone: 4×16, R-50.

alignment between audio and visual streams, produces lower classification accuracy compared to A→FS, which relaxes the requirement on alignment. This is consistent with findings in [124] that it is beneficial to have tolerance on alignment between the modalities, since class-level audio signals might happen out-of-sync to visual frames (*e.g.*, when shooting 3 pointers in basketball, the net-touching sound only comes after the action finishes). Finally, the straightforward A→FS connection performs similarly to the more complex Audiovisual Nonlocal [248] fusion (77.0% vs 77.2%). We use A→FS as our default lateral connection for its good performance and simplicity.

Audio pathway capacity. We study the impact of the number of channels of the Audio pathway (β_A) in Table 4.6a. As expected, when we increase the number of channels (*e.g.*, increasing β_A from 1/8 to 1/2, which is the ratio between Audio and Slow pathway's channels), accuracy improves at the cost of increased computation. However, performance starts to degrade when we further increase it to 1, likely due to overfitting. We use $\beta_A = 1/2$ across all our experiments.

DropPathway. We apply Audio pathway dropping to adjust the incompatibility of learning speed across modalities. Here we conduct ablative experiments to study the effects of different drop rates P_d. The results are shown in Table 4.6b. As shown in the table, a high value of P_d (0.5 or 0.8) is required to slow down the Audio pathway when training audio and visual pathways jointly. In contrast, when we train AVSlowFast without DropPathway ("-"), the accuracy degrades to be even worse than visual-only models (75.2% vs 75.6%). This is because the Audio pathway learns too fast and starts to overfit and dominate the visual feature learning. The gain from 75.2% → 77.0% reflects the full impact of DropPathway.

model	dataset	pretrain	top-1	top-5	GFLOPs
VGG* [94]	Kinetics-600	Kinetics-400	23.0	N/A	N/A
SE-ResNext [73]	Kinetics-600	ImageNet	21.3	38.7	N/A
Inception-ResNet [73]	Kinetics-600	ImageNet	23.2	N/A	N/A
Audio-only (ours)	Kinetics-600	-	**26.5**	44.7	14.2
VGG [20]	Kinetics-400	-	21.6	39.4	N/A
Audio-only (ours)	Kinetics-400	-	**24.8**	**43.3**	14.2

Table 4.7: **Results of Audio-only models.** VGG* model results are taken from "iTXN" submission from Baidu Research to ActivityNet challenge, as documented in this report [73].

Hierarchical audiovisual synchronization. We study the effectiveness of hierarchical audiovisual synchronization in Table 4.6c. We use AVSlowFast with and without AVS, and vary the layers for multiple losses. We observe that adding AVS as an auxiliary task is beneficial (+0.6% gain). Furthermore, having synchronization loss at multiple levels slightly increases the performance (without extra cost during testing). This suggests that it is beneficial to have a feature representation that is generalizable across audio and visual modalities and hierarchical AVS could facilitate producing such representation.

Per-class analysis on Kinetics. Comparing AVSlowFast to SlowFast (77.0% *vs.* 75.6% for 4×16, R50 backbone), classes that benefited most from audio include ["*dancing macarena*" +24.5%], ["*whistling*" +24.0%], ["*beatboxing*" +20.4%], ["*salsa dancing*" +19.1%] and ["*singing*" +16.0%], etc. Clearly, all these classes have distinct sound signatures to be recognized. On the other hand, classes like ["*skiing (not slalom or crosscountry)*" -12.3%], ["*triple jump*" -12.2%], ["*dodgeball*" -10.2%] and ["*massaging legs*" -10.2%] have the largest performance loss, as sound of these classes tend to be much less correlated the action.

Audio-only classification. To understand the effectiveness of our Audio pathway, we evaluate it in terms of Audio-only classification accuracy on Kinetics (in addition to Kinetics-400, we also train and evaluate on Kinetics-600 to be comparable to methods that use this data in challenges [73]). In Table 4.7, we compare our Audio-only network to several other audio models. We observe that our Audio-only model performs better than existing methods by solid margins (+3.3% top-1 accuracy on Kinetics-600 and +3.2% on Kinetics-400, compared to best-performing methods), which demonstrates the effectiveness of our Audio pathway design. Note

also that unlike some other methods in Table 4.7, we train our audio network from scratch on Kinetics, without any pretraining.

4.4 Experiments: AVA Action Detection

In addition to the action classification tasks, we also apply AVSlowFast models on action detection which requires both localizing and recognizing actions.

Dataset. The AVA dataset [84] focuses on spatiotemporal localization of human actions. Spatiotemporal labels are provided for one frame per second, with people annotated with a bounding box and (possibly multiple) actions. There are 211k training and 57k validation video segments. We follow the standard protocol [84] of evaluating on 60 classes. The metric is mean Average Precision (mAP) over 60 classes, using a frame-level IoU threshold of 0.5.

Detection architecture. We follow the detection architecture introduced in [62], which is adapted from Faster R-CNN [196] for video. During training, the input to our audiovisual detector is $\alpha_F T$ RGB frames sampled with temporal stride τ and spatial size 224×224, to SlowFast pathways, and the corresponding log-mel-spectrogram covering this time window to Audio pathway. During testing, the backbone feature is computed fully convolutionally with RGB frame of shorter side being 256 pixels [62], as is standard in Faster R-CNN [196].

Results. We compare to several other existing methods in Table 4.8. AVSlowFast, with both R50 and R101 backbones, outperforms SlowFast with a consistent margin of ~1.2%, and only increases FLOPs[2] slightly, *e.g.* for R50 *by only 2%*, whereas going from SlowFast R50 to R101 (without audio) increases computation significantly by 180%.

Interestingly, the ActivityNet Challenge 2018 [73] hosted a separate track for multiple modalities but no team could achieve gains using audio information on AVA. Our result shows, for the first time, that audio can be beneficial for action detection, where spatiotemporal localization is required, even with a very low computation overhead of just 2%.

For system-level comparison to other approaches, Table 4.8 shows that AVSlowFast achieves state-of-the-art performance on AVA under Kinetics-400 pretraining.

[2]FLOPs are for fully-convolutional inference of a clip with 256×320 spatial size for AV/SlowFast models, test-time computational cost is proportional to this.

model	inputs	AVA	pretrain	val mAP	GFLOPs
I3D [84]	V+F			15.6	N/A
ACRN, S3D [222]	V+F			17.4	N/A
ATR, R50+NL [111]	V+F			21.7	N/A
9-model ensemble [111]	V+F			25.6	N/A
I3D+Transformer [75]	V			25.0	N/A
LFB, + NL R50 [252]	V	v2.1	K400	25.8	N/A
LFB, + NL R101 [252]	V			26.8	N/A
SlowFast 4×16, R50	V			24.3	65.7
AVSlowFast 4×16, R50	A+V			25.4	67.1
SlowFast 8×8, R101	V			26.3	184
AVSlowFast 8×8, R101	A+V			**27.8**	210
SlowFast 4×16, R50	V			24.7	65.7
AVSlowFast 4×16, R50	A+V	v2.2	K400	25.9	67.1
SlowFast 8×8, R101	V			27.4	184
AVSlowFast 8×8, R101	A+V			**28.6**	210

Table 4.8: **Comparison on AVA detection**. Both AVSlowFast and SlowFast use 8×8 frame inputs. For R101, both also use NL blocks [248].

For comparisons with future work, we show results on the newer v2.2 of AVA, which provides updated annotations. We see consistent results as for v2.1. As for per-class results, we found classes like ["*swim*" +30.2%], ["*dance*" +10.0%], ["*shoot*" +8.6%], and ["*hit (an object)*" +7.6%] has the largest gain from audio, which makes sense as these are some of the classes with the most distinct sound signatures.

Per-class analysis on AVA We compare per-class results of AVSlowFast to its SlowFast counterparts in Fig. 4.4. Classes with largest absolute gain (marked with bold black font) are "*swim*", "*dance*", "*shoot*", "*hit (an object)*" and "*cut*". Further, the classes "*push (an object)*" (3.2×) and "*throw*" (2.0×) largely benefit from audio in relative terms (marked with orange font in Fig. 4.4). As expected, all these classes have strong sound signature that are easy to recognize from audio. On the other hand, the largest performance loss arises for classes such as "watch (e.g., TV)", "read", "eat" and "work on a computer", which either do not have a distinct sound signature ("read", "work on a computer") or have strong background noise sound ("watch (e.g.,

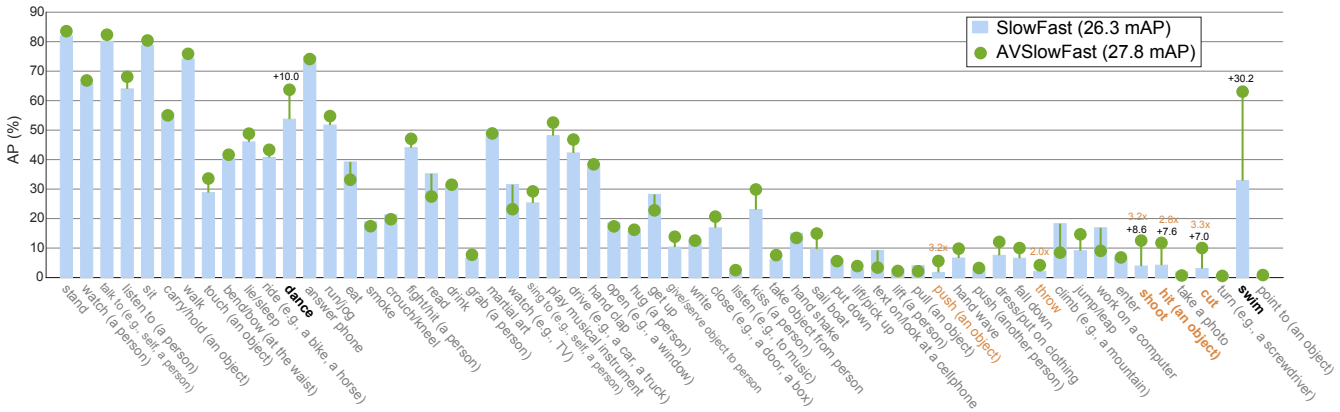

Figure 4.4: **AVA per-class average precision**. AVSlowFast (27.8 mAP) *vs.* its SlowFast counterpart (26.3 mAP). The highlighted categories are the 5 highest absolute increases (**bold**) and top 5 relative increases over SlowFast (**orange**). Best viewed in color with zoom.

TV)"). We believe explicitly modeling foreground and background sound might be a fruitful future direction to alleviate these challenges.

4.5 Experiments: Self-supervised Learning

To further study the generalization of AVSlowFast models, we apply it to self-supervised learning (SSL). The goal here is not to propose a new SSL pretraining task. Instead, we are interested in how well a self-supervised video representation can be learned with AVSlowFast using existing tasks. We use the audiovisual synchronization [6, 133, 175] and image rotation prediction [74] (0°, 90°, 180°, 270°; as a four-way softmax-classification) losses as pretraining tasks. With the AVSlowFast weights learned on Kinetics-400, we then re-train the last *fc* layer of AVSlowFast on UCF101 [215] and HMDB51 [137] following standard practice to evaluate the SSL feature representation. Table 4.9 lists the results. The results indicate that features learned by AVSlowFast are significantly better than baselines including the recently introduced CBT method [223] (+23.4% for UCF101 and +14.6% for HMDB51), which also uses ROT as well as a contrastive bidirectional transformer (CBT) loss by pretraining on the larger Kinetics-600. We believe this is clearly demonstrating that the strength of the architecture also generalizes to the self-supervised setting. In addition, we also ablate the contribution of individual tasks of AVS and ROT in Table 4.9 (bottom). On UCF101, SlowFast/AVSlowFast trained under either ROT or AVS objective show strong individual performance, while the combination of them perform the

method	inputs	#param	FLOPs	pretrain	UCF	HMDB
Shuffle&Learn [161, 223]	V	58.3M	N/A	K600	26.5	12.6
3D-RotNet [112, 223]	V	33.6M	N/A	K600	47.7	24.8
CBT [223]	V	N/A	N/A	K600	54.0	29.5
AVSlowFast 4×16	A+V	38.5M	36.2G	K400	76.8	41.0
AVSlowFast 8×8	A+V	38.5M	63.4G	K400	77.4	42.2
AVSlowFast 16×4	A+V	38.5M	117.9G	K400	**77.4**	**44.1**
ablation (split1)						
SlowFast 4×16 (ROT)	V	33.0M	34.2G	K400	71.9	**42.0**
AVSlowFast 4×16 (AVS)	A+V	38.5M	36.2G	K400	73.2	39.5
AVSlowFast 4×16	A+V	38.5M	36.2G	K400	**77.0**	40.2

Table 4.9: **Comparison using the linear classification protocol**. We only train the the last *fc* layer after self-supervised pretraining on Kinetics-400 (abbreviated as K400). Top-1 accuracy averaged over three splits is reported when comparing to previous work (top), results on split1 is used for ablation (bottom). All SlowFast models use use R50 backbones with $T \times \tau$ sampling.

best. Whereas on the smaller HMDB51, all three variants of our method perform similarly well and audio seems less important. Another aspect is that, although many previous approaches on self-supervsied feature learning focus on reporting number of parameters, the FLOPs are another important factor to consider – as shown in Table 4.9 (top), the performance keeps increasing when we take higher temporal resolution clips by varying $T \times \tau$ (*i.e.* larger FLOPs), even though model *parameters remain identical*.

Although we think the linear classification protocol serves as a better method to evaluate self-supervised feature learning (as features are frozen and therefore less sensitive to hyper-parameter settings such as learning schedule and regularization, especially when these datasets are relatively small), we also evaluate by fine-tuning all layers of AVSlowFast on the target datasets to compare to a larger corpus of previous work on self-supervised feature learning. Table 4.10 shows that AVSlowFast achieves competitive performance comparing to prior work under this setting. When using this protocol, we believe it is reasonable to also consider methods that train multiple layers on UCF/HMDB from scratch, such as optical-flow based motion streams [60, 208]. It is interesting that this stream, despite being an AlexNet-like model [30], is comparable or better, than many newer models, pretrained on (the large) Kinetics-400 using self-supervised learning

method	inputs	#param	pretrain	UCF101	HMDB51
Shuffle & Learn [161]	V	58.3M	UCF/HMDB	50.2	18.1
OPN [145]	V	8.6M	UCF/HMDB	59.8	23.8
O3N [64]	V	N/A	Kinetics-400	60.3	32.5
3D-RotNet [112]	V	33.6M	Kinetics-400	62.9	33.7
3D-ST-Puzzle [127]	V	33.6M	Kinetics-400	65.8	33.7
DPC [86]	V	32.6M	Kinetics-400	75.7	35.7
CBT [223]	V	N/A	Kinetics-600	79.5	44.6
Multisensory [175]	A+V	N/A	Kinetics-400	82.1	N/A
AVTS [132]	A+V	N/A	Kinetics-400	85.8	**56.9**
VGG-M motion [60, 208]	V	90.7M	-	83.7	54.6
AVSlowFast	A+V	38.5M	Kinetics-400	**87.0**	54.6

Table 4.10: **Comparison for Training all layers.** Results using the popular protocol of fine-tuning all layers after self-supervised pretraining. Top-1 accuracy averaged over three splits is reported. We use AVSlowFast 16×4, R50 for this experiment. While this protocol has been used in the past, we think it is suboptimal for evaluation of self-supervised representations, as the training of all layers can significantly impact performance; *e.g.* an AlexNet-like VGG-M motion stream [60, 208] can perform among state-oft-the-art self-supervised approaches, without any pretraining.

techniques. More details including training schedules are provided in §4.7.5.

4.6 Remarks

This chapter has presented AVSlowFast Networks, an architecture for integrated audiovisual perception. We show the effectiveness of the AVSlowFast representation with state-of-the-art performance on six datasets for video action classification, detection, and self-supervised learning tasks. We hope that AVSlowFast, as a unified audiovisual backbone, will foster further research in video understanding.

4.7 Appendix – Implementation Details

We provide more details on our implementation and experimental protocols in this section.

4.7.1 Details: Kinetics Action Classification

We train our models on Kinetics from scratch without any pretraining. Our training and testing closely follows [62]. We use a synchronous SGD optimizer and train with 128 GPUs using the recipe in [82]. The mini-batch size is 8 clips per GPU (so the total mini-batch size is 1024). The initial base learning rate η is 1.6 and we decrease the it according to half-period cosine schedule [154]: the learning rate at the n-th iteration is $\eta \cdot 0.5[\cos(\frac{n}{n_{\text{max}}}\pi) + 1]$, where n_{max} is the maximum training iterations. We adopt a linear warm-up schedule [82] for the first 8k iterations. We use a scale jittering range of [256, 340] pixels for R101 model to improve generalization [62]. To aid convergence, we initialize all models that use Non-Local blocks (NL) from their counterparts that are trained without NL. We only use NL on res_4 (instead of res_3+res_4 used in [248]).

We train with Batch Normalization (BN) [108], and the BN statistics are computed within each 8 clips. Dropout [96] with rate 0.5 is used before the final classifier layer. In total, we train for 256 epochs (60k iterations with batch size 1024, for ~240k Kinetics videos) when $T \leq 4$ frames, and 196 epochs when the Slow pathway has $T > 4$ frames: it is sufficient to train shorter when a clip has more frames. We use momentum of 0.9 and weight decay of 10^{-4}.

4.7.2 Details: EPIC-Kitchens Classification

We fine-tune from Kinetics pretrained AVSlowFast 8×8, R101 (w/o NL) for this experiment. For fine-tuning, we freeze all BNs by converting them into affine layers. We train using a single machine with 8 GPUs. Initial base learning rate η is set to 0.01 and 0.0006 for verb and noun. We train with batch size 32 for 24k and 30k for verb and noun respectively. We use a step wise decay of the learning rate by a factor of 10× at 2/3 and 5/6 of full training. For simplicity, we only use a single center crop for testing.

4.7.3 Details: Charades Action Classification

We fine-tune from the Kinetics pretrained AVSlowFast 16×8, R101 + NL model, to account for the longer activity range of this dataset, and a per-class sigmoid output is used to account for the mutli-class nature of the data. We train on a single machine (8 GPUs) for 40k iterations using a batch size of 8 and a base learning rate η of 0.07 with one 10× decay after 32k iterations. We

use a Dropout rate of 0.7. For inference, we temporally max-pool scores [62, 248]. All other settings are the same as those of Kinetics.

4.7.4 Details: AVA Action Detection

We follow the detection architecture introduced in [62], which is adapted from Faster R-CNN [196] for video. Specifically, we set the spatial stride of res_5 from 2 to 1, thus increasing the spatial resolution of res_5 by $2\times$. RoI features are then computed by applying RoIAlign [92] spatially and global average pooling temporally. These features are then fed to a per-class, sigmoid-based classifier for multi-label prediction. Again, we initialize from Kinetics pretrained models and train 52k iterations with initial learning rate η of 0.4 and batch size 16 (we train across 16 machines, so effective batch size 16×16=256). We pre-compute proposals using an off-the-shelf Faster R-CNN person detector with ResNeXt-101-FPN backbone. It is pretrained on ImageNet and the COCO human keypoint data and more details can be found in [62, 252].

4.7.5 Details: Self-supervised Evaluation

For self-supervised pretraining, we train on Kinetics-400 for 120k iterations with per-machine batch size 64 across 16 machines and initial learning rate 1.6, similar to §4.7.1, but with step-wise schedule. The learning rate is decayed with $10\times$ three times at 80k, 100k and 110k iterations. We use linear warm-up (starting from learning rate 0.001) for the first 10k iterations. As noted in §4.5, we adopt the curriculum learning idea for audiovisual synchronization [132] to first train with easy negatives for the first 60k iterations and then switch to a mix of easy and hard negatives ($1/4$ hard, $3/4$ easy) for the remaining 60k iterations. The easy negatives com from different videos, while hard negatives have a temporal displacement of at least 0.5 seconds.

For the "linear classification protocol" experiments on UCF and HMDB, we train 320k iterations (echoing [131], we found it beneficial to train long iterations in this setting) with an initial learning rate of 0.01, a half-period cosine decay schedule and a batch size of 64 on a single machine with 8 GPUs. For the "train all layers" setting, we train 80k / 30k iterations with batch size 16 (also on a single machine), an initial learning rate of 0.005 / 0.01 and a half-period cosine decay schedule, for UCF and HMDB, respectively.

4.7.6 Details: Kinetics-Sound dataset

The original 34 classes selected in [6] are based on an earlier version of the Kinetics dataset. Some classes are removed since then. Therefore, we use the following 32 classes that are kept in current version of Kinetics-400 dataset: “blowing nose”, “blowing out candles”, “bowling”, “chopping wood”, “dribbling basketball”, “laughing”, “mowing lawn”, “playing accordion”, “playing bagpipes”, “playing bass guitar”, “playing clarinet”, “playing drums”, “playing guitar”, “playing harmonica”, “playing keyboard”, “playing organ”, “playing piano”, “playing saxophone”, “playing trombone”, “playing trumpet”, “playing violin”, “playing xylophone”, “ripping paper”, “shoveling snow”, “shuffling cards”, “singing”, “stomping grapes”, “strumming guitar”, “tap dancing”, “tapping guitar”, “tapping pen”, “tickling”.

Chapter 5

Disentangle Visual Factors using Unlabeled Videos

As discussed in Chapter 1, there is a great potential of learning from videos to facilitate better visual understanding in general. In this chapter, we present an approach that can learn disentangled representations for object identity and pose with the help of unlabeled videos.

Consider the NYC street scene shown in Fig. 5.1 (left). As a human, it is not difficult to imagine what a red sedan would look like in place of the yellow taxis. This is likely because we have been exposed to thousands of different cars in various poses in our lifetime, and have learned how to *disentangle* a car's identity from its pose. In this chapter, we propose to learn a model to perform this task – specifically, synthesizing a novel pose of an object instance conditioned on the pose of a different reference object (see Fig. 5.1, right), without any labels.

This task would require the model to disentangle the object's identity and pose. For example, in Fig. 5.1, in order to encode the ID information of the red sedan, the model needs to capture the appearance and shape that is unique to that specific car instance, *independent* of pose. At the same time, the model also needs to encode the pose information specified by the taxis (the pose reference image), *independent* of identity. It can then combine the identity of the red sedan with the pose of the taxis to create a new image with the desired pose.

There has been a long line of research on learning disentangled representations for objects [49, 79, 100, 106, 106, 193, 206, 229, 267]. Early works like Tenenbaum & Freeman [229] operate in a fully-supervised setting in which the factors of interest (content and style in their case)

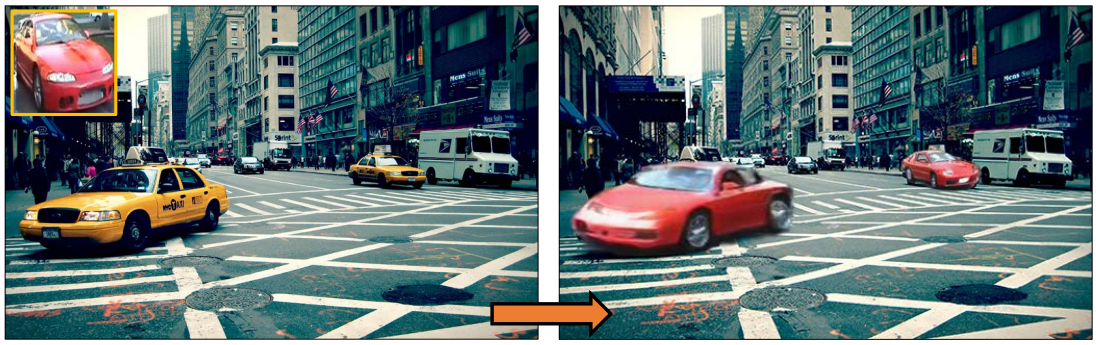

Figure 5.1: We propose an unsupervised method to disentangle identity (red sedan) and pose (taxis) of objects for image generation.

are annotated for each training image. We instead aim to solve this task in an *unsupervised* setting, without any pose or identity annotations. Unsupervised disentanglement of identity and pose is an extremely challenging problem, since the two factors are highly intertwined. For example, shape constitutes an important part of an object's identity – to distinguish a side-view van from a side-view sedan, we need to analyze their specific shape differences. On the other hand, the difference between pose and shape is often subtle and interdependent – as the pose of the car changes, so does its shape. In this work, we are particularly interested in utilizing the disentangled representation to manipulate objects (e.g., changing pose) in the image, which makes the quality of the disentangled representation even more important, since an incompletely-disentangled representation might lead to undesired artifacts (e.g., the identity of the object changes when changing its pose).

To disentangle the identity and pose factors in an unsupervised way, recent image generation methods either introduce cyclic constraints [79, 100, 106, 116] (similar in spirit to cycleGAN [279]) or inject priors on the representation based on domain knowledge [206]. Though promising, these methods typically only work well when there is no large change of pose in the objects. The reason is quite intuitive: due to the lack of direct supervision (i.e., ground-truth target images), the supervisory signals provided by either the proposed constraints or the prior on the representation are often insufficient to produce visually pleasing results.

In this work, we take a different approach. We utilize *unlabeled videos* to automatically construct *training triplets*, each consisting of an identity reference image, a pose reference image, and a pseudo ground-truth target image, to train our model. The requirement for the

pseudo ground-truth target is that it should consist of an object that has the identity of the identity reference and the pose of the pose reference. We exploit the fact that frames in a short video clip are highly likely to contain instances of the same object, to sample the identity reference and target image. We then find a nearest neighbor of the target image in pose space to construct the pose reference image. Though an imperfect approximation to the true ground-truth, directly feeding input/output pairs to our model provides a much stronger supervisory signal than only enforcing cyclic constraints, and enables it to achieve the desired disentanglement. To supplement the direct supervision and further encourage disentanglement and realism, we propose to optimize two novel loss functions – disentanglement loss and pixel verification loss. For the disentanglement loss, we construct two explicit constraints that force the identity encoder to only capture identity information and the pose encoder to only capture pose information – the same identity feature is used to generate two different poses of the same object, while two different objects with the same pose are produced from the same pose feature. The pixel verification loss promotes realism by exploiting the fact that, a pixel in the generated image should be able to trace back to its root in the identity image.

Our model is a novel conditional adversarial learning framework based on Generative Adversarial Networks (GANs) [81]. The network consists of an identity encoder, a pose encoder, and a target decoder, and is trained with aforementioned loss functions to learn to disentangle identity and pose. We conduct extensive experiments on the challenging YouTube-BoundingBoxes [191] dataset, and demonstrate better realism, diversity, and ID/pose disentanglement, compared to existing unsupervised approaches.

5.1 Related Work

Disentangled representation Unsupervised methods for disentangling factors of variation typically employ cyclic constraints [49, 100, 106, 116, 146]. One limitation with cyclic constraints is that, though necessary (i.e., they would be satisfied with perfect disentanglement), they are often not sufficient for generating high-quality disentangled representations. We instead propose to employ a simple yet effective procedure to retrieve direct pseudo targets during training, to enforce a much stronger constraint. Others place disentanglement in the context of

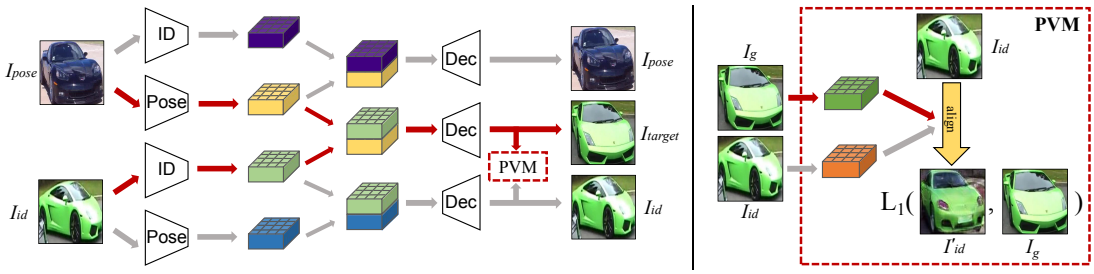

Figure 5.2: An illustration of the generator. Our generator takes as input both the identity reference image I_{id} and the pose reference image I_{pose}, and tries to generate an output image that matches I_{target}, which has the same identity as I_{id} but with the pose of I_{pose}. Notice how the pose encoded feature is used to generate both I_{target} and I_{pose}, so it cannot contain any identity information. Likewise, the identity encoded feature is used to generate both I_{target} and I_{id}, so it cannot contain any pose information. Furthermore, we propose a novel pixel verification module (PVM, details shown on the right) which computes a verifiability score between I_g and I_{id}, indicating the extent to which pixels in I_g can be traced back to I_{id}.

cross-domain translation [79, 106], which requires a clear definition of *domains*. For example, to disentangle the identity and pose of cars, one would need to define the pose as content (according to the definition in [106]), and define one domain for each car identity, which would require one encoder-decoder pair for each identity. In contrast, our work does not need to define domains.

Some work learn disentangled representations by enforcing explicit priors (e.g., a canonical appearance and a deformation field) [206] or by focusing on specific domains like faces with AU [188], identity [15], or pose and identity [234] annotations. In contrast, we avoid manually parameterizing the form of the representation or making strong domain-specific assumptions, and grant our model more freedom to learn directly from data. Finally, [194] learn a disentangled representation via a visual-analogy task, whereby a query image is transformed analogously to an example pair of reference images. Unlike visual-analogy, which takes three input images, our task only requires two, serving as the ID/pose reference respectively.

Novel view synthesis from a single RGB image is a highly under-determined problem that requires 3D understanding of the object. Some of the disentanglement work mentioned above take novel view synthesis as their application [15, 49, 116, 194, 234]. In addition to these, there are also works tackling this problem from different perspectives. For example, by making use of a large stock of 3D CAD models [126, 195, 268] potentially with a large amount of human involvement [126]. Others perform view synthesis in HOG space rather than RGB space [219].

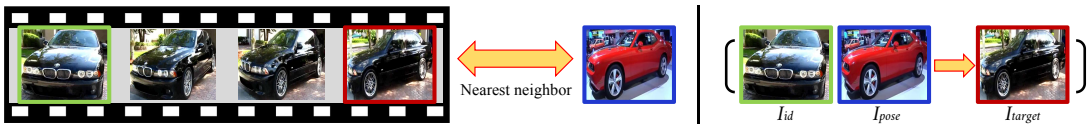

Figure 5.3: Constructing ID, pose, and target training triplets. With this procedure, we automatically obtain supervision to train our model.

More recent works train CNNs to function like a graphics rendering engine [56, 138, 267] or learn appearance flow to synthesize novel views [278]. Unlike these approaches, our method does not require any 3D shape models, human intervention, or ground-truth training examples.

Conditional image-to-image translation The most successful image-to-image translation algorithms are based on Generative Adversarial Networks (GANs) [81]. Examples that learn in a supervised setting—with annotated paired input and output images—include Pix2Pix [109], Pix2PixHD [245], and [32]. Unsupervised approaches leverage cycle-consistency [279], learn a shared latent space between domains [35, 106], or impose constraints for disentangling the factors [116]. Our work leverages a large collection of *unlabeled videos* to automatically construct pseudo ground-truth targets. In this way, we can exploit the advantages of the supervised setting, without having to annotate any images (i.e., remain unsupervised).

5.2 Approach

Our goal is to learn a model that takes as input two images A and B, which are different object instances of the same category (e.g., car), and generate a new image with A's *identity* and B's *pose*. Importantly, in our setting, we do not have any identity or pose annotations (i.e., the images are unlabeled) during both training and testing.

5.2.1 Network architecture

Generator To factorize identity and pose, we use a two-branch generator network that processes the two streams of inputs separately. As shown in Fig. 5.2 (red arrows), the ID encoder takes as input the ID reference image, and processes it into a feature map that aims at capturing pose-invariant *identity* information. Analogously, the Pose encoder is responsible for generating a feature map that captures identity-invariant *pose* information from the input pose reference image.

The concatenated ID and pose feature maps (along the channel dimension) are fed into the decoder. In this way, the decoder has the necessary ID/pose information to generate the target. Overall, our generator can be expressed as:

$$I_g = G(E_i(I_{id}), E_p(I_{pose})),$$

where I_{id} is the ID reference image and I_{pose} is the pose reference image. E_i and E_p are the ID and pose encoders and G is the decoder. The ID encoder consists of consecutive `Conv - ReLU` blocks whereas the Pose encoder consists of consecutive `Conv - Norm - ReLU` blocks. We add normalization (Instance Normalization following [245]) to the Pose encoder to remove instance-specific feature means and variances which are highly-correlated with object identity [106]. For the decoder, we follow the architecture used in [245] (from residual blocks and onwards), except we replace transposed convolutions with `Upsample - Conv` to mitigate checkerboard artifacts [169].

Discriminator For the output to preserve both *realism* and *identity*, we set up two discriminators. The first is the Real/Fake discriminator D_{real}, which takes in as input a single RGB image and is trained to classify it as real or fake. The job of the generator is therefore to fool the discriminator, thus pushing the generated image I_g to look as real as possible. The second discriminator D_{pair} focuses on preserving the object's identity in the generation and is trained to classify whether an input pair shares the same identity or not. The generator is thus trained to match the identity of the generated image to that of the input ID image. Following [245], we adopt a multi-scale (2 scale) discriminator, which together help enforce realism both locally (e.g., specific object details) and globally (e.g., overall shape). However, rather than using a patch discriminator as in [245], we use a single global image discriminator (i.e., output a single scalar). In Sec. 5.3.2, we show that this leads to better realism of the generated images for our data.

5.2.2 Constructing ID-pose-target training triplets

The key difference between our work and previous unsupervised disentanglement works (e.g., [49, 79, 100, 106, 116, 146]) is that rather than rely only on indirect cyclic constraints, we instead construct a *pseudo ground-truth* target image I_{target} so that we can train the model in a supervised way (but still without any labels). In this way, we can use I_{target} to facilitate the disentanglement

of pose and identity and directly guide the model to generate the correct image.

We first sample two images from the same video clip as I_{id} and I_{target}. The assumption is that these images will contain the same object instance, which is generally true for short clips. We then retrieve a nearest neighbor of I_{target} from other videos (images can also be used) using a pre-trained convnet, to serve as the pose reference image I_{pose}. Fig. 5.3 illustrates this process. The key insight is that retrieving objects with the same pose is much easier than retrieving objects with the same identity – objects with the same pose share a large amount of edges, which can be well-captured with an off-the-shelf feature extractor (we use the `conv4` feature map of VGG-16 [209] pre-trained on ImageNet); see Fig. 5.4. Therefore, we can safely treat I_{target} as the pseudo ground-truth target for $G(E_i(I_{id}), E_p(I_{pose}))$. Although an approximation to the real ground-truth, we show in the experiments that using direct supervision from I_{target} is highly effective for training our model for image generation.

To ensure diversity in the pose pairs that are sampled, we propose a balanced sampling scheme. We first cluster all images into M different poses by performing k-means on their `conv4` features, and then initialize an M-by-M count matrix C with zeros (rows/columns correspond to unique pose of I_{id}/I_{pose}). Each time we sample an id-pose-target triplet, we choose the pose pair corresponding to the entry with the lowest value in C, and increment that count by one. This diversifies the poses of the (I_{id}, I_{pose}) training pairs.

5.2.3 Loss functions

To generate images that are realistic and identity/pose-preserving, we use the following loss functions.

Disentanglement loss To directly supervise our model with the pseudo ground-truth target, we minimize the L_1 difference between our model's generation and the target:

$$\mathcal{L}^1_{dis} = ||I_{target} - G(E_i(I_{id}), E_p(I_{pose}))||_1.$$

However, since there are many possible solutions for minimizing this loss, it alone will not necessarily enforce the desired disentanglement. To ensure that the ID/Pose encoder only encodes information about identity/pose, in addition to generating I_{target}, we also ask our model

Figure 5.4: Retrieving nearest neighbors with `conv4` feature of VGG-16 network. As can be seen, the nearest neighbors in `conv4` space resemble the pose of the query very well.

to reconstruct I_{id} and I_{pose}:

$$\begin{aligned}\mathcal{L}_{dis}^{2} =& ||I_{id} - G(E_i(I_{id}), E_p(I_{id}))||_1 \\ &+ ||I_{pose} - G(E_i(I_{pose}), E_p(I_{pose}))||_1.\end{aligned}$$

As shown in Fig. 5.2, this will force the ID encoder to not capture any pose information since its output is used to generate two targets with distinct poses (I_{id} and I_{target}). Simultaneously, the pose encoder is forced to not capture any ID information since the same pose feature is used to generate two images with different identities (I_{pose} and I_{target}). Our final disentanglement loss is:

$$\mathcal{L}_{dis} = \mathcal{L}_{dis}^{1} + \mathcal{L}_{dis}^{2}.$$

Instead of directly computing the L_1 distance in RGB space, we adopt the perceptual loss [113] as it also captures the distance at a semantic level.

Pixel verification loss Recall that our final generated image should preserve the identity as the input ID reference image. This implies that *for (almost) every pixel in our generation, we should be able to trace it back to the ID image*. For example, for a car's front light pixel in our generation, we should be able to find the same front light pixel in the ID image, if our generation correctly preserves its identity. This will only be untrue when there are unobserved parts in the ID image that need to be generated. However, we can still assume that even for those unseen parts, their low-level color and texture (which are generally shared throughout an image) could still be taken from some weighted combination of pixels in the ID image.

To this end, we propose a novel pixel verification module (PVM) that matches every pixel in the generation back to the ID image. Specifically, PVM first transforms the ID image to spatially

Figure 5.5: Our generation results for `car`. The top row shows the input pose images, while the leftmost column shows the input ID images. From these results, it is clear that our method has learned to disentangle the identity and pose. As for each ID image, we can change it to different poses, while maintaining its identity.

align it to the generated image. For this, it matches each pixel in I_g to each pixel in I_{id} using their features of the penultimate layer of the decoder (i.e., right before they get converted into RGB space; same spatial resolution as RGB image), which results in a weight matrix $W \in \mathbb{R}^{P \times P}$, where P is the total number of pixels in both I_{id} and I_g, and W_{ij} indicates the affinity between the i-th pixel in I_g and the j-th pixel in I_{id}. To make each row of W sum to 1, we pass W through a softmax function along its rows. Then, PVM transforms the ID image by:

$$I'_{id}(i) = \sum_j W(i,j) \cdot I_{id}(j), \ \forall i$$

The result I'_{id} is the ID image aligned to the generation. The PVM then computes the L_1 difference between I'_{id} and I_g to compute the pixel verification loss:

$$\mathcal{L}_{pv} = ||I'_{id} - I_g||_1,$$

where a low $\mathcal{L}_{pv}$ value indicates high degree of verifiability in the generation. Thus minimizing this loss ensures that every generated pixel can be traced back to the ID image, and thus acts as an extra constraint for improving realism and identity preservation. We note PVM is related to the MatchTrans module proposed in [256], however PVM does not constrain a local search window, thus allowing larger pose changes.

Auxiliary video clip classification loss Prior GAN reserach [80, 170] demonstrate the benefit of auxiliary classification tasks when training the discriminator. In our setting, we work with

unlabeled videos and thus do not have any annotated labels. However, we can use the clip index as a proxy for identity label by assuming that e.g., cars within the same/different clip correspond to the same/different instance. In practice, these assumptions hold true for short clips.

To train the auxiliary classifier, we add one fully-connected layer with the output dimension as the number of video clips to the penultimate layer of the discriminator, and train it to classify which video clip the input image comes from (using cross-entropy loss). We add this auxiliary classification loss to both D_{real} and D_{pair}, and denote them as L_{aux}^{real} and L_{aux}^{pair}, respectively. This helps in two ways: On the one hand, it helps instill the concept of multiple object identities to the discriminator, which then teaches the generator to generate more diverse outputs. On the other hand, it directly forces the generator to generate results with the desired identity, as specified by the auxiliary classifier.

Adversarial loss Finally, to fuel the adversarial game between the generator and the discriminators (D_{real} and D_{pair}), we employ the following adversarial losses:

$$\mathcal{L}_{GAN}^{real} = \log(D_{real}(I_{target})) + \log(1 - D_{real}(I_g)),$$
$$\mathcal{L}_{GAN}^{pair} = \log(D_{pair}(I_{id}, I_{target})) + \log(1 - D_{pair}(I_{id}, I_g)),$$

where $I_g = G(E_i(I_{id}), E_p(I_{pose}))$. $\mathcal{L}_{GAN}^{real}$ and $\mathcal{L}_{GAN}^{pair}$ encourage realism and conditional identity-preservation (i.e., I_g and I_{id} to have the same identity), respectively.

Total loss Combining all the loss functions, we form the following min-max optimization problem:

$$\min_{G} \max_{D_{real}, D_{pair}} \mathcal{L}_{dis} + \mathcal{L}_{pv} + \mathcal{L}_{aux} + \mathcal{L}_{GAN},$$

where $\mathcal{L}_{aux} = \mathcal{L}_{aux}^{real} + \mathcal{L}_{aux}^{pair}$ and $\mathcal{L}_{GAN} = \mathcal{L}_{GAN}^{real} + \mathcal{L}_{GAN}^{pair}$. We alternate between fixing the generator G and training the discriminators D to maximize the losses, and fixing D and training G to minimize the losses.

5.3 Experiments

In this section, we compare to state-of-the-art baselines, and perform ablative studies to demonstrate the effectiveness of our pixel-verification loss, disentanglement loss, and the choice of discriminator output.

Figure 5.6: Our generation results for `bus`. The top row shows the input pose images, while the leftmost column shows the input ID images.

Datasets. We use 3 classes (`Car`, `Bus`, and `Truck`) from YouTube-BoundingBoxes [191] (YTBB) video dataset. We choose these classes as they represent unique challenges. Specifically, cars can have very different shapes (e.g., comparing sedans, SUVs, vans), buses generally have lots of textures (logos, paints), whereas the appearance of trucks exhibits large uncertainty (it is hard to predict one view from another). Since these are real-world YouTube videos, they are quite challenging – fast motion, drastic illumination change, compression artifacts, etc. We use Faster-RCNN trained on MS COCO to detect instances of the object in the videos. We retain detections which have 0.9 confidence or higher, which removes inaccurate and strongly-occluded instances. This results in 2233/186, 3008/302, 1833/137 clips for training/testing on `Car`, `Bus`, and `Truck`, respectively.

Baselines. **Pix2pixHD** [245]: state-of-the-art conditional image-to-image translation approach. For the input, we directly concatenate the ID and Pose image[1] over the channel axis (i.e., a 6-channel input). The model is trained to output the 3-channel RGB image corresponding to the pseudo ground-truth target. We use the authors' implementation.

FusionImage [116]: solely relies on cyclic constraints which, as we will show, are not strong enough to induce the desired disentanglement due to the challenging nature of our data (e.g., drastic pose changes). For fair comparison, we adopt our generator and discriminator architectures (based on the state-of-the-art architectures from Pix2PixHD) and only change the

[1]We also tried concatenating the ID image and a one-hot encoding of the pose image, obtained from clustering and it does not work as well.

Figure 5.7: Our generation results for `truck`. The top row shows the input pose images, whereas the leftmost column shows the input ID images. Note how the generation in column 1 (in blue dotted box) flipped the pose by 180 degrees, exhibiting incorrect frontal views.

losses to those defined in [116].

DrNet [49]: pits an identity classifier to classify, using pose features, whether two images are from the same video (i.e., have the same identity), and a pose encoder that tries to maximally confuse the identity classifier. This way, it achieves disentanglement by forcing the pose encoder to not capture identity information. DrNet does not have a target image and therefore only makes use of indirect supervisory signals. We implement DrNet with our encoder and decoder architecture for fair comparison.

Evaluation metrics. We create an evaluation set of 5000 randomly-sampled identity-pose-target triplets from the test set of each category, and manually verify the correctness of the target images. We use the following metrics.

LPIPS distance [272]: For a generated image $I_g = G(E_i(I_{id}), E_p(I_{pose}))$, we measure its LPIPS distance to the target image I_{target}. This metric essentially captures two aspects: 1) how realistic I_g is, since it has to be realistic to have a low distance to the real image I_{target}; 2) how well I_g preserves the identity of I_{id} and pose of I_{pose}, since I_{target} is a ground-truth combination of the two.

Fréchet Inception Distance (FID) [95]: measures both realism and diversity of the generated data by comparing its distribution to that of real data using the `pool3` features of the Inception-v3 network [227]. We compute FID between the set of generated images $\{I_g^1, I_g^2, ..., I_g^N\}$

and the corresponding target images $\{I^1_{target}, I^2_{target}, ..., I^N_{target}\}$.

ID and Pose preservation scores: We measure the preservation of ID and Pose factors as another way to evaluate disentanglement. For the ID preservation score, we fine-tune an ImageNet pre-trained ResNet-50 on our data to minimize: $\max(f(x_1) \cdot f(y) - f(x_1) \cdot f(x_2) + m, 0)$, where f extracts a L_2-normalized feature from the penultimate layer of ResNet-50, x_1 and x_2 are two instances from the same video clip whereas y is from another clip. This triplet loss enforces the affinity between positive pairs (frames from the same clip) to be higher than that between a negative pair by a margin m. During evaluation, we compute the affinity between the generated image I_g and identity image I_{id} (we sigmoid the affinity to [0, 1]). It is harder to evaluate pose preservation since we do not have pose annotations in general. Thus, we only evaluate on `car` by making use of the Multi-View Car Dataset [177], which has pose annotations, to train a car pose classifier and compute the pose preservation score in a similar way.

Implementation details. We train our model using Adam [129] with an initial learning rate of 10^{-4}. For data augmentation, we apply standard color jittering (brightness, contrast, saturation) and randomly perturb the bounding box around the objects. To stabilize training, we perform model averaging following [270]. We generate 128x128 images. We will open-source our code/data/models upon acceptance.

5.3.1 Qualitative Results

We first present our model's results for `car`, `bus`, and `truck` in Figs. 5.5, 5.6 and 5.7. For each category, the leftmost column shows the input ID reference images, while the first row shows the input pose reference images. Each entry in the matrix corresponds to our model's generated image (e.g., entry C3 is result with ID image C and Pose image 3 as input). First, it is clear from these results that our model has learned to disentangle the identity and pose, so that it can generate new images with the identity of one ID image and the pose of many different pose images (see the generated cars in rows A and B of Fig. 5.5). As mentioned before, buses usually have lots of textures (logos, paints, etc.) which makes preserving identity trickier. Still, one can see that our method preserves the fine texture details well (e.g., the blue paint on the bottom of the bus in C1 of Fig. 5.6). `truck` is more challenging due to the uncertainty of its appearance (e.g., it's sometimes impossible to infer a truck's side-view given only its frontal view). Still, our

	car				bus			truck		
	LPIPS	FID	ID	Pose	LPIPS	FID	ID	LPIPS	FID	ID
Ours	**0.33**	**18.57**	**0.63**	0.65	**0.37**	**16.67**	**0.63**	**0.35**	**25.25**	**0.62**
Pix2PixHD [245]	0.36	28.99	0.60	0.64	0.40	35.88	0.60	0.39	75.03	0.58
FusionImg [116]	0.50	238.57	0.52	0.57	0.49	230.42	0.56	0.43	68.11	0.60
DrNet [49]	0.48	24.45	0.52	**0.69**	0.48	38.04	0.52	0.46	27.79	0.53
Ours w/o L_{pv}	0.34	18.92	0.63	0.65	0.37	19.61	0.63	0.36	26.34	0.62
w/o L^2_{dis}	0.35	19.49	0.63	0.64	0.38	24.38	0.63	0.38	36.85	0.62
w/ PatchGAN	0.33	26.61	0.63	0.64	0.37	32.10	0.63	0.35	40.48	0.62

Table 5.1: Quantitative results on YTBB `car`, `bus`, and `truck`. For LPIPS and FID, lower is better; for ID and Pose score, higher is better. As explained in the text, we only have pose score for `car` since we do not have supervised pose classifiers for bus and truck.

method is able to capture the gist of the pose while maintaining the identity. One failure mode we observe is that our model can get confused with similar-looking views (e.g., it incorrectly generates a frontal view in column 1 of Fig. 5.7) and this is partly because of the error from the nearest neighbor search during the triplet generation process.

Comparison to baselines. We next show comparisons to baselines in Fig. 5.8. Note that these are representative examples for each method. First, FusionImage [116] experiences severe mode collapse as its output is completely independent of the pose input. DrNet [49] simply copies the content of the pose image, losing entirely the identity from the ID image. Pix2PixHD [245] is able to disentangle the ID and Pose factors. However, our results look more realistic (1st row) and preserves the identity/pose better (2nd and 5th row respectively). We believe the reason for the failures of FusionImage and DrNet is because the indirect cyclic constraints they optimize are not sufficient to induce disentanglement for our difficult data, and therefore lead to degenerate solutions (mode collapse/identity mapping). Unlike Pix2PixHD, our method not only optimizes the generated image to be similar to the target, but also encourages our two encoders to carry disentangled representations and thus leads to overall better generation results.

5.3.2 Quantitative Results

We quantitatively evaluate our method against the baselines, measuring realism, diversity, and id/pose disentanglement. We also investigate the pixel verification loss, disentanglement loss, and choice of discriminator output.

How real are our generated images? Our method outperforms all baselines for all cate-

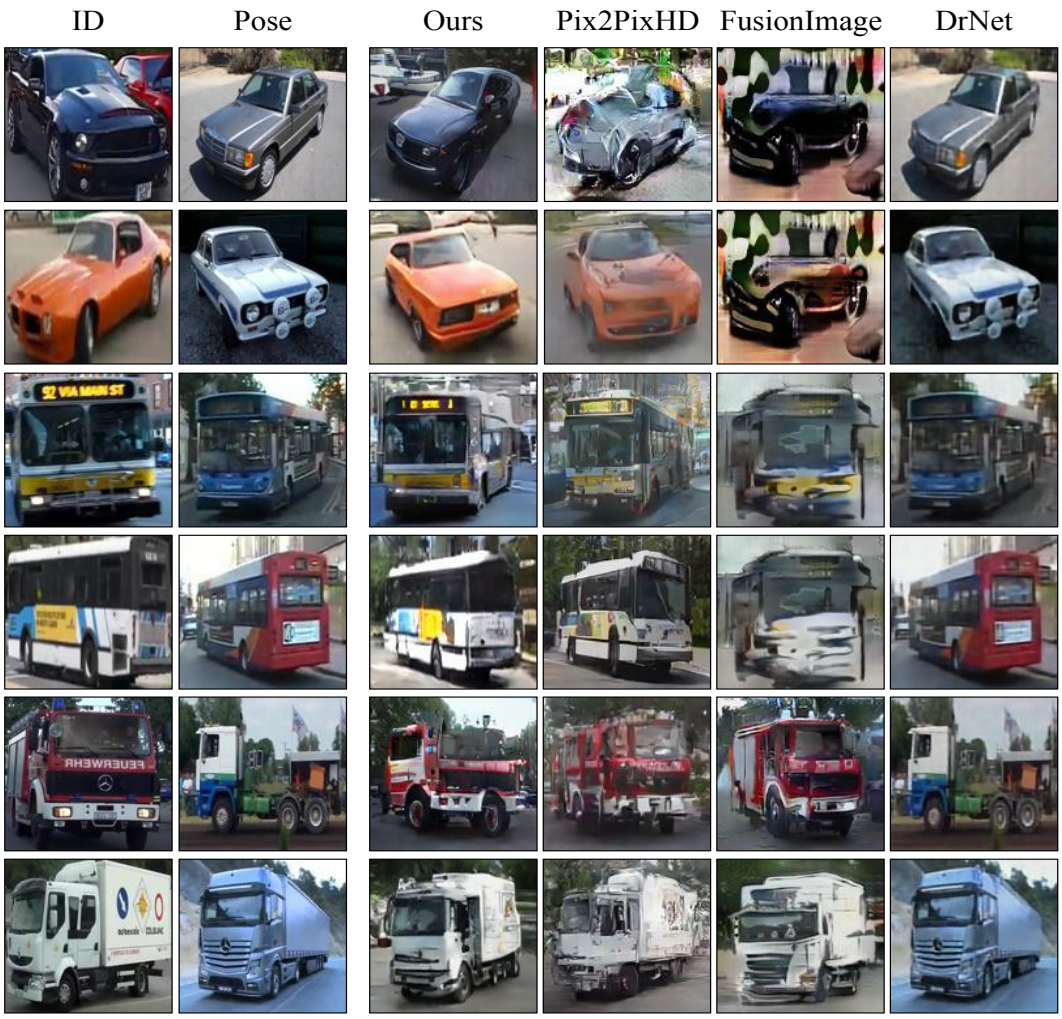

Figure 5.8: Comparison to baselines. The first and second column show the input ID and pose images respectively. Please see text for details.

gories in FID (see Table 5.1), which shows that our generated images are more realistic and diverse compared to those of the baselines.

How well do our generated images match the target image? By comparing the LPIPS distance, we can see that our results are closest to the ground-truth target.

How well does our model disentangle id and pose? Our method also outperforms the baselines on both identity and pose[2] preservation scores, which implies the highest degree of disentanglement between the two factors.

These results are telling in two aspects. Compared to FusionImage and DrNet, our approach

[2]Except for DrNet, which achieves a better pose preservation score since it incorrectly copy-pastes the pose image.

clearly benefits from having a direct supervisory signal. On the other hand, the importance of explicitly enforcing disentanglement is revealed when comparing our approach to Pix2PixHD. Overall, both the qualitative and quantitative results demonstrate that our method is able to generalize to different object categories despite their various unique challenges.

Ablation studies. We next perform ablation studies (results shown in the bottom part of Table 5.1). First, we remove the pixel-verification loss L_{pv}. This consistently hurts FID by a sizable margin, which suggests that pixel-verification is effective in terms of boosting the overall realism and diversity of the generation. If we also remove part of the disentanglement loss L^2_{dis} (so we are only left with the perceptual loss L^1_{dis}), the performance further drops, both in terms of FID and LPIPS, which again demonstrates that our disentanglement loss is helping to learn a good disentangled representation. Finally, an interesting finding is that for our data, the PatchGAN adopted in Pix2Pix [109] is not as effective (as reflected by FID) as having a scalar output discriminator, which has the receptive field of the entire image. Qualitative inspection reveals that this is because a large receptive field helps generate images that respect the global structure better (e.g., produce straighter lines).

5.3.3 Application: Image Composition

One potentially useful application of our approach is image composition. In standard image composition approaches [25, 45, 142, 228], users are required to find an image of the object that is in the *correct* pose (or a 3D CAD model matching its identity, which is even harder). For example, to replace all three cars in Fig. 5.9 (bottom row) with a sports car, images of the sports car facing three different directions would be needed. With our approach, we only need a single image of the desired car, *in any view*. The results in Fig. 5.9 are produced by alpha-blending our generation into the image.

5.4 Remarks

In this chapter, we proposed a novel approach for disentangling the identity and pose of two input objects, and combining those factors to generate a new object with one's identity and the other's pose. We demonstrated state-of-the-art generation results on cars, buses, and trucks.

Our method is not without limitations. Although better than the baselines, our results are not

Figure 5.9: Image composition application. Left: original image, Right: modified image with our generations alpha-blended in.

perfect and one prominent failure mode is confusion amongst similar looking poses (e.g., frontal and rear view trucks). This is partly due to the error in nearest neighbor search for generating the training triplets. We believe this issue could be mitigated with a much larger dataset, since our approach can find the nearest neighbor pose image from any image or video. Another improvement that is desired is in the realism of our generation, as artifacts are visible upon close inspection. Thus, our method would benefit from future advances in generative modeling.

Chapter 6

Weakly-supervised Visual Grounding of Phrases with Linguistic Structures

So far, we have mainly focused on how to effectively learn from videos, with minimal supervision. However, there are many other data modalities that could be utilized to facilitate visual learning. Among them, natural language has many distinct advantages. As constructed by humans to communicate with each other effectively and efficiently, language is rich in semantics and structures (i.e., compositionality). In this chapter, we will discuss how we can transfer the linguistic structures (in the form of syntactic parsing tree) into the visual domain for the task of weakly supervised visual grounding of phrases.

Visual recognition research has made tremendous strides in recent years, achieving unprecedented performance in various tasks including image classification [91, 136, 209], object detection [78, 196], semantic segmentation [89, 152], and image captioning [34, 122, 265]. However, traditional supervised frameworks for these tasks often rely on large datasets with expensive bounding box or pixel-level segmentation annotations. As the field pushes toward solving larger-scale and more complex problems, obtaining massive annotated datasets is becoming a critical bottleneck.

Weakly-supervised approaches that learn from image-level supervision have been proposed to alleviate the need for expensive and unwieldy annotation. Most previous work use category tags to train models that can localize objects without any bounding box or segmentation annotations [38, 63, 120, 122, 173, 182, 210, 212, 214, 249]. While great progress has been made, learning

from a list of category tags ignores the rich semantics and structure in natural language that we humans use to describe visual data. For example, in Fig. 6.1, a tag-based description would simply list {man, sandwich, table} whereas a natural language description might say "a man that is cutting sandwich on a table". Importantly, the natural language description provides *structure*, which can benefit a weakly-supervised learning algorithm. For instance, the description implies that "a man" and "sandwich" occupy *spatially-distinct* regions in the image, and that a visual grounding (localization) of the entire sentence should be the *union* of the groundings of "a man" and "sandwich on a table". By exploiting these constraints and regularities that are shared between linguistic and visual data, localization in the challenging weakly-supervised setting can be facilitated.

In this chapter, we propose a weakly-supervised visual localization approach that learns from image-level descriptions (i.e., without any region-to-phrase correspondence annotations). In particular, we aim to create spatial *attention masks* that produce localizations at the pixel-level. Our key idea is to utilize the rich structure in a natural language description by transforming it into a hierarchical parse tree of phrases (see Fig. 6.2). In this way, we can extract two types of linguistic structural constraints for visual grounding: (1) *compositionality* of attention masks among children and their parent phrases (e.g., the mask of "a hand with a donut" should be the union mask of "a hand" and "with a donut"), and (2) *complementarity* among the attention masks between sibling phrases (mask of "a grey cat" should be spatially-disjoint with that of "starring at a hand with a donut"). Furthermore, the parse tree representation augments the total amount of supervision and enables learning from linguistic descriptions at various levels, i.e., from words, to phrases, and eventually to the full sentence, for the same image.

Our model is an end-to-end deep neural network that consists of a language encoder, a visual encoder, a semantic projection module, and loss modules. The language/visual encoder process input phrases/images into vector representations, which are then projected into a common semantic space by the semantic projection module. In addition to the discriminative loss which directly enforces the annotated image-phrase pairs to be close to each other in semantic space, we propose a novel structural loss which enforces the linguistic structural constraints to be satisfied when generating localized attention masks in the images. Given a test image, our model can

Figure 6.1: We propose to localize phrases in images, by exploiting linguistic structure. For example, from the phrase "a man that is cutting sandwich", we can infer that "a man" and "sandwich" should be *exclusive* to each other spatially. At the same time, they should *jointly occupy* the spatial extent of "a man that is cutting sandwich". We enforce these structural constraints as part of a novel deep network architecture for weakly-supervised visual grounding of phrases.

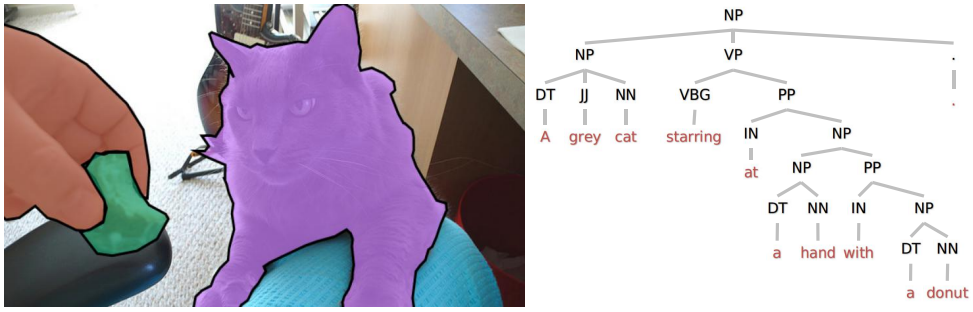

Figure 6.2: An illustration of the concept. We exploit structures present in natural language to provide regularities and constraints for grounding free-form language on images. Note that we **do not** use any ground-truth masks during training.

generate an attention mask for any arbitrary phrase without needing access to the structures of the linguistic input (e.g., it can localize single word inputs).

Our work is most related to [197], which learns to localize textural phrases in a weakly-supervised setting, and [122], which learns to align text to regions for image captioning. However, unlike our approach, these methods do not explicitly exploit the hierarchical structure in the language description, and instead treat it simply as a sequence of words. Furthermore, their visual localizations are in the form of bounding boxes, which are insufficient for representing objects with irregular shapes. Finally, our work is also related to approaches in image captioning [158, 265] and visual question answering [46, 276]. While these tasks require localization as a sub-task to be successful, the end goal itself is not localization. Thus, these approaches do not leverage any structural *localization* constraints as we do.

Contributions. To our knowledge, we are the first to leverage the hierarchical structure of natural language descriptions for weakly-supervised visual localization. We design a novel deep network with a new *structural loss*, which makes use of the parse tree structures induced by the descriptions. The structural loss is combined with a discriminative loss, which enforces that attended image regions and phrases are consistently encoded, to produce pixel-level spatial attention masks. We demonstrate the effectiveness of our approach through localization experiments on the Microsoft COCO and Visual Genome datasets [150, 210].

6.1 Related work

Weakly-supervised learning with categorical labels/tags. Weakly-supervised visual learning approaches focus on learning granular detectors given only coarse annotations. This is an extremely useful paradigm since granular annotations (e.g., bounding boxes, segmentations) are much more costly compared to coarse image-level annotations.

Most previous weakly-supervised approaches work with categorical labels/tags. For example, weakly-supervised detection methods aim to train object/attribute detectors with only image-level labels (e.g., whether an object/attribute exists in the image or not) instead of bounding boxes [18, 39, 50, 179, 211, 214, 244, 254]. Despite being a much harder problem, there are also previous efforts in weakly-supervised semantic segmentation, which requires per-pixel predictions [182, 183, 264]. Although these methods have demonstrated promising results, the type of annotation used is not very "natural". Referring back to Fig. 6.2, it would be somewhat unnatural for someone to tag the image with all depicted objects – e.g., "cat, hand, donut". Instead, it would be much more natural to tag it with a descriptive sentence, similar to what one would expect in a social media post. Assuming this is the case, paring down a sentence to a set of object tags seems to be sub-optimal as the process loses valuable linguistic structure present in the original sentence. Unlike these two lines of work, we perform weakly-supervised learning with *sentence-level* supervision.

Vision and language. The interplay of vision and language has been studied extensively in recent years, partly because research in both vision and language has matured and has made tremendous progress with the help of deep learning. Given the natural connection between

language and vision (e.g., visual language grounding) it is no surprise that multimodal learning that considers both carries significant promise.

Image captioning [34, 54, 114, 122, 130, 158, 265] has received a great amount of attention in the past two years. Most models adopt an encoder-decoder architecture, in which the encoder encodes information from the image (usually using a CNN) into a hidden state and the decoder then decodes it into a sequence of word tokens (a sentence). The decoder often takes a form of an RNN (e.g., LSTM) and is conditioned on previously generated word tokens. Visual question answering (VQA) [5, 46, 155] is another popular problem in this space. In VQA, the encoder, in addition to the image, takes a question (often encoded by an RNN) and the decoder decodes the answer.

In both image captioning and VQA, localization of relevant regions helps in generating captions/answers, which motivates many recent models to incorporate *attention mechanisms* to focus on the relevant spatial regions as part of the decoding process [46, 122, 155, 158, 265]. However, since localization is not the final goal, most of these models are either not optimized for it explicitly or simply take off-the-shelf strongly-supervised object detectors for localization. In contrast, our goal is to perform weakly-supervised *localization* directly, and we propose a novel structural loss to facilitate this task. Other than captioning and VQA, there are also works trying to localize phrases in images [101, 102, 186, 197, 240, 240, 243]. However, most existing work use strong supervision as the ground-truth correspondence between phrases and *image regions*, which is costly to acquire at scale. While Rohrbach et al. [197] work with weakly-annotated image-phrase pairs, they treat the phrase input as a sequence of tokens, whereas our approach *explicitly* makes use of the structure present in the sentence input.

Structure from language. We note that we are not the first to utilize linguistic structure for a vision task. In [243], the authors propose to model "partial match coreference" relations between phrases. However, the approach requires strong supervision and produces bounding boxes instead of per-pixel predictions. In the VQA approach of [4], the network structure is dynamically constructed using a question; however, it does not directly aim at localization. To our knowledge, we are the first to leverage the hierarchical structure in natural language descriptions for weakly-supervised visual localization.

6.2 Approach

Our goal is to train a model that takes as input a set of weakly-annotated image-sentence pairs (without any explicit region-to-phrase correspondence annotations), and learns to visually ground (i.e., localize) arbitrary linguistic phrases in the form of pixel-level spatial attention masks. The key idea is to transfer the linguistic structure to the image domain as constraints to guide the model to produce more accurate localizations. To this end, we propose a novel end-to-end deep network that encodes both the association of phrases and images (with a discriminative loss), as well as the structure of the phrases (with a structural loss).

6.2.1 Transforming a sentence into a parse tree

Given an image and its associated sentence description, we first transform the sentence into a parse tree with an off-the-shelf NLP parser [213], as shown in Fig. 6.2. In most cases, the structure in the parse tree can also be well-represented in the image (e.g., in Fig. 6.2, "A grey cat" and "a hand with a donut" should be exclusive to each other according to the linguistic structure, which is also true in the visual domain). We therefore transfer this structure to its corresponding image when visually grounding different nodes in the tree. In order to ensure that a node corresponds to a region in the image, we only consider nodes that contain at least one noun. This removes meaningless/ambiguous nodes like "with", "and", "on it", or "in it".

6.2.2 Network architecture

The proposed architecture for grounding phrases in an image is shown in Fig. 6.3. There are four parts to the architecture: visual encoder, language encoder, semantic embedding sub-network, and the loss functions.

The visual encoder and language encoder are responsible for encoding the raw input of images and phrases, respectively, into semantic representations. The semantic embedding sub-network projects the representations from both modalities to a *common semantic space* in which the visual and language data are directly comparable (i.e., enabling one to compute an image-phrase similarity). In addition to embedding an entire image into the semantic space, the semantic embedding sub-network also projects individual image regions into the common semantic space. To extract these regions a spatial *attention mask* over the image is computed.

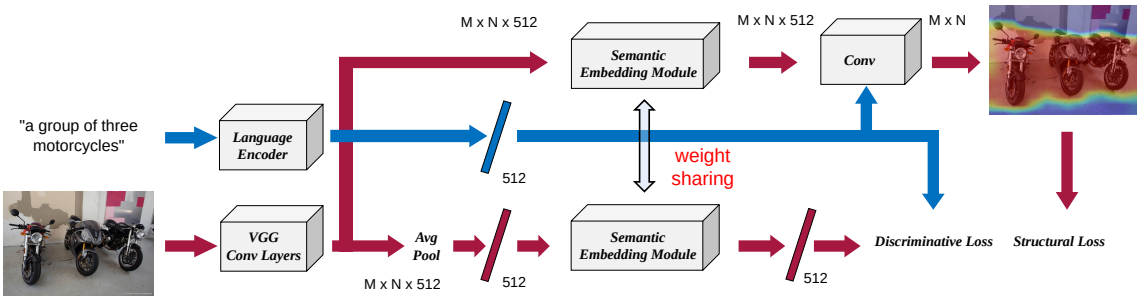

Figure 6.3: Our architecture consists of 4 submodules: visual encoder, language encoder, semantic embedding module, and loss functions. We adopt the convolutional layers of the VGG network as the visual encoder. For the language encoder, we use a two-layer LSTM network. The output of the language encoder directly lives in the semantic space, whereas the output of the visual encoder is projected into the semantic space by the semantic embedding module, which is a two-layer-perceptron with Dropout inserted in between the layers. Alongside projecting the visual feature of the full image, the semantic embedding module is also responsible for projecting the feature in each spatial location, to the embedding space: in this case, the output of the visual encoder bypasses average pooling and is directly fed into the embedding module. The embedding modules used for these two purposes share their weights. After projection, both the full image feature and the spatial feature are matched against the language codes, to generate a matching score and an attention mask, respectively. The matching score is used to compute the discriminative loss while the structural constraints are enforced onto the attention mask.

Finally, the image-phrase matching scores and attention masks are fed into the discriminative and structural loss modules, respectively, to optimize the network for learning semantics and localization.

In testing, the network takes a phrase as input, and outputs a corresponding attention mask to localize the phrase.

Visual encoding. We use a convolutional neural network (CNN) to encode the visual content in an image. Specifically, we adopt the VGG-16 network [209] (denoted as VGG), for its high performance and moderate computational cost. For our use, we remove the fully-connected layers and only keep the convolutional layers (`conv1_1` through `conv5_3`), so that we can preserve spatial information for localization. We initialize the network weights by pre-training on ImageNet [48].

Language encoding. We use a recurrent neural network (RNN) to encode the text descriptions. The network is able to take as input both short phrases like “A man” as well as long ones like “A man riding on the top of an elephant”. To better model long phrases, we adopt LSTM cells [97] in a two-layer RNN, with a Dropout module [217] inserted in between to prevent over-fitting.

For a phrase with tokens $\{W_1, W_2, ..., W_T\}$, its representation is computed as the RNN hidden vector at time step T. We pre-train the weights of our language encoder on a combined set of Google's Billion Words [31] dataset and COCO captions in the training (`train2014`) set with the next word prediction task.

Joint semantic embedding. We next describe how to project the visual and language representations into a common semantic embedding space. We directly take the output of the language encoder as the language code in the semantic space. For the output of the visual encoder, i.e., `conv5_3` map of VGG, we apply the semantic embedding sub-network to obtain the visual code in the semantic space.

Since we want our network to learn to localize the relevant image regions for a given phrase (recall we only have image-level phrase annotations), we feed the `conv5_3` feature map into a global *average-pooling* layer instead of a max-pooling layer, which is used in the standard VGG network. As argued in [277], average-pooling better preserves location information since it is forced to localize in order to maximize its response over the relevant image regions. A 2-layer-perceptron (fully-connected network) follows the average-pooled feature to compute a vector representation for the image. In order to match the scales of the language and visual codes, we add a batch normalization layer [108] at the end of the semantic embedding sub-network.

Generating spatial attention masks. In addition to projecting the `conv5_3` feature map into a visual code, the semantic embedding sub-network also serves to produce a spatial *attention mask* for each textual phrase. The attention mask is used both to enforce the structural loss constraints during learning (described in Sec. 6.2.3) as well as to produce localizations during testing. Specifically, we pass individual features at each spatial position of the `conv5_3` feature map through the network. The resulting attention mask shows how well the visual feature at each spatial location matches the input textual phrase.

6.2.3 Loss for weakly-supervised visual grounding

Finally, we introduce the loss functions that we use to induce visual grounding in a weakly-supervised setting. Our architecture is trained end-to-end with two loss terms:

$$L = L_{struct} + L_{disc}. \tag{6.1}$$

The structural loss L_{struct} enforces structure encoded in the text phrases to be satisfied by their respective visual attention masks. The discriminative loss L_{disc} enforces positive/negative image-phrase pairs to be close/far from each other in the semantic embedding space.

Structural loss We propose to exploit the rich hierarchical structure from the language input to help disambiguate the visual grounding of phrases. Unlike existing work that either treat the sentence descriptions as a list of nouns (e.g., [39, 50, 179, 214]) or use the entire sentence itself as-is (e.g., [54, 122, 197]), we leverage the structure in the descriptions to learn visual groundings on images.

Specifically, we exploit two types of structural constraints present in the parse tree: *parent-child* (PC) and *sibling-sibling* (SIB) constraints. The PC (or *inclusive*) constraint enforces the attention mask of any node in the tree to match the *union* of the attention masks of all of its children nodes. For example, the attention mask of "a hand with a donut" should encapsulate the masks of both "a hand" and "with a donut" as shown in Fig. 6.2. The SIB (or *exclusive*) constraint enforces the attention masks of siblings to be exclusive to each other (for example, "A grey cat" and "starring at a hand with a donut" in Fig. 6.2).

More formally, the structural loss is defined as $L_{struct} = \lambda_{PC} L_{PC} + \lambda_{SIB} L_{SIB}$, where λ_{PC} and λ_{SIB} are weights to balance the PC and SIB loss terms, and

$$L_{PC} = \frac{1}{|P|} \sum_{k \in P} ||A_k - \max_{l \in child(k)} A_l||^2, \tag{6.2}$$

$$L_{SIB} = -\frac{1}{|S|} \sum_{m \in S} \sum_{pixels \in A} W_m \cdot \log \frac{\max_n A_{m,n}}{\sum_n A_{m,n}}, \tag{6.3}$$

where A is the attention mask generated for a given phrase. P denotes the set of valid parent nodes, $child(k)$ returns all children nodes of parent node k, S is the set of all siblings, n indexes each node in sibling set m (i.e., nodes that are sibling to each other), $\max$ and $\log$ are computed per-pixel, and $(\cdot)$ is element-wise multiplication.

L_{PC} tries to bring the attention mask of a parent node and the *union* mask of all its children nodes to be close to one another, while L_{SIB} introduces competition such that the attention masks of sibling nodes are *exclusive* for every pixel. W_m is the average per-pixel attention over sibling set m: $\frac{1}{n} \sum_n A_{m,n}$. Its purpose is to enforce stronger exclusivity among sibling attention masks that have high-values in the same pixels. Without this term, the exclusivity is enforced

on *every* pixel equally regardless of whether it is relevant to the current sibling set. This can be problematic when all siblings produce low values for a given pixel (implying that the pixel is irrelevant to the current sibling set), since it will try to undesirably inflate the value for one of the sibling masks. Empirically, we find this to be the case.

Note that even though we only explicitly enforce the PC constraint between a parent and its immediate children, transitivity ensures that the constraints are carried out through all descendents. Also, since we only consider a node if it contains at least one noun, each node in a sibling pair (e.g., *NounPhrase-VerbPhrase*, *NounPhrase-PrepositionalPhrase*, or *VerbPhrase-PrepositionalPhrase*) is guaranteed to have its own "object-of-interest".

Discriminative loss This loss function is used to match the corresponding image-phrase pairs. Given an input image I_i and a set of corresponding phrases (both positive and negative ones) $\{P_i^1, P_i^2, ..., P_i^n\}$, we compute the discriminative loss as:

$$L_{disc} = -Y_i^j \cdot Sigmoid(\phi_V(I_i) \cdot \phi_L(P_i^j)), \tag{6.4}$$

where $Y_i^j \in \{-1, 1\}$ is the indicator variable denoting whether P_i^j is a negative/positive match to I_i, and $\phi_V(I)$ and $\phi_L(P)$ denote the visual and language code, respectively. The positive phrases are those in the parse tree associated with the input image I_i, while the negative phrases are randomly sampled from those in the parse tree associated with any other image. This loss tries to bring the visual and language codes for the positive image-phrase pair to be as close as possible, while separating the codes in the negative pair as much as possible. We measure the affinity between the codes with a dot-product in the semantic embedding space.

6.3 Results

Datasets. We conduct experiments on Visual Genome [135] and MS COCO [150] datasets. First, we evaluate on the Visual Genome dataset, which provides caption annotations for image regions. Since the image regions are annotated with bounding boxes, we use the "pointing game" [271] to evaluate the capability of our model to visually ground phrases in images. However, since the pointing game evaluation only cares about the maximum point and does not evaluate the full extent of the attention masks produced by our model, we further evaluate our model

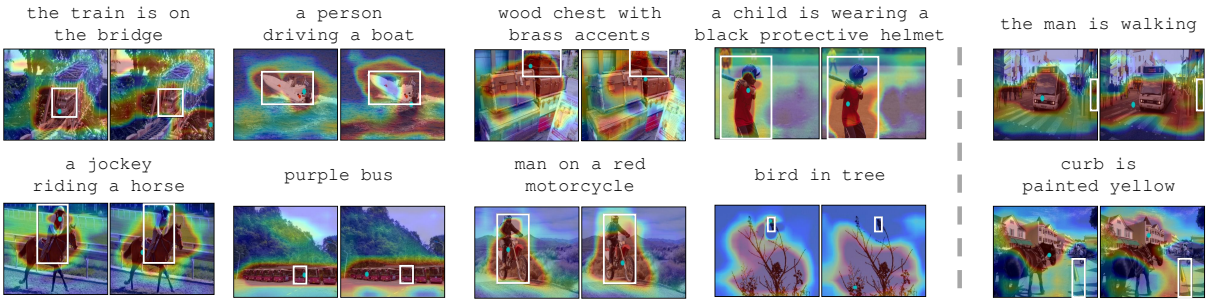

Figure 6.4: We show qualitative "pointing game" results on Visual Genome. We compare with the baseline model trained without structural constraints (*Disc-only*). In each image pair, the left is our result and the right is the baseline's result. The ground-truth bounding box is annotated with a white solid line, whereas the maximum point of our prediction is denoted with a cyan dot. The ground-truth phrase associated with each image is shown on top of each image. The last column shows difficult examples containing small or infrequent objects.

against the ground-truth category segmentation masks on COCO. For this, we treat category labels as free-form phrases to feed into our language encoder.

Baselines. We compare to a number of baselines: *Token* is a model that treats the natural language input as a list of object tags during training – we only take all the leaf nodes with noun POS tags in the parse tree to train the model. This baseline is meant to represent existing weakly-supervised learning methods that only learn from a list of category labels. *Disc-only* is a variant of our model that only has the discriminative loss (without the structural loss). *PC* and *SIB* are each also trained with the discriminative loss but *only* with either the parent-child constraint (Eq. 6.2) or the sibling constraint (Eq. 6.3). *Ours* is our full model with all loss terms.

Implementation details. We pre-train the visual (CNN) encoder on ImageNet classification, and pre-train the language (RNN) encoder using the language modeling (next word prediction) task on the combined set of Google Billion Words and COCO captions. For pre-training, we use the Adam [128] solver since it is has been demonstrated to be more robust to sparse updates, which are common in language tasks. We use SGD with a mini-batch size of 8 images and their associated phrases. In each batch, the positive samples are images and their corresponding phrases, whereas negative samples are formed by taking an image and sample phrases that do *not* correspond to the image. We find that fixing the language encoder after pre-training is important to avoid a degenerate solution in which all phrases collapse to almost the same encoding. For the weights in L_{struct}, we set $\lambda_{PC} = 0.01$ and $\lambda_{SIB} = 0.0001$ based on qualitative examples (see

supp. materials for details on the impact of λ_{PC} and λ_{SIB}).

6.3.1 Training on MS COCO

MS COCO is a large dataset designed for object detection, instance segmentation, and image captioning. We use the `train2014` and `val2014` sets, which contain 82,783 and 40,504 images, respectively, for training and validation. We train all variants of our model using images and associated image-level captions on the training set. We take the trained models and evaluate their localization accuracy using the "pointing game" metric and semantic segmentation.

6.3.2 Pointing game on Visual Genome

Visual Genome [135] is a recent effort to provide rich annotations on a subset of 328,000 images from MS COCO (it also annotates images from the Flickr30k dataset [186], which we do not use). To test the capability of our model in localizing *phrases*, we evaluate on the MS COCO validation set using the region-phrase annotations (i.e., one phrase corresponding to a bounding box in an image) provided by Visual Genome.

Since our approach outputs per-pixel predictions (via the attention mask) instead of bounding boxes, we cannot directly evaluate against the bounding box annotations. This is also why we cannot directly compare with the work of [197], which outputs bounding boxes rather than segmentation masks. We therefore instead use the *pointing game* evaluation metric [271]: For each phrase provided in the annotation, we pass the phrase, together with the image to which it associates, through our model and obtain the attention mask on the image. We then compute the maximum point on the attention mask, and check whether it falls inside the ground-truth bounding box. If yes, it is counted as a *Hit*; otherwise, it is a *Miss*. The final accuracy is $\frac{\#Hit}{\#Hit+\#Miss}$.

Table 6.1 shows the results. The model trained with tokens (*Token*) does not perform as well as the model trained using phrases (*Disc-only*), which demonstrates the value of using natural language for image localization. The parent-child constraint (*PC*) further improves performance over (*Disc-only*), which demonstrates the effectiveness of the parent-child constraint. While the sibling constraint (*SIB*) performs almost the same as *Disc-only*, when combined with the parent-child constraint, it produces a large boost in performance (*Ours*, our full model), which

	Random	Disc-only	Token	PC	SIB	Ours
Accuracy	0.115	0.230	0.222	0.236	0.231	**0.244**

Table 6.1: Localization accuracy as measured by the "pointing game" [271] on Visual Genome. Our model outperforms all baselines, including variants of our method that lack one or more loss terms. See text for details.

suggests that the constraints are complementary. Finally, we also add an extra baseline called *Random* as a sanity check. For every test image, we select 100 random points and compute the probability of the randomly sampled point falling inside the box. This baseline clearly performs the worst.

Fig. 6.4 shows qualitative example predictions on the Visual Genome dataset. Overall, our model generates quite accurate masks. For example, for "the train is on the bridge" our model accurately pinpoints the train and the bridge whereas the *Disc-only* baseline produces high responses on many irrelevant pixels. A similar thing happens for "a person driving a boat". Furthermore, in some cases, even though the maximum point of the baseline attention mask falls within the ground-truth bounding box, we can clearly see that the generated mask is not as clean as that of our model; e.g., for "a child is wearing a black protective helmet" and "man on a red motorcycle", the baseline model tends to generate leaky masks compared with our results. This is likely because the baseline model does not have any constraints to exploit other than the discriminative loss, whereas our model explicitly enforces structural priors onto the generated attention masks.

Finally, in the first two rows of Fig. 6.6, we show different visual groundings corresponding to different input phrases produced by our model for the same input image. For example, in the first image, our model generates entirely different groundings for "the clock tower is tall" and "buildings by the street". This result demonstrates that our model learns to focus on the right concepts instead of simply computing a language-independent saliency map.

6.3.3 Semantic segmentation on MS COCO

Since the "pointing game" only evaluates on the maximum prediction point, we next evaluate the full extent of our attention masks through a segmentation task. For this, we use the MS COCO segmentation annotations [150]. Note that only category labels are provided (e.g., "cat") for

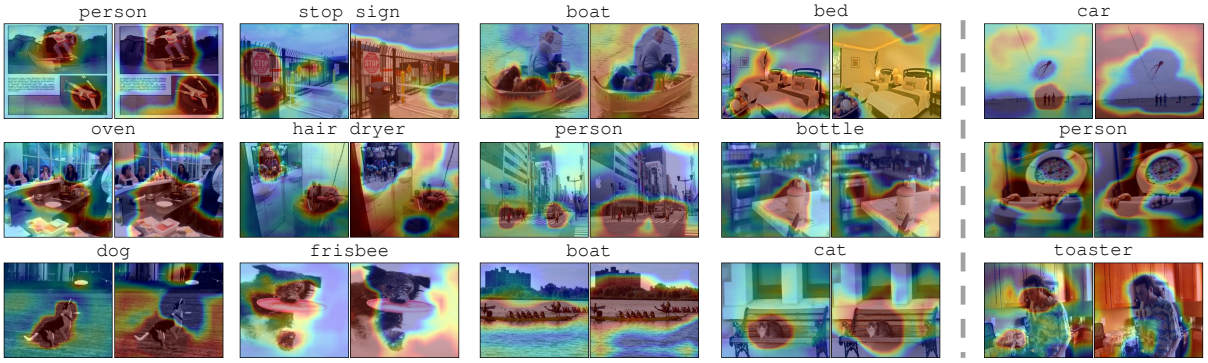

Figure 6.5: Segmentation results on the MS COCO segmentation task. In each image pair, we show the results of our model on the left and that of the *Disc-only* baseline on the right. The category label is shown on top of each image pair. The last column shows difficult examples containing small or infrequent objects. See text for details.

	IOU@0.3	IOU@0.4	IOU@0.5	Avg mAP
Disc-only	0.302	0.199	0.110	0.203
PC	0.327	0.213	0.118	0.219
SIB	0.316	0.203	0.114	0.211
Token	0.334	0.240	0.138	0.238
Ours	**0.347**	**0.246**	**0.159**	**0.251**

Table 6.2: Segmentation mAP on MS COCO across all 80 categories. Our model produces more accurate segmentations compared to alternate weakly-supervised baselines.

evaluating segmentation on MS COCO. In this case, it makes more sense to evaluate our model on semantic segmentation rather than instance segmentation, since our model will only take a single category label as the language input. We therefore merge instances of the same category into one ground-truth semantic segmentation mask, and evaluate on the validation set (the test set evaluates only instance segmentation and requires submission to a private server).

In order to perform semantic segmentation, we convert our continuous-valued attention masks to binary foreground/background predictions. For this, we simply binarize it with a threshold, which is set to be the medium value in the predicted attention mask range (i.e., $\theta = \frac{1}{2}(\max(A) - \min(A))$. We compute the resulting segmented region's prediction score as the average per-pixel attention score within the region. Finally, we compute the intersection-over-union (IOU) metric for the predicted foreground region against the ground-truth foreground mask.

Table 6.2 shows the segmentation results, in term of mean Average Precision (mAP) over all 80 MS COCO categories at different IOU thresholds. Both *PC* and *SIB* provide consistent improvement over *Disc-only* across different IOU thresholds. This demonstrates that both structural constraints effectively transfer their respective structure from the language domain to the visual domain. Moreover, with our full model *Ours*, which combines both *PC* and *SIB*, mAP is boosted even further. This again shows the complementarity of the parent-child and sibling constraints. Interestingly, the model trained only with noun tokens (*Token*) performs quite well, outperforming both *Disc-only* and *PC*/*SIB*. This is mainly because for this category semantic segmentation task, the language input is only nouns, and this can favor a model that is also trained using only the noun tokens. Despite this, our full model still outperforms the *Token* baseline and achieves the best performance among all methods. Further, we want to highlight that our model is much more general, as compared to the *Token* baseline; it can localize regions beyond simple objects or noun tokens (e.g., we can localize *adjective-noun* or even more complex *referring* phrases).

We show example segmentation results in Fig. 6.5. One can easily see the improvements brought by the structural constraints. First, our model localizes more accurately (e.g., for "dog" and "frisbee"), just like it does on the Visual Genome dataset. Second, we again observe the prominent behavior of our model that it tends to generate much cleaner attention masks compared to the *Disc-only* baseline (e.g., "person", "cat", "boat", "stop sign"). By explicitly encoding the linguistic structural constraints on the visual attention masks, our model is able to obtain more accurate localizations. The last column shows typical failure cases with small or infrequent objects in the image.

Finally, similar to what we showed on the Visual Genome dataset, the last two rows of Fig. 6.6 show different groundings generated by our model for the same image, given different category labels as language inputs. Our model is able to detect different objects in the same image.

6.3.4 Discussion of failure cases

While we have shown promising results for this very difficult task of weakly-supervised visual phrase grounding, there are still a few sources of difficulties for our model.

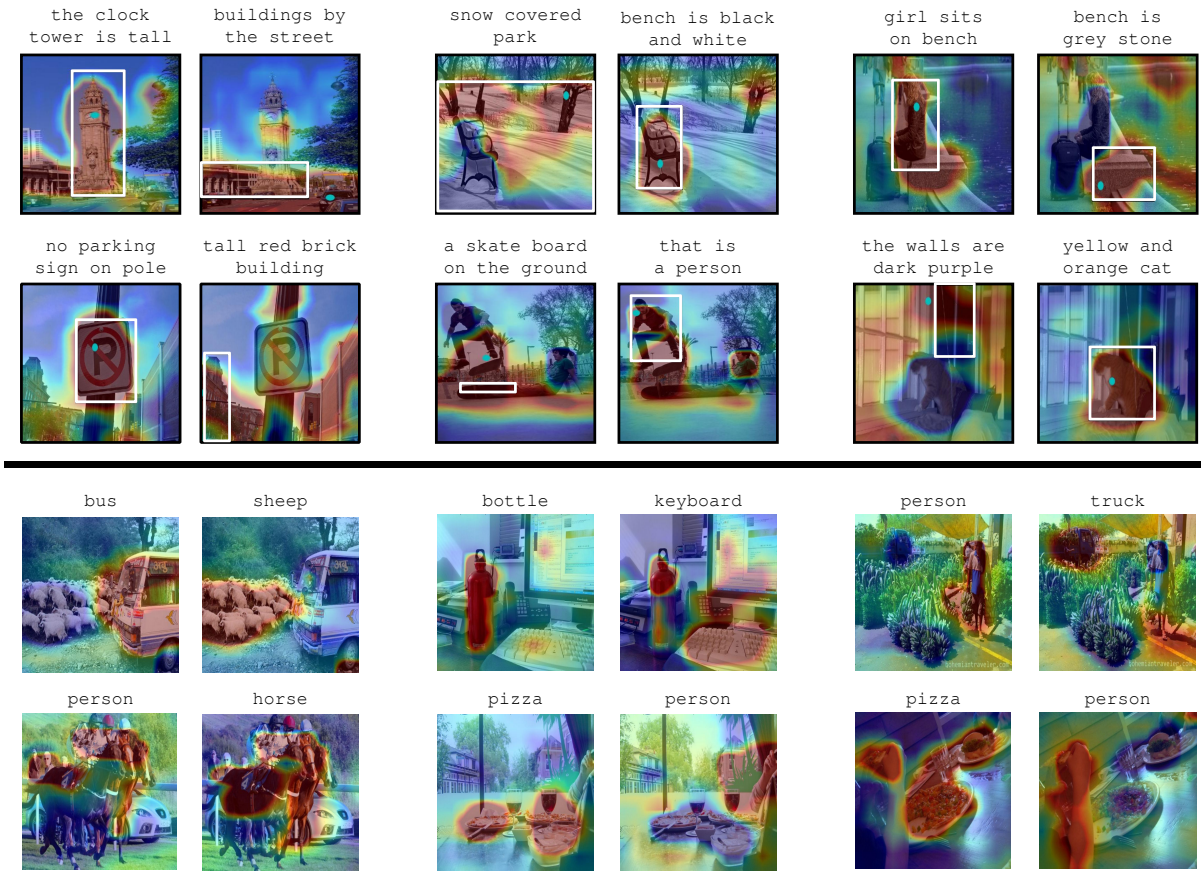

Figure 6.6: Here we show for the same image, visual groundings generated with different language inputs. The first two rows are visual grounding results on Visual Genome, with different phrases as language input. The last two rows are visual grounding results on MS COCO, with category labels as language input. In both cases, our model generates very different visual groundings corresponding to different language input, and correctly focuses on the relevant objects-of-interest.

First, the parse trees produced by the NLP parser can be wrong, which will make the resulting structural constraints insensible. Empirically, we observe parse tree errors in roughly 20% of the images. Another cause of failure is due to the language representation, in that the hidden vectors computed for an entire sentence might not correctly focus on the relevant objects (i.e., entities having corresponding image regions) in the sentence. The third difficulty is that our task is fundamentally weakly-supervised, which in itself limits how well one can do (particularly with respect to full supervision) with *limited training data*. We hypothesis that with a much larger weakly-annotated dataset, we could expect a performance boost on this challenging task.

6.4 Remarks

We presented a weakly-supervised approach that takes image-level captions, without any explicit region-to-phrase correspondence annotations, and learns to localize arbitrary linguistic phrases in images. We designed a novel end-to-end deep network with two new structural loss constraints: a parent-child inclusivity constraint and a sibling-sibling exclusivity constraint. Together, these constraints transfer the structure present in natural language to the visual domain so that the network can learn to produce more accurate visual localizations. Our experiments on the Microsoft COCO and Visual Genome datasets demonstrate that our approach produces more accurate localizations compared to several baselines that either do not consider any structure or consider only one of the constraints in isolation.

Chapter 7

Conclusion

In this thesis, we have presented algorithms that could learn from multimodal data (i.e., videos and language) for visual understanding, using minimal human supervision. To understand the semantics in videos, we have discussed approaches for various building-block tasks including unsupervised video object proposal, video object detection and audiovisual video recognition. Meanwhile, in order to minimize the amount of human supervision required for training visual understanding models, we have discussed various algorithms on weakly and self-supervised learning. For instance, we have discussed how to learn useful representations from unlabeled videos. Also, we talked about how to disentangle visual factors (identity and pose) for image generation using unlabeled videos. Finally, we presented an approach to transfer linguistic structures into the visual domain for the task of visual grounding of phrases.

Of course, these are the first steps on multimodal weakly/self-supervised learning. In the future, we would like to continue pushing the boundary of self-supervised learning of visual representations. Unlike for images, self-supervised video representations are still largely lagging behind their supervised counterparts. To close this gap, we believe in the power of multimodal learning as multimodality is what makes videos different (and more informative) compared to images. Also, as most self-supervised learning work focus on learning semantic representation from 2D pixels (or spatiotemporal voxels), self-supervised learning of 3D geometric understanding is largely under-explored. Since videos provide multiple views of the same object (implicit geometry), we would like to explore learning geometric understanding from unlabeled videos. Furthermore, as self-actuated movement is proven to be an essential part of acquiring visual

perception (e.g., "two kitten experiment" by Held and Hein [93]), we would like to explore this analogy in the context of visual representation learning and locomotor skills. A concrete direction that is particularly interesting is to develop a close-loop between "move smarter" and "see better" (i.e., train agents to "move better" in order to "see better" and vice versa).

REFERENCES

[1] http://image-net.org/challenges/LSVRC/2015/results#vid.

[2] Huda Alamri, Chiori Hori, Tim K Marks, Dhruv Batra, and Devi Parikh. Audio visual scene-aware dialog (avsd) track for natural language generation in dstc7. In *DSTC7 at AAAI Workshop*, 2018.

[3] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *PAMI*, 2012.

[4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016.

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[6] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *ICCV*, 2017.

[7] Relja Arandjelovic and Andrew Zisserman. Objects that sound. In *ECCV*, 2018.

[8] Pablo Arbelaez, Jordi Pont-Tuset, Jonathan Barron, Ferran Marques, and Jagannath Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[9] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*, 2017.

[10] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *NIPS*, 2016.

[11] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[12] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[13] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016.

[14] Dan Banica, Alexandru Agape, Andreea Ion, and Cristian Sminchisescu. Video object segmentation by salient segment chain composition. In *ICCV Workshops*, 2013.

[15] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Towards open-set

identity preserving face synthesis. In *CVPR*, 2018.

[16] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. Object level visual reasoning in videos. In *ECCV*, 2018.

[17] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

[18] T. Berg, A. Berg, and J. Shih. Automatic Attribute Discovery and Characterization from Noisy Web Data. In *ECCV*, 2010.

[19] Lynne Bernstein, Edward Auer, Jintao Jiang, and Silvio Eberhardt. Auditory perceptual learning for speech perception can be enhanced by audiovisual training. *Frontiers in Neuroscience*, 2013.

[20] Yunlong Bian, Chuang Gan, Xiao Liu, Fu Li, Xiang Long, Yandong Li, Heng Qi, Jie Zhou, Shilei Wen, and Yuanqing Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017.

[21] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *PAMI*, 2001.

[22] William Brendel and Sinisa Todorovic. Video Object Segmentation by Tracking Regions. In *ICCV*, 2009.

[23] T. Brox and J. Malik. Object Segmentation by Long Term Analysis of Point Trajectories. In *ECCV*, 2010.

[24] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *PAMI*, 2011.

[25] Peter J Burt and Edward H Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)*, 1983.

[26] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *CVPR*, 2010.

[27] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[28] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose

estimation with iterative error feedback. In *CVPR*, 2016.

[29] Anna Llagostera Casanovas, Gianluca Monaci, Pierre Vandergheynst, and Remi Gribonval. Blind audiovisual source separation based on sparse redundant representations. *IEEE Transactions on Multimedia*, 2010.

[30] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[31] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[32] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[33] Xinlei Chen and Abhinav Gupta. Spatial memory for context reasoning in object detection. In *ICCV*, 2017.

[34] Xinlei Chen and C Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, 2015.

[35] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[36] Joon Son Chung and Andrew Zisserman. Out of time: automated lip sync in the wild. In *ACCV*, 2016.

[37] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *CVPR*, 2017.

[38] Ramazan Cinbis, Jakob Verbeek, and Cordelia Schmid. Multi-fold MIL Training for Weakly Supervised Object Localization. In *CVPR*, 2014.

[39] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *PAMI*, 2016.

[40] B. Coifman, D. Beymer, P. Mclauchlan, and J. Malik. A realtime computer vision system for vehicle tracking and traffic surveillance. *Transportation Research C 6C*, 1998.

[41] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise

convolutional networks for semantic segmentation. *arXiv preprint arXiv:1503.01640*, 2015.

[42] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[43] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*, 2006.

[44] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.

[45] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on Graphics (TOG)*, 2012.

[46] Abhishek Das, Harsh Agrawal, C Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Human attention in visual question answering: Do humans and deep networks look at the same regions? In *EMNLP*, 2016.

[47] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[49] Emily L Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017.

[50] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010.

[51] Thang Ba Dinh, Nam Vo, and Gérard Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011.

[52] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised Visual Representation Learning by Context Prediction. In *ICCV*, 2015.

[53] P. Dollar and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[54] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini

Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[55] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, P. Hausser, C. Hazrba, V. Golkov, P. Smagt, D. Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[56] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[57] Ian Endres and Derek Hoiem. Category Independent Object Proposals. In *ECCV*, 2010.

[58] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

[59] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

[60] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

[61] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *ICCV*, 2017.

[62] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast Networks for Video Recognition. In *ICCV*, 2019.

[63] R. Fergus, P. Perona, and A. Zisserman. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *CVPR*, 2003.

[64] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *CVPR*, 2017.

[65] Katerina Fragkiadaki, Pablo Arbelaez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *CVPR*, 2015.

[66] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *ICCV*, 2015.

[67] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus

Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*, 2016.

[68] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling LSTMs and modality attention. In *ICCV*, 2019.

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

[70] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IROS*, 2017.

[71] Jin Gao, Haibin Ling, Weiming Hu, and Junliang Xing. Transfer learning based visual tracking with gaussian processes regression. In *ECCV*, 2014.

[72] R. Gao, R. Feris, and K. Grauman. Learning to separate object sounds by watching unlabeled video. In *ECCV*, 2018.

[73] Bernard Ghanem, Juan Carlos Niebles, Cees Snoek, Fabian Caba Heilbron, Humam Alwassel, Victor Escorcia, Ranjay Khrisna, Shyamal Buch, and Cuong Duc Dao. The ActivityNet large-scale activity recognition challenge 2018 summary. *arXiv preprint arXiv:1808.03766*, 2018.

[74] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[75] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video Action Transformer Network. In *CVPR*, 2019.

[76] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

[77] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[78] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *PAMI*, 2015.

[79] Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *NIPS*, 2018.

[80] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[82] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[83] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient Hierarchical Graph Based Video Segmentation. In *CVPR*, 2010.

[84] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

[85] Peng Han, Wenwu Yuan, Zhiwu Lu, and Ji-Rong Wen. Video detection by learning with deep representation and spatio-temporal context. 2015.

[86] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *ICCV Workshops*, 2019.

[87] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[88] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[89] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

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

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

[93] Richard Held and Alan Hein. Movement-produced stimulation in the development of

visually guided behavior. *Journal of comparative and physiological psychology*, 1963.

[94] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. CNN architectures for large-scale audio classification. In *ICASSP*, 2017.

[95] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[96] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

[98] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *PAMI*, 2015.

[99] Di Hu, Feiping Nie, and Xuelong Li. Deep multimodal clustering for unsupervised audiovisual learning. In *CVPR*, 2019.

[100] Qiyang Hu, Attila Szabo, Tiziano Portenier, Paolo Favaro, and Matthias Zwicker. Disentangling factors of variation by mixing them. In *CVPR*, 2018.

[101] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. In *ECCV*, 2016.

[102] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *CVPR*, 2016.

[103] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *ECCV*, 2008.

[104] Jinggang Huang and David Mumford. Statistics of natural images and models. In *CVPR*, 1999.

[105] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[106] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised

image-to-image translation networks. In *ECCV*, 2018.

[107] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *CVPR*, 2019.

[108] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[109] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[110] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, 2012.

[111] Jianwen Jiang, Yu Cao, Lin Song, Shiwei Zhang Yunkai Li, Ziyao Xu, Qian Wu, Chuang Gan, Chi Zhang, and Gang Yu. Human centric spatio-temporal action localization. In *ActivityNet Workshop on CVPR*, 2018.

[112] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2018.

[113] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[114] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016.

[115] M. Jones and D. Snow. Pedestrian detection using boosted features over many frames. In *ICPR*, 2008.

[116] Donggyu Joo, Doyeon Kim, and Junmo Kim. Generating a fusion image: One's identity and another's shape. In *CVPR*, 2018.

[117] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *PAMI*, 2012.

[118] Kai Kang, Hongsheng Li, Tong Xiao, Wanli Ouyang, Junjie Yan, Xihui Liu, and Xiaogang Wang. Object detection in videos with tubelet proposal networks. In *CVPR*, 2017.

[119] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *TCSVT*, 2017.

[120] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *ECCV*, 2016.

[121] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[122] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 2015.

[123] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[124] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition. In *ICCV*, 2019.

[125] Christian Keysers, Evelyne Kohler, M Alessandra Umiltà, Luca Nanetti, Leonardo Fogassi, and Vittorio Gallese. Audiovisual mirror neurons and action recognition. *Experimental brain research*, 2003.

[126] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 2014.

[127] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2019.

[128] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[129] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[130] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[131] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005*, 2019.

[132] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video

models from self-supervised synchronization. In *NIPS*, 2018.

[133] Bruno Korbar, Du Tran, and Lorenzo Torresani. SCSampler: Sampling salient clips from video for efficient action recognition. In *ICCV*, 2019.

[134] Philipp Krahenbuhl and Vladlen Koltun. Learning to propose objects. In *CVPR*, 2015.

[135] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. URL `https://arxiv.org/abs/1602.07332`.

[136] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[137] Hildegard Kuehne, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[138] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[139] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010.

[140] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *CVPR*, 2010.

[141] Junseok Kwon and Kyoung Mu Lee. Tracking by sampling trackers. In *ICCV*, 2011.

[142] Jean-Francois Lalonde and Alexei A Efros. Using color compatibility for assessing image realism. In *ICCV*, 2007.

[143] François Laurent, Mario Valderrama, Michel Besserve, Mathias Guillard, Jean-Philippe Lachaux, Jacques Martinerie, and Geneviève Florence. Multimodal information improves the rapid detection of mental fatigue. *Biomedical Signal Processing and Control*, 2013.

[144] Byungjae Lee, Enkhbayar Erdenee, Songguo Jin, Mi Young Nam, Young Giu Jung, and Phill Kyu Rhee. Multi-class multi-object tracking using changing point detection. In *ECCV*, 2016.

[145] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017.

[146] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang.

Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[147] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *CVPR*, 2011.

[148] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[149] Yang Li, Jianke Zhu, and Steven CH Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *CVPR*, 2015.

[150] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[151] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[152] Jon Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015.

[153] Xiang Long, Chuang Gan, Gerard De Melo, Jiajun Wu, Xiao Liu, and Shilei Wen. Attention clusters: Purely attention based local feature integration for video classification. In *CVPR*, 2018.

[154] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[155] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *NIPS*, 2016.

[156] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. In *CVPR*, 2015.

[157] Aravindh Mahendran, James Thewlis, and Andrea Vedaldi. Self-supervised segmentation by grouping optical-flow. In *ECCV*, 2018.

[158] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *ICLR*, 2016.

[159] Dominic W Massaro and Michael M Cohen. Tests of auditory–visual integration efficiency within the framework of the fuzzy logical model of perception. *The Journal of the*

Acoustical Society of America, 2000.

[160] Harry McGurk and John MacDonald. Hearing lips and seeing voices. *Nature*, 1976.

[161] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016.

[162] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[163] Arsha Nagrani, Samuel Albanie, and Andrew Zisserman. Seeing voices and hearing faces: Cross-modal biometric matching. In *CVPR*, 2018.

[164] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[165] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[166] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[167] Peter Ochs and Thomas Brox. Higher Order Motion Models and Spectral Clustering. In *CVPR*, 2012.

[168] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *PAMI*, 2014.

[169] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

[170] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017.

[171] Kei Omata and Ken Mogi. Fusion and combination in audio-visual integration. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2007.

[172] Dan Oneata, Jerome Revaud, Jakob Verbeek, and Cordelia Schmid. Spatio-temporal object detection proposals. In *ECCV*, 2014.

[173] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?

Weakly-supervised learning with convolutional neural networks. In *CVPR*, 2015.

[174] Wanli Ouyang, Ping Luo, Xingyu Zeng, Shi Qiu, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Yuanjun Xiong, Chen Qian, et al. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *arXiv preprint arXiv:1409.3505*, 2014.

[175] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *ECCV*, 2018.

[176] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.

[177] Mustafa Ozuysal, Vincent Lepetit, and Pascal Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009.

[178] Guillem Palou and Philippe Salembier. Hierarchical video representation with trajectory binary partition tree. In *CVPR*, 2013.

[179] M. Pandey and S.Lazebnik. Scene Recognition and Weakly Supervised Object Localization with Deformable Part-Based Models. In *ICCV*, 2011.

[180] A. Papazoglou and V. Ferrari. Fast Object Segmentation in Unconstrained Video. In *ICCV*, 2013.

[181] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollar. Exploring Weak Stabilization for Motion Feature Extraction. In *CVPR*, 2013.

[182] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, 2015.

[183] Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *CVPR*, 2015.

[184] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollar. Learning to segment object candidates. In *NIPS*, 2015.

[185] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011.

[186] B. Plummer, L. Wang, C. Cervantes, J. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-

sentence models. In *ICCV*, 2015.

[187] Gerasimos Potamianos, Chalapathy Neti, Guillaume Gravier, Ashutosh Garg, and Andrew W Senior. Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 2003.

[188] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

[189] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017.

[190] Avinash Ravichandran, Chaohui Wang, Michalis Raptis, and Stefano Soatto. Superfloxels: A mid-level representation for video sequences. In *ECCV Workshops*, 2012.

[191] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *CVPR*, 2017.

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

[193] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *ICML*, 2014.

[194] Scott Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *NIPS*, 2015.

[195] Konstantinos Rematas, Chuong H Nguyen, Tobias Ritschel, Mario Fritz, and Tinne Tuytelaars. Novel views of objects from a single image. *TPAMI*, 2017.

[196] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[197] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. *ECCV*, 2016.

[198] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 2004.

[199] Daniel L Ruderman. The statistics of natural images. *Network: computation in neural systems*, 5(4):517–548, 1994.

[200] Jean-Luc Schwartz, Frederic Berthommier, and Christophe Savariaux. Audio-visual scene analysis: evidence for a "very-early" integration process in audio-visual speech perception. In *International Conference on Spoken Language Processing*, 2002.

[201] Arda Senocak, Tae-Hyun Oh, Junsik Kim, Ming-Hsuan Yang, and In So Kweon. Learning to localize sound source in visual scenes. In *CVPR*, 2018.

[202] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[203] Gilad Sharir and Tinne Tuytelaars. Video object proposals. In *CVPR Workshops*, 2012.

[204] J. Shi and J. Malik. Motion Segmentation and Tracking Using Normalized Cuts. In *ICCV*, 1998.

[205] Abhinav Shrivastava and Abhinav Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016.

[206] Zhixin Shu, Mihir Sahasrabudhe, Riza Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance. In *ECCV*, 2018.

[207] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[208] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *NIPS*, 2014.

[209] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[210] K. K. Singh, F. Xiao, and Y. J. Lee. Track and transfer: Watching videos to simulate strong human supervision for weakly-supervised object detection. In *CVPR*, 2016.

[211] Krishna Kumar Singh and Yong Jae Lee. End-to-end localization and ranking for relative attributes. In *ECCV*, 2016.

[212] P. Siva, C. Russell, and T. Xiang. In Defence of Negative Mining for Annotating Weakly Labelled Data. In *ECCV*, 2012.

[213] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *ACL*, 2013.

[214] H. O. Song, Y. J. Lee, S. Jegelka, and T. Darrell. Weakly-supervised Discovery of Visual Pattern Configurations. In *NIPS*, 2014.

[215] Khurram Soomro, Amir Roshan Zamir, and M Shah. A dataset of 101 human action classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[216] Salvador Soto-Faraco, Daria Kvasova, Emmanuel Biau, Nara Ikumi, Manuela Ruzzoli, Luis Mors-Fernndez, and Mireia Torralba. *Multisensory Interactions in the Real World*. Elements in Perception. Cambridge University Press, 2019. doi: 10.1017/9781108578738.

[217] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[218] Barry E Stein. *The new handbook of multisensory processing*. MIT Press, 2012.

[219] Hao Su, Fan Wang, Li Yi, and Leonidas Guibas. 3d-assisted image feature synthesis for novel views of an object. In *ICCV*, 2015.

[220] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. Hierarchical feature aggregation networks for video action recognition. *arXiv preprint arXiv:1905.12462*, 2019.

[221] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. FBK-HUPBA Submission to the EPIC-Kitchens 2019 Action Recognition Challenge. *arXiv preprint arXiv:1906.08960*, 2019.

[222] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, 2018.

[223] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 2019.

[224] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*, 2010.

[225] J. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013.

[226] Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *arXiv:1412.1441*, 2014.

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

[228] Michael W Tao, Micah K Johnson, and Sylvain Paris. Error-tolerant image compositing. In *ECCV*, 2010.

[229] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural Computation*, 2000.

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

[231] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[232] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[233] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *arXiv preprint arXiv:1904.02811*, 2019.

[234] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

[235] Subarna Tripathi, Zachary C Lipton, Serge Belongie, and Truong Nguyen. Context matters: Refining object detection in video with recurrent neural networks. *arXiv preprint arXiv:1607.04648*, 2016.

[236] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[237] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. Multiple Hypothesis Video Segmentation from Superpixel Flows. In *ECCV*, 2010.

[238] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[239] Paul Viola, Michael Jones, and Daniel Snow. Detecting Pedestrians Using Patterns of

Motion and Appearance. *IJCV*, 2005.

[240] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *CVPR*, 2016.

[241] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *ICCV*, 2015.

[242] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[243] Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. Structured matching for phrase localization. In *ECCV*, 2016.

[244] S. Wang, J. Joo, Y. Wang, and S. C. Zhu. Weakly supervised learning for attribute localization in outdoor scenes. In *CVPR*, 2013.

[245] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[246] Wenguan Wang, Jianbing Shen, and Fatih Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015.

[247] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

[248] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[249] M. Weber, M. Welling, and P. Perona. Unsupervised Learning of Models for Recognition. In *ECCV*, 2000.

[250] Longyin Wen, Dawei Du, Zhen Lei, Stan Z Li, and Ming-Hsuan Yang. Jots: Joint online tracking and segmentation. In *CVPR*, 2015.

[251] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 1997.

[252] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, 2019.

[253] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[254] F. Xiao and Y. J. Lee. Discovering the spatial extent of relative attributes. In *ICCV*, 2015.

[255] F. Xiao and Y. J. Lee. Track and segment: An iterative unsupervised approach for video object proposals. In *CVPR*, 2016.

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

[257] Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. Weakly-supervised visual grounding of phrases with linguistic structures. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[258] Fanyi Xiao, Yong Jae Lee, Kristen Grauman, Jitendra Malik, and Christoph Feichtenhofer. Audiovisual slowfast networks for video recognition. *arXiv preprint arXiv:2001.08740*, 2019.

[259] Fanyi Xiao, Haotian Liu, and Yong Jae Lee. Identity from here, pose from there: Self-supervised disentanglement and generation of objects using unlabeled videos. In *International Conference on Computer Vision (ICCV)*, 2019.

[260] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.

[261] B. Xiong, S. Jain, and K. Grauman. Pixel objectness: Learning to segment generic objects automatically in images and videos. *PAMI*, 2018.

[262] C. Xu, C. Xiong, and J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012.

[263] Chenliang Xu and Jason J Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012.

[264] Jia Xu, Alexander G Schwing, and Raquel Urtasun. Tell me what you see and i will show you where it is. In *CVPR*, 2014.

[265] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with

visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[266] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Craft objects from images. In *CVPR*, 2016.

[267] Jimei Yang, Scott Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *NIPS*, 2015.

[268] Shunyu Yao, Tzu-Ming Harry Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, William T. Freeman, and Joshua B. Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *NIPS*, 2018.

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

[270] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1710.10916*, 2017.

[271] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *ECCV*, 2016.

[272] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[273] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. In *ECCV*, 2018.

[274] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *CVPR*, 2015.

[275] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, 2012.

[276] Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167*, 2015.

[277] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[278] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View

synthesis by appearance flow. In *ECCV*, 2016.

[279] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ECCV*, 2017.

[280] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. *ICCV*, 2017.

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

[282] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.

[283] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.

[284] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. ECO: Efficient convolutional network for online video understanding. In *ECCV*, 2018.